$NLogSpace < PTIME$

Michael A. Taitslin *†

Department of Computer Science
Tver State University
Tver, 170000, Russia
E-mail: Michael.Taitslin@tversu.ru

Abstract

We propose a problem in PTIME which, we believe, is not in NLogSpace. If it is the case, it would settle a long staying open problem of Stephen A.Cook posted in 1969 (see [1]).

1. Introduction

One of the deepest results in General Complexity theory is the well known Cook's Theorem (see [2]):

"The following three conditions are equivalent for any set A of strings on an alphabet Γ and for any function $L(n) \geqslant log_2 n$ on the positive integers.

(a) A is accepted by some deterministic auxiliary PDM within storage $L(n)$.

(b) A is accepted by some nondeterministic auxiliary PDM within storage $L(n)$.

(c) A is accepted by some time bounded computer within time $T(n) = 2^{cL(n)}$ for some constant c."

In [2], there is the following paragraph:

"The major open question concerning these results is whether the pushdown tape really adds to the computing power of auxiliary PDM's. That is, we cannot show that Theorem 1 becomes false if "auxiliary PDM" is replaced by "Turing machine" in parts (a) and (b). This modified Theorem 1 would assert the converse of Corollary 5; that is, a set is accepted by a time-bounded computer within time $2^{cL(n)}$ if and only if it is accepted by a Turing machine within storage $L(n)$."

See also [1], [3], and [4].

We propose a problem in PTIME which, we believe, is not in NLogSpace.

Let

$$\vartheta_1, \vartheta_2, \ldots, \vartheta_{(LAST+1)^3}$$

be the enumeration of the 3-elements sequences of the elements of set

$$|\mathcal{A}| = \{0,\ 1, \ldots, LAST\}$$

in increasing order (mean,

$$\vartheta_1 = \langle 0,\ 0,\ 0\rangle,$$

$$\vartheta_2 = \langle 0,\ 0,\ 1\rangle, \ldots,$$

$$\vartheta_{(LAST+1)^3} = \langle LAST,\ LAST,\ LAST\rangle).$$

We consider finite structures

$$\mathcal{A} = (\{0, 1, \ldots, LAST\},\ f,\ \ a,\ c,\ ',\ 0,\ LAST)$$

of signature

$$\Omega = \langle f^{(2)},\ \ a,\ c,\ ',\ 0,\ LAST\rangle$$

where $LAST$ is a positive natural number, a and c are elements of $\{0, 1, \ldots, LAST\}$, f is a binary operation on $\{0, 1, \ldots, LAST\}$, i' is $i + 1$ (the next natural number) if the natural number i is less than $LAST$, and

$$LAST' = LAST.$$

Such a structure is called **standard**.

*The address for correspondence: 40 Dmitry Donskoi St., apt.2, Tver, 170006, Russia

†The work of the author was partially supported by the Russian Foundation of Basic Research (project code: 04-01-00015).

The **standard** presentation of a standard structure is the following word in alphabet $\{0,\ 1,\ *\}$:

$$\#(LAST) * \#(a) * \#(c) * \#(f)$$

where for a natural number i,

$$\#(i)$$

is the binary presentation of i, and

$$\#(f)$$

is the code of f.

A sequence

$$\xi = \xi_1 \xi_2 \dots \xi_\ell$$

of 0 and 1 of length $\ell = (LAST+1)^3$ is the code of f iff for positive natural $u \leqslant (LAST+1)^3$,

$$\xi_u = \begin{cases} 1, \text{ if } \vartheta_u \in graph(f), \\ 0, \text{ if } \vartheta_u \notin graph(f). \end{cases}$$

Here for any $i, j, k \in \{0, 1, \dots, LAST\}$, $(i, j, k) \in graph(f)$ iff $f(i, j) = k$.

The problem is: for the standard presentation of a given standard structure, define whether or not c belongs to the sub-groupoid generated by a in

$$(\{0, 1, \dots, LAST\},\ f).$$

It is trivial (and well known) that the problem is in $PTIME$. We believe that the problem is not in $NLogSpace$.

2. Program schemes

Sometimes, we write (uv) instead of $f(u, v)$.

Definition 1. *A test is an expression of one of the forms either $v_1 = v_2$, or $v_1 \neq v_2$, where v_1 and v_2 are variables. Variables v_1 and v_2 occur in the test. Other variables do not occur in the test.*

A program scheme in the signature Ω is:

(1) $x \leftarrow a$, where x is a variable; a variable is used in the program scheme iff the variable is x;

(2) $x \leftarrow c$, where x is a variable; a variable is used in the program scheme iff the variable is x;

(3) $x \leftarrow x'$, where x is a variable; a variable is used in the program scheme iff the variable is x;

(4) $x \leftarrow 0$, where x is a variable; a variable is used in the program scheme iff the variable is x;

(5) $x \leftarrow LAST$, where x is a variable; a variable is used in the program scheme iff the variable is x;

(6) $x \leftarrow (yz)$, where x, y and z are different variable; a variable is used in the program scheme iff the variable is x, y, or z;

(7) $x \leftarrow y$, where x and y are variables; a variable is used in the program scheme iff the variable is either x, or y;

(8) `if` φ `then` Π_1 `else` Π_2 `fi`, *where φ is a test, and Π_1, Π_2 are program schemes; a variable is used in the program scheme iff either the variable is used in Π_1 or in Π_2, or the variable occurs in φ;*

(9) `while` φ `do` Π `od`, *where φ is a test, and Π is a program scheme; a variable is used in the program scheme iff either the variable is used in Π, or the variable occurs in φ;*

(10) $\Pi_1; \Pi_2$, where Π_1, Π_2 are program schemes; a variable is used in the program scheme iff the variable is used in Π_1 or in Π_2;

(11) Π_1 `OR` *Π_2, where Π_1, Π_2 are program schemes; a variable is used in the program scheme iff the variable is used in Π_1 or in Π_2.*

A program scheme is called an assignment iff the program scheme has one of the form (1)—(7).

A program scheme is called a program scheme in X iff X is the set of all the variables used in the program scheme.

Let Π be a program scheme in X_1, X be a set of variables, $X \supseteq X_1$, and let $\mathcal{M}$ be a structure of the signature Ω. This Π defines a binary relation on the set of all X-valuations for $\mathcal{M}$ (i.e., all mappings $\sigma : X \longrightarrow |\mathcal{M}|$), as follows.

(12) $(x \leftarrow t)(\sigma_1, \sigma_2)$ iff $\sigma_1(t) = \sigma_2(x)$, and $\sigma_1(y) = \sigma_2(y)$, for any $y \in X$ different from x;

(13) $(\texttt{if}\ \varphi\ \texttt{then}\ \Pi_1\ \texttt{else}\ \Pi_2\ \texttt{fi})(\sigma_1, \sigma_2)$ *iff either*

$\sigma_1 \models \varphi$ *and* $\Pi_1(\sigma_1, \sigma_2)$, *or*

$\sigma_1 \models \neg\varphi$ *and* $\Pi_2(\sigma_1, \sigma_2)$;

(14) $(\texttt{while}\ \varphi\ \texttt{do}\ \Pi\ \texttt{od})(\sigma_1, \sigma_2)$ *iff either*

$\sigma_1 \models \neg\varphi$ *and* $\sigma_1 = \sigma_2$, *or*

$\sigma_1 \models \varphi$ *and there exists* σ_3 *such that* $\Pi(\sigma_1, \sigma_3)$ *and*

$$(\texttt{while}\ \varphi\ \texttt{do}\ \Pi\ \texttt{od})(\sigma_3, \sigma_2).$$

(15) $(\Pi_1; \Pi_2)(\sigma_1, \sigma_2)$ *iff there exists* σ_3 *such that*

$\Pi_1(\sigma_1, \sigma_3)$ *and* $\Pi_2(\sigma_3, \sigma_2)$*;*

(16) $(\Pi_1\ \texttt{OR}\ \Pi_2)(\sigma_1, \sigma_2)$ *iff* $\Pi_1(\sigma_1, \sigma_2)$ *or* $\Pi_2(\sigma_1, \sigma_2)$.

For any set $X \supseteq X_1$ *of variables, any program scheme in* X_1 *defines a global predicate of the arity* $2\,|X|$ *where* $|X|$ *is the number of the elements of* X.

Namely, if $X = \{x_1, \ldots, x_n\}$, *for*

$$a_1, \ldots, a_n, b_1, \ldots, b_n \in |\mathcal{M}|,$$

$$\Pi(a_1, \ldots, a_n, b_1, \ldots, b_n) \text{ iff } \Pi(\sigma_1, \sigma_2)$$

where $\sigma_1(x_i) = a_i$ *and* $\sigma_2(x_i) = b_i$ *for* $i = 1, \ldots, n$.

For X*-valuation* σ_1 *for* Ω*-structure* $\mathcal{M}$ *and program scheme* Π *in signature* Ω, *we say that* Π *stops on* σ_1 *iff there is an* X*-valuation* σ_2 *such that* $\Pi(\sigma_1, \sigma_2)$.

Sometimes, we write $(\sigma_1 \Pi \sigma_2)$ *instead of* $\Pi(\sigma_1, \sigma_2)$.

A program scheme Π in $X = \{x_1, \ldots, x_n\}$ is called *closed* iff Π is $\Pi_1; \Pi_2$ where Π_1 is

$$x_1 \leftarrow b_1; \ldots; x_n \leftarrow b_n$$

for some $b_1, \ldots, b_n \in \{a, c, 0, LAST\}$. A closed program scheme Π stops on a structure $\mathcal{M}$ of the signature Ω iff Π stops on an X-valuation for Ω-structure $\mathcal{M}$. Two closed program schemes Π_1 and Π_2 are called equivalent iff for any Ω-structure $\mathcal{M}$, Π_1 stops on $\mathcal{M}$ iff Π_2 stops on $\mathcal{M}$.

A global predicate is a set of standard structures of signature Ω. A global predicate is in $NLogSpace$ iff there is a Turing machine M in $NLogSpace$ such that M accepts the standard presentation of a standard structure iff the structure is in the global predicate.

The following result is trivial and can be found in [7].

Theorem 1 ([7], Lemma 3.3.3)**.** *For any global predicate from* $NLogSpace$*, there is a closed program scheme of signature* Ω *such that the scheme stops on a standard structure iff the structure belongs to the global predicate.*

Thus, it is enough to prove that there is no closed program scheme of signature Ω such that the scheme stops on a standard structure iff in the structure, c belongs to the sub-groupoid generated by a in

$$(\{0, 1, \ldots, LAST\},\ f).$$

A test in X is an expression of one of the forms either $v_1 = v_2$, or $v_1 \neq v_2$ where v_1 and v_2 are variables from X. A test of the form $v_1 = v_2$ is positive. A test of the form $v_1 \neq v_2$ is negative.

The meaning of a test is as following: the test does not change the value of any variable; the next X-valuation exists and is the given X-valuation iff the test holds for the given X-valuation. Any test in X can be presented as a program scheme in X. Indeed, test φ can be represent as

$$\texttt{while}\ \neg\varphi\ \texttt{do}\ v \leftarrow v\ \texttt{od}$$

where v is a variable from X.

A linear program in $z_1, \ldots, z_m$ is an assignment in $z_1, \ldots, z_m$ and the length of the linear program is 1, a test in $z_1, \ldots, z_m$ and the length of the linear program is 1, or an expression of the form $\Pi_1; \Pi_2$ where Π_1 and Π_2 are linear programs in $z_1, \ldots, z_m$. In the last case, the length of the linear program is the sum of the lengths of Π_1 and Π_2.

A program in $z_1, \ldots, z_m$ is a union of linear programs in $z_1, \ldots, z_m$. The program is closed iff it is a union of closed linear programs in $z_1, \ldots, z_m$. The program is regular iff it is a regular language in the alphabet of all the used assignments and tests.

Any linear program in $z_1, \ldots, z_m$ is a program scheme. Let $X = \{z_1, \ldots, z_m\}$. For a standard Ω-structure $\mathcal{M}$, this program scheme defines the binary relation on the set of all X-valuations for $\mathcal{M}$. The same binary relation defines this linear program. A program defines the binary relation on the set of all X-valuations for $\mathcal{M}$ which is the union of the binary relations defined by the linear programs belonging to the program. In details, for any two X-valuations σ_1

and σ_2 and program Π, $(\sigma_1 \Pi \sigma_2)$ iff there is a linear program Π_1 in Π such that $(\sigma_1 \Pi_1 \sigma_2)$.

The following theorem is trivial.

Theorem 2 (see [7], Proposition 1.2.2)**.** *For any program scheme Π in $X = \{z_1, \ldots, z_m\}$, there is a regular program Π_1 in X such that for any standard Ω-structure $\mathcal{M}$, Π and Π_1 define the same binary relation on the set of all X-valuations for $\mathcal{M}$.*

If the program scheme Π in $z_1, \ldots, z_m$ is closed, then the program Π_1 in $z_1, \ldots, z_m$ can also be token closed.

A program Π in $X = \{z_1, \ldots, z_m\}$ and a program scheme Π_1 in X are called strong equivalent iff for any standard Ω-structure $\mathcal{M}$, Π and Π_1 define the same binary relation on the set of all X-valuations for $\mathcal{M}$.

A closed program stops on a standard $\mathcal{M}$ iff there is a linear program in it which stops on $\mathcal{M}$. A closed program and a closed program scheme are equivalent iff for any standard Ω-structure $\mathcal{M}$, either the program and the program scheme both stop on $\mathcal{M}$, or the program and the program scheme both do not stop on $\mathcal{M}$.

3. Idempotent quasigroups

A groupoid $(G, \circ)$ is said to be idempotent if

$$(x \circ x) = x$$

for all $x \in G$.

A groupoid $(G, \circ)$ is said to be quasigroup iff for any $a, b \in G$, the equations

$$(a \circ x) = b, \ \ (y \circ a) = b$$

are each solvable in $(G, \circ)$ and the solutions are unique.

Four-element idempotent quasigroup $(Q_4, \circ)$ has elements 1, 2, 3, and 4. The operation $\circ$ in the quasigroup is defined by the following rules:

$$(1 \circ 1) = (2 \circ 3) = (3 \circ 4) = (4 \circ 2) = 1;$$

$$(2 \circ 2) = (1 \circ 4) = (4 \circ 3) = (3 \circ 1) = 2;$$

$$(3 \circ 3) = (1 \circ 2) = (2 \circ 4) = (4 \circ 1) = 3;$$

$$(4 \circ 4) = (2 \circ 1) = (1 \circ 3) = (3 \circ 2) = 4.$$

Let $\circ_1$ is $\circ$.

Suppose idempotent quasigroup $(Q_{4^i}, \circ_i)$ containing 4^i elements was defined. Let $a_1, a_2, \ldots, a_{4^i}$ be all the elements of Q_{4^i}. Consider four isomorphic copies of $(Q_{4^i}, \circ_i)$. Suppose any two of them have no common element. For $j = 1, 2, 3, 4$, let τ_j be an isomorphism of $(Q_{4^i}, \circ_i)$ and the copy of number j. Let $Q_{4^{i+1}}$ be the union of all the elements of the copies. Then $Q_{4^{i+1}}$ contains 4^{i+1} elements. For $k = 1, 2, \ldots, 4^i$ and $j = 1, 2, 3, 4$, let $a_{k,j}$ denote $\tau_j(a_k)$. Then for $k_1, k_2 \in \{1, 2, \ldots, 4^i\}$ and $j_1, j_2 \in \{1, 2, 3, 4\}$,

$$(a_{k_1, j_1} \circ_{i+1} a_{k_2, j_2}) = \tau_{(j_1 \circ j_2)}(a_{k_1} \circ_i a_{k_2}).$$

It is easy to see that $(Q_{4^{i+1}}, \circ_{i+1})$ is an idempotent quasigroup.

So we defined idempotent quasigroup $(Q_{4^i}, \circ_i)$ for any positive integer number i.

For any positive integer number i, any element a of quasigroup $(Q_{4^i}, \circ_i)$ defines an 1-1 mapping ρ_a from Q_{4^i} onto Q_{4^i} such that for any $b \in Q_{4^i}$, $(b \circ_i \rho_a(b)) = a$.

In future, we omit subscript i and write $\circ$ instead of $\circ_i$ if it is understandable what i is meant.

4. Register complexity of terms

Definition 2. *Fix a variable list $X = z_1, \ldots, z_m$. Fix a positive integer number I. Consider set Q_{4^I}.*

To every sequence α of assignments using variables only from X, the only operation symbol f and constant symbols only from Q_{4^I} (such an assignment has one of the forms $u \leftarrow v$, $u \leftarrow a$, or $u \leftarrow (vw)$ where $u, v, w \in X$ and $a \in Q_{4^I}$), we assign a mapping $\mu(\alpha)$ from X to the set of all the $\langle f, \ a \mid a \in Q_{4^I} \rangle$-terms. This mapping is defined by induction on the length s of sequence α.

If α is empty (mean, $s = 0$), then $\mu(\alpha)(x) = x$ for any $x \in X$.

If α is $\alpha'; \alpha_s$, then one of the three following cases takes place.

Case 1. If α_s is $x_i \leftarrow x_j$, then

$$\mu(\alpha)(y) = \mu(\alpha')(y)$$

for $y \in X \setminus \{x_i\}$, and

$$\mu(\alpha)(x_i) = \mu(\alpha')(x_j).$$

Case 2. If α_s is $x_i \leftarrow a$ where a is a constant symbol from Q_{4^I}, then

$$\mu(\alpha)(y) = \mu(\alpha')(y)$$

for $y \in X \setminus \{x_i\}$, and

$$\mu(\alpha)(x_i) = a.$$

Case 3. If α_s is $x_i \leftarrow (x_{j_1} x_{j_2})$, then

$$\mu(\alpha)(y) = \mu(\alpha')(y)$$

for $y \in X \setminus \{x_i\}$ and

$$\mu(\alpha)(x_i) = (\mu(\alpha')(x_{j_1})\mu(\alpha')(x_{j_2})).$$

Such a sequence α of assignments computes a term t in variable $y \in X$ iff t is $\mu(\alpha)(y)$.

Let such a sequence $\alpha = \alpha_1; \dots; \alpha_s$ of assignments compute a term t in variable $y \in X$. For $i = 1, \dots, s$, the assignment α_i is essential in the computation iff α_i is $x \leftarrow r$, and $\mu(\alpha_{i+1}; \dots; \alpha_s)(y)$ contains x. If $i = s$, it means that x coincides with y.

The register complexity of a term t (denoted $\mathrm{rc}(t)$) is the least number n such that t is computed by a sequence of assignments using at most n variables.

Example 1. Term

$$((a(aa))(((aa)(aa))(((aa)(aa))((aa)(aa)))))$$

is not computed by any sequence of assignments using at most two variables.

Example 2. Term

$$((a(aa))((a(aa))((a(aa))(a(aa)))))$$

can be computed using only two variables.

By $g(i)$ we denote terms defined inductively as following:

$$g(0) = x_1, \;\; g(i+1) = (g(i)g^*(i))$$

where for $s = 2^i$, $g^*(i) = g(i)(x_{s+1}, \dots, x_{2s})$.

For instance,

$$g(1) = (x_1x_2), \;\; g(2) = ((x_1x_2)(x_3x_4)),$$

$$g(3) = (((x_1x_2)(x_3x_4))((x_5x_6)(x_7x_8))),$$

$$g(4) = ((((x_1x_2)(x_3x_4))((x_5x_6)(x_7x_8))) \\ (((x_9x_{10})(x_{11}x_{12}))((x_{13}x_{14})(x_{15}x_{16}))))$$

and so on.

It is usual to represent $\langle f, \; a \mid a \in Q_{4^I} \rangle$-terms by finite binary trees. Then $g(i)$ is the full binary tree of depth i whose different leaves are marked by different variables.

For any $a \in Q_{4^I}$, if every leaf of the full binary tree of depth i is marked by a, then the tree represents the term denoted by $(\mathfrak{i})_a$. Thus,

$$(\mathfrak{i})_a = g(i)(a, \dots, a)$$

($(\mathfrak{i})_a$ is obtained from $g(i)$ by substitution a for each variable).

It is obvious that the register complexity of $(\mathfrak{i})_a$ is 1.

In future, $\mathfrak{i}$ denotes one of $(\mathfrak{i})_a$ for some $a \in Q_{4^I}$. So $\mathfrak{i}$ denotes any one of 4^I terms.

Definition 3. *$\langle f, \; a \mid a \in Q_{4^I} \rangle$-terms t_1 and t_2 are called independent iff they are different.*

Let $s = 2r$ and $r = 2^i$.

$\langle f, \; a \mid a \in Q_{4^I} \rangle$-terms $t_1, \dots, t_s$ are called independent iff the following conditions hold:

- *terms $t_1, \dots, t_r$ are independent;*
- *terms $t_{r+1}, \dots, t_s$ are independent;*
- *for every subterm t' of term $g(i)$, the term*

  $$t'(t_1, \dots, t_r)$$

  differs from $t_{r+1}, \dots, t_s$;
- *for every subterm t' of term $g(i)$, the term*

  $$t'(t_{r+1}, \dots, t_s)$$

  differs from $t_1, \dots, t_r$.

Theorem 3 (see [5], [6]).

Let $s = 2^i$ and $\langle f, \; a \mid a \in Q_{4^I} \rangle$-terms

$$t_1, \dots, t_s$$

be independent.

Then $\mathrm{rc}(g(i)(t_1, \dots, t_s)) \geqslant i + 1$.

Example 3. For any natural number $i > 2$, terms

$$\mathfrak{1}, \; \mathfrak{2}, \dots, \; \mathfrak{s}$$

where $s = 2^i$ are independent.

By Theorem 3, $\mathrm{rc}(g(i)(\mathfrak{1}, \; \mathfrak{2}, \dots, \; \mathfrak{s})) \geqslant i + 1$.

5. $NLogSpace < PTIME$

Theorem 4. *There is no closed program scheme of signature*

$$\Omega = \langle f^{(2)},\ a,\ c,\ ',\ 0,\ LAST\rangle$$

such that for any standard structure

$$(G,\ f,\ a,\ c,\ ',\ 0,\ LAST) \qquad \text{(A)}$$

of the signature Ω, the given program scheme stops on the structure iff c belongs to the substructure generated by a in $(G,\ f)$.

Proof. By contradiction. Suppose such a closed program scheme Π exists.

It means that for any standard structure (A) of the signature Ω, Π stops on the structure iff c belongs to the substructure generated by a in $(G,\ f)$.

Moreover, the following lemma holds.

Lemma 5 (see [7], Lemma 1.3.4)**.** *Any closed program scheme is equivalent to a closed program scheme of the form*

$$\Pi';\ \texttt{while}\ \varphi\ \texttt{do}\ \Pi_*\ \texttt{od}$$

where φ is a test, Π' is a sequence of assignments, and Π_* is a program scheme without `while`.

Let Π uses variables $z_1, \ldots, z_n$ only. There is a closed regular program Π_1 in $z_1, \ldots, z_n$ over signature Ω such that Π and Π_1 define the same binary relation on the set of all $\{z_1, \ldots, z_n\}$-valuations for any standard structure (see Theorem 2).

Any program scheme without `while` is strong equivalent to a program which is a finite union of linear programs. Thus, we suppose that for linear programs $\Pi^1, \ldots, \Pi^q$ in $z_1, \ldots, z_n$, program Π_1 has the form

$$\Pi'; (\varphi; (\Pi^1 + \cdots + \Pi^q);)^* \neg\varphi$$

where $+$ denotes the union, and * denotes the set iteration. We assume that $q < n$ and that for $i = 1,\ldots, q$, the length of Π^i is less than n.

A linear program (program, or program scheme) of the signature Ω is called *canonical* iff for some variables u and v, the linear program (program, or program scheme, respectively) is $u \leftarrow a; v \leftarrow c; \Pi_*$, and all the assignments in linear program (program, or program scheme, respectively) Π_* have one of the forms $x \leftarrow (yz)$, $x \leftarrow x'$, $x \leftarrow y$, $x \leftarrow 0$, or $x \leftarrow LAST$ where x, y, and z are different variables.

It is trivial that any program scheme is equivalent to a canonical program scheme. If the given program scheme is closed then an equivalent canonical program scheme can also be token closed. Thus, without loss of generality, we suppose that the program Π_1 is canonical.

For a linear program Ξ in $X = \{z_1, \ldots, z_n\}$, we say that for a given X-valuation σ, $\Xi(\sigma)$ is NULL iff Ξ does not stop on σ.

Consider a linear program

$$\alpha_1; \ldots; \alpha_p$$

in $X = \{z_1, \ldots, z_n\}$.

For any $i = 1, 2, \ldots, p$ and any $j = 1, 2, \ldots, n$, linear program

$$\alpha_1; \ldots; \alpha_i$$

assigns a value

$$\mu(\alpha_1; \ldots; \alpha_i)(z_j)$$

to z_j. The value $\mu(\alpha_1; \ldots; \alpha_i)(z_j)$ is either NULL, or a term over a, c, 0, and $LAST$.

In details, first, all the variables receive values from

$$\{a,\ c,\ 0,\ LAST\}.$$

The given X-valuation is not NULL.

If the given X-valuation is NULL, then the value of any variable is NULL and the next X-valuation is NULL for any test and any assignment.

Suppose that the given X-valuation is not NULL.

Tests do not change the values of the variables. The next X-valuation is the given X-valuation iff the test holds for the given X-valuation. Otherwise the next X-valuation is NULL.

An assignment $x \leftarrow u$ where u is a term does not change the value of any variable different from x and the next X-valuation is not NULL. Assignment $x \leftarrow 0$ assigns value 0 to x. Assignment $x \leftarrow LAST$ assigns value $LAST$ to x. Assignment $x \leftarrow x'$ assigns value t' to x where t is the old value of x. Assignment $x \leftarrow y$ assigns value t to x where t is the value of y. Assignment $x \leftarrow (yz)$ assigns value $(t_1 t_2)$ to x where t_1 is the value of y, and t_2 is the value of z.

Consider the following class $\mathfrak{K}$ of standard structures. Any structure from $\mathfrak{K}$ is a standard structure

$$(E_{nJ},\ f_{nJ},\ a,\ c,\ ',\ 0,\ LAST). \tag{B}$$

for a positive natural number J.

Q_{4^I}-terms are all the elements of Q_{4^I}, and any expression (t_1t_2) where t_1 and t_2 are Q_{4^I}-terms.

Q_{4^I}-terms are some words in alphabet

$$\{d,\ (,\ )\mid d\in Q_{4^I}\}.$$

For $d\in Q_{4^I}$, the length of d is 1. The length of (t_1t_2) is the sum of the length of t_1, the length of t_2, and 2. So the length of an Q_{4^I}-term is the length of the word in alphabet $\{d,\ (,\ )\mid d\in Q_{4^I}\}$.

For a positive natural number J and $s=2^{nJ}$, consider the set G_{nJ} of all the subterms of terms $g(nJ)(\mathfrak{1},\dots,\mathfrak{s})$.

Let for t_1 and t_2 from G_{nJ},

$$f_{nJ}(t_1,t_2)=\begin{cases}(t_1t_2), \text{ if } (t_1t_2) \text{ is in } G_{nJ},\\ t_1 \text{ otherwise.}\end{cases}$$

Let k be a natural number. Let ℓ be a natural number, $s_\ell=2^k(\ell+1)$, and $r_\ell=2^k\ell+1$.

For $b\in Q_{4^I}$, let $\bar{b}^{k,\ell}$ be the term

$$g(k)((\mathfrak{r}_\ell)_b,\ \dots,\ (\mathfrak{s}_\ell)_b).$$

For $s=2^{nJ}$, consider the congruence $=$ on $(G_{nJ},\ f_{nJ})$ defined by the following rules:

- for $d,b\in Q_{4^I}$, $k=0,\ 1,\ \dots,\ nJ-1$, and for any even natural number $\ell<2^{nJ-k}$,

$$(\bar{d}^{k,\ell}\ \bar{b}^{k,\ell+1})=\overline{(d\circ b)}^{\,k+1,\ell/2}.$$

Thus, for $k=0,\ 1,\ \dots,\ nJ-1$, and for any even natural number $\ell<2^{nJ-k}$, any term

$$g(k+1)(\mathfrak{r}_\ell,\ \dots,\ \mathfrak{s}_{\ell+1})$$

is equivalent to one of pairwise non-equivalent terms

$$\{\bar{d}^{k+1,\ell/2}\mid d\in Q_{4^I}\}.$$

For that k and ℓ, we say that term $\bar{d}^{k+1,\ell/2}$ is situated on $(\ell/2)$th node of level $(nJ-k-1)$.

$(E_{nJ},\ f_{nJ})$ denotes the factor-structure of

$$(G_{nJ},\ f_{nJ})$$

modulo $=$.

In the rest of the paper, we fix J, fix $a\in Q_{4^I}$, and put $m=nJ$ and $s=2^{nJ}$.

As any structure (B) from $\mathfrak{K}$ is standard, E_m is

$$\{0,\ 1,\ \dots,\ LAST\}.$$

In other words, there is an enumeration of E_m by all the numbers from $\{0,\ 1,\ \dots,\ LAST\}$. Thus, any class modulo $=$ of Q_{4^I}-terms from G_m has the number in the enumeration. We attach the number of the class modulo $=$ of Q_{4^I}-terms from G_m to any Q_{4^I}-term from the class. Thus, any Q_{4^I}-term from G_m has the number.

Let us compute $LAST$. There are $(i+1)$ different subterms of $\mathfrak{i}$. So there are exactly

$$4^I(1+2^{nJ}+2^{nJ-1}\cdots+2+1)=4^I2^{nJ+1}$$

all the elements of E_m. Denote $F(nJ)=2^{nJ+1}$. Thus $LAST=4^IF(nJ)-1$.

We will consider only *acceptable* enumerations of E_m.

An enumeration of E_m is called acceptable iff it can be produced by such a way.

First we are numbering all the elements of set $b\mid b\in Q_{4^I}\}$. After that we are numbering all the terms from

$$\{(\mathfrak{i})_b\mid b\in Q_{4^I}\}$$

for $i=1,2,\dots,s$.

For that, we consider a permutation ρ of $\{1,2,\dots,s\}$. First we are numbering all the terms when $i=\rho(1)$. Then we are numbering all the terms when $i=\rho(2)$. Then we are numbering all the terms when $i=\rho(3)$, and so on. Finally we are numbering all the terms when $i=\rho(s)$.

Then we are numbering all the other elements of E_m by such a way that for any $d,b\in Q_{4^I}$, any $k_1,k_2=0,\ 1,\ \dots,\ nJ$, and for any natural numbers $\ell_1<2^{nJ-k_1}$ and $\ell_2<2^{nJ-k_2}$, the number of any term $\bar{d}^{\,k_1,\ell_1}$ is less than the number of any term from $\bar{b}^{\,k_2,\ell_2}$ if either $k_1<k_2$, or $k_1=k_2$ and $\ell_1<\ell_2$.

We fix such an enumeration.

All other acceptable enumerations are obtained as follows.

For any $k = 0, 1, \ldots, nJ$, and for any even natural number $\ell < 2^{nJ-k}$, we consider an automorphism $\tau_{k,\,\ell}$ of $(Q_{4^I}, \circ_i)$. For any $k = 0, 1, \ldots, nJ-1$, and for any odd natural number $\ell < 2^{nJ-k}$, we consider an automorphism $\tau_{k,\,\ell}$ of $(Q_{4^I}, \circ_i)$ such that for any $b, d \in Q_{4^I}$,

$$(\overline{\tau_{k,\,\ell}(b)}^{\,k,\ell}\,\overline{\tau_{k,\,\ell+1}(d)}^{\,k,\ell+1}) = \overline{\tau_{k+1,\,\ell/2}(b \circ d)}^{\,k+1,\ell/2}$$

and

$$(\overline{\tau_{k,\,\ell}(a)}^{\,k,\ell}\,\overline{\tau_{k,\,\ell+1}(a)}^{\,k,\ell+1}) = \overline{\tau_{k+1,\,\ell/2}(a)}^{\,k+1,\ell/2}. \tag{C}$$

For any set of automorphisms

$$\{\tau_{k,\,\ell} \mid k = 0, 1, \ldots, nJ;\ \ell < 2^{nJ-k}\}$$

such that (C) holds, we consider the enumeration of E_m such that for $k = 0, 1, \ldots, nJ$ and any natural number $\ell < 2^{nJ-k}$, the number of $\bar{b}^{\,k,\ell}$ in the enumeration is the number of $\overline{\tau_{k,\,\ell}(b)}^{\,k,\ell}$ in the fixed enumeration. The number of an element of $b \mid b \in Q_{4^I}\}$ in the enumeration coincides with the number of the element in the fixed enumeration.

The number of an element x of E_m in an acceptable enumeration τ is denoted by $\nu_\tau(x)$. Let $\nu_\tau(x)$ also denote the element of the number in the fixed enumeration.

For different x and y from E_m, let $\nu_\tau(f_\tau(x,y))$ be $f_m(\nu_\tau(x), \nu_\tau(y))$. Let $f_\tau(x,x) = f_m(x,x)$ for $x \in E_m$ if x belongs to the substructure generated by a in $(E_m,\ f_m)$. Let $f_\tau(x,x) = x$ for $x \in E_m$ if x does not belong to the substructure generated by a in $(E_m,\ f_m)$.

The definition guarantees that for pairwise different numbers n_1, n_2, and n_3, $f_\tau(n_1,n_2) = n_3$ iff $f_m(n_1,n_2) = n_3$. So for different numbers n_1 and n_2, the number of $f_\tau(n_1,n_2)$ does not depend on an acceptable enumeration τ. We write (xy) instead of $f_\tau(x,y)$ if it is understandable what τ is meant.

It follows from (C) that c belongs to the substructure generated by a in $(E_m,\ f_m)$ iff c belongs to the substructure generated by a in $(E_m,\ f_\tau)$.

Take $c = g(nJ)((\mathfrak{1})_a, \ldots, (\mathfrak{s})_a)$. By Theorem 3 and Example 3, $\mathrm{rc}(c) \geqslant nJ+1$. Identify c with the containing c class modulo $=$. Then $c \in E_m$.

We will consider class $\mathfrak{K}_1$. A standard structure

$$(E_m,\ f_\tau,\ a,\ c,\ ',\ 0,\ LAST). \tag{D}$$

belongs to $\mathfrak{K}_1$ iff τ is an acceptable enumeration and c belongs to the substructure generated by a in $(E_m,\ f_\tau)$.

As c belongs to the substructure generated by a in $(E_m,\ f_\tau)$, program Π_1 stops on (D). Moreover, Π_1 stops on (D) for any acceptable enumeration τ of E_m.

We take $\Pi_2 \in \Pi_1$ such that Π_2 stops on (D). The choice of $\Pi_2 \in \Pi_1$ depends on the given acceptable enumeration of E_m.

A linear program Π_2 from Π_1 is called *correct* iff there are an acceptable enumeration τ of E_m such that Π_2 stops on (D).

A standard term is a, a number, or (t_1t_2) where t_1 and t_2 are standard terms. A number is either 0, or t' where t is a standard term or a number. An a-term is either a, or (t_1t_2) where t_1 and t_2 are a-terms. If a standard term t is a, we associate $\{a\}$ to t. If a standard term t is number n_1, we associate $\{n_1\}$ to t. We associate to (t_1t_2) the union of the sets associated to t_1 and to t_2. If the set associated to term t is S and $S = \{r_1, \ldots, r_\ell\}$, we write $t(r_1, \ldots, r_\ell)$.

For any acceptable enumeration, any standard term is equal to a natural number less than $LAST + 1$. Indeed, if t_1 and t_2 are equal to numbers n_1 and n_2, then (t_1t_2) is uniquely defined and has a number in the enumeration.

Let $i \in \{1, \ldots, \ell\}$. The result of the substitution (t_1t_2) for number r_i in $t(r_1, \ldots, r_\ell)$ is (t_1t_2) if $t(r_1, \ldots, r_\ell)$ is r_i. The result of the substitution (t_1t_2) for number r_i in $t(r_1, \ldots, r_\ell)$ is (t_5t_6) if $t(r_1, \ldots, r_\ell)$ is (t_3t_4), t_5 is the result of the substitution (t_1t_2) for number r_i in t_3, and t_6 is the result of the substitution (t_1t_2) for number r_i in t_4. $t(r_1, \ldots, r_{i-1}, (t_1t_2), r_{i+1}, \ldots, r_\ell)$ denotes the result of the substitution (t_1t_2) for number r_i in $t(r_1, \ldots, r_\ell)$.

Let

$$\Pi_2 = \alpha_1; \ldots; \alpha_p$$

and $0 < i \leqslant p$. Let

$$\Pi_{2,i} = \alpha_1; \ldots; \alpha_i.$$

We can suppose that p does not exceed

$$n^2(F(nJ)4^I)^n.$$

Indeed, if for $0 < i < j \leqslant p$,

$$\mu(\alpha_1; \ldots; \alpha_i)(z_\ell) = \mu(\alpha_1; \ldots; \alpha_j)(z_\ell)$$

for any $\ell \in \{1, \ldots, n\}$, and the same operator of the same union member of $(\Pi^1 + \cdots + \Pi^q)$ is the last operator of both $\Pi_{2,i}$ and $\Pi_{2,j}$, then we can delete

$$\alpha_{i+1}; \ldots; \alpha_j$$

from Π_2.

If α_i is either a test $u = v$ or a test $v = u$, $\mu(\Pi_{2,i})(u)$ is c, $\mu(\Pi_{2,i})(v)$ is a standard term $(t_1 t_2)$ where t_1 and t_2 are equal to numbers n_1 and n_2 in the enumeration, n_1 is not the number of c, and $i \leqslant j \leqslant p$, it follows from $\Pi_{2,j}$ that c is equal to a standard term $(n_1 n_2)$. In the case, if t_1 is not a number, then it follows from $\Pi_{2,j}$ that n_1 is equal to a standard term t_1. In the case, if t_2 is not a number, then it follows from $\Pi_{2,j}$ that n_2 is equal to a standard term t_2.

If α_i is either a test $u = v$ or a test $v = u$, $\mu(\Pi_{2,i})(u)$ is ℓ where ℓ is a number, $\mu(\Pi_{2,i})(v)$ is not a number and is a standard term $(t_1 t_2)$ where t_1 and t_2 are equal to numbers n_1 and n_2 in the enumeration, n_1 is not equal to ℓ, and $i \leqslant j \leqslant p$, it follows from $\Pi_{2,j}$ that ℓ is equal to a standard term $(n_1 n_2)$. In the case, if t_1 is not a number, then it follows from $\Pi_{2,j}$ that n_1 is equal to a standard term t_1. In the case, if t_2 is not a number, then it follows from $\Pi_{2,j}$ that n_2 is equal to a standard term t_2. In the case, if t_1 is a or t_2 is a, it follows from $\Pi_{2,j}$ that ℓ is equal to a-term (aa).

Let d be either c, or a number. Let $1 \leqslant j \leqslant p$. If it follows from $\Pi_{2,j}$ that d is equal to a standard term $t(r_1, \ldots, r_k)$, and it follows from $\Pi_{2,j}$ that r_i is equal to a standard term $(t_1 t_2)$, then it follows from $\Pi_{2,j}$ that d is equal to a standard term

$$t(r_1, \ldots, r_{i-1}, (t_1 t_2), r_{i+1}, \ldots, r_k).$$

If it follows from $\Pi_{2,j}$ that c is equal to a standard term $t(r_1, \ldots, r_k)$, $1 \leqslant i \leqslant k$, and r_i is the number of an element situated on a node of a level, we say that the node of the level is attended by $\Pi_{2,j}$. In the case, if $\ell \leqslant j$ and for a variable v, the number of $\mu(\Pi_{2,\ell})(v)$ is r_i, we say that in moment ℓ the node of the level is attended by $\Pi_{2,j}$.

If $1 \leqslant \ell \leqslant \ell_1 \leqslant j$ and in moment ℓ a node of a level is attended by $\Pi_{2,j}$, we say that in moment ℓ_1 the node of the level is attended by $\Pi_{2,j}$.

We also say that the node of the level is marked by n_i in the attendance. If n_i is the number of an element of E_m, we also say that the element is the mark of the node in the attendance.

The parent of a node is a great parent of the node. The parent of a great parent of a node is a great parent of the node.

If in moment of the attendance by $\Pi_{2,j}$ of a node, all the great parents of the node were attended, the attendance of the node is called right.

If an attendance of a node is not right, the attendance of the node is called left if in moment of the attendance by $\Pi_{2,j}$ of the node, all the descendants of the node were attended. In any moment, there is no left attendance of a node of a level less than $m - n$. Otherwise, for some independent $t_1, \ldots, t_{2^n}$, $\mathrm{rc}(g(n)(t_1, \ldots, t_{2^n}))$ does not exceed n.

If it follows from $\Pi_{2,j}$ that c is equal to a standard term $t(a)$, we say that $\Pi_{2,j}$ attend each leaf of $g(nJ)$ for c and double check that the leaf is marked by a for c. In that case, we also say that $\Pi_{2,j}$ produces a complete round.

Any correct Π_2 must attend each leaf of $g(nJ)$ for c and double check that the leaf is marked by a for c. If it is not the case, it is possible to change the mark of the leaf, and to get e such that Π_2 stops on

$$(E_m,\ f_\tau,\ a,\ e,\ ',\ 0,\ LAST) \tag{E}$$

from $\mathfrak{K}_1$ where e does not belong to the substructure generated by a in $(E_m,\ f_m)$.

For any $k \in \{0,\ 1,\ \ldots,\ nJ - 1\}$ and any natural number $\ell < 2^{nJ-k}$, any correct Π_2 when it produces a complete round attends ℓth node of level k when the node is marked by $\bar{a}^{k,\ell}$.

It is impossible that α_i is either a test $u = v$ or a test $v = u$, $\mu(\Pi_{2,i})(u)$ is c, and $\mu(\Pi_{2,i})(v)$ is an a-term. Otherwise, for some independent $t_1, \ldots, t_{2^m}$, $\mathrm{rc}(g(nJ)(t_1, \ldots, t_{2^m})) < n + 1$.

For any acceptable enumeration τ, we chose $\Pi_2(\tau)$ such that $\Pi_2(\tau)$ produces the minimal number of the complete rounds.

Let $J = 2J_1$. There are $(4^I F(nJ))^n$ possible values of variables $z_1, \ldots, z_n$. There are 2^{nJ_1} possibilities for the first node of the level nJ_1 to be attended by Π_2 when it produces a complete round. There are $(4^I)^{nJ_1}$ acceptable enumerations of the elements of the form $\bar{a}^{k,\ell}$ situated on the descendants of the great parents of the first node.

Thus, for big enough I and J, there are more than

$$\frac{(4^I)^{nJ_1}}{2^{nJ_1+1}(4^I F(nJ))^n n^2}$$

pairwise different acceptable enumerations τ of E_m such that in $\Pi_2(\tau)$ for any enumeration τ from those, the same node of the level is the first attended by $\Pi_2(\tau)$ node of the level, the same nodes of the levels less than $(nJ_1 + 1)$ are attended in time of the attendance, and in time of the attendance, each variable has the same value for any enumeration.

We can also suppose that for all those enumerations, in the moment of the first attendance of a node of level nJ_1, the same operator of the same linear program from Π^1,..., Π^q is realized.

Let us call the enumarations the good enumerations.

For the first complete round, there are nJ_1 moments when the next descent is realized. Between the moments, there take place constructions of marks of descent nodes as well as constructions of numbers for checks. A check is a test of the form $x = y$ or $y = x$ where the value of x is a number, and the value of y is a-term which the register complexity is 1.

We suppose that first all the checks are produced and then the constructions of the marks of the descent nodes are produced.

Under the assumption, the proof is finished as follows.

After all the checks, there are less than $(4^I F(nJ))^n$ possible values of variables $z_1, \ldots, z_n$.

Thus for many good enumerations, the same operator of the same linear program from Π^1,..., Π^q is realized and the same values of $z_1, \ldots, z_n$ are immediately after all the checks.

Take one of those enumerations. It cannot produces the beginning of the complete round for more than $n^2(F(nJ)4^I)^n$ enumerations from those enumerations.

Among them, there are two different τ and ρ such that the part of $\Pi_2(\rho)$ from immediately after all the checks to the first attendance of a node of the level nJ_1 in the first complete round does not contain the same part of $\Pi_2(\tau)$.

It is possible to substitute the part of $\Pi_2(\rho)$ for the part of $\Pi_2(\tau)$.

It reduces the number of the complete rounds produced by $\Pi_2(\tau)$. $\square$

In [7], there is a problem as to comparison of expressive power of logics of the regular programs and the context-free deterministic programs (Conjecture Γ3 in [7]). The conjecture is: There is a context-free deterministic program which stop predicate cannot be represent by a formula of logic of the regular programs.

We proved the conjecture.

Acknowledges

I would like to thank Professor Sergei Dudakov for useful comments and corrections of errors in the first versions of the paper. I would like to thank Professor Stephen A.Cook for useful comments and suggestions.

References

[1] S. Cook. Variations on pushdown machines. In *Proc. 1st ACM Symp. on Theory of Computing*, pages 229–232, 1969.

[2] S. Cook. Characterizations of push-down machines in terms of time-bounded computers. *Journal of the ACM*, 18(1):4–18, 1971.

[3] S. Cook and R. Sethi. Storage requirements for deterministic polynomial time recognizable languages. In *Proc. 6th ACM Symp. on Theory of Computing*, pages 33–39, 1974.

[4] S. Cook and R. Sethi. Storage requirements for deterministic polynomial time recognizable languages. *Journal of Computer and System Sciences*, 13(1):25–37, 1976.

[5] I. Musikaev and M. Taitslin. On dynamic theories of free algebras. *Math. USSR Sbornik*, 66(2):313–327, 1990.

[6] I. Musikaev and M. Taitslin. Limitations of the program memory and the expressive power of dynamic logics. *Information and Computation*, 103(2):195–203, 1993.

[7] A. Stolboushkin and M. Taitslin. Dynamic logics. In V. Mel'nikov, editor, *Cybernetics and Computer Technology*, volume 2, pages 180–230. Nauka, Moscow, USSR, 1986. Russian.