S©MR

СИБИРСКИЕ ЭЛЕКТРОННЫЕ
МАТЕМАТИЧЕСКИЕ ИЗВЕСТИЯ
Siberian Electronic Mathematical Reports
http://semr.math.nsc.ru
ISSN 1813-3304

Vol. 2?, No. ?, pp. 144–145 (202?)
https://doi.org/10.33048/semi.2026.20.???

УДК 510.52 + 519.16
MSC 03D15,03B70,68Q15,68Q17

THE DEFINABILITY OF THE TURING MACHINE COMPUTATIONS BY BOOLEAN FORMULAE

I.V. LATKIN

Communicated by

Abstract: We prove the possibility of encoding long-continued computations of the deterministic two-tape Turing machines using relatively short-length quantified Boolean formulae. This simulation enables us to slightly improve the earlier-found lower complexity bound for decidable theories that are nontrivial relative to some equivalence relation (this relation may also be equality); namelly, each of these theories is consistent with the formula asserting that there are two non-equivalent elements. Furthermore, it will be proved through this modeling that the complexity classes Deterministic Exponential Time and Deterministic Polynomial Space coincide. Moreover, any language recognized in polynomial time can be recognized in almost logarithmic space.

Keywords: computational complexity, definability of computations, exponential time, polynomial space, the lower complexity bound of the language recognition.

Latkin, I.V., The definability of the Turing machine computations by Boolean formulae.

The author thanks Mikhail G. Peretyat'kin and Alexander V. Seliverstov for their attention to the work and useful discussion.

Received Desember, 20, 2025, Published December, 31, 2026.

1 Introduction

At the beginning, we recall some designations. Although the majority of them are widely applied, this reminder will help to avoid misunderstandings.

A function $\exp_k(n)$ is called *k-iterated* (or *k-story*, or *k-fold*) exponential if, for every natural k, it is calculated in the following way: $\exp_0(n) = n$, $\exp_{k+1}(n) = 2^{\exp_k(n)}$. The length of a word X is denoted by $|X|$, i.e., $|X|$ is the number of symbols in X, together with indices. If A is a set, then $|A|$ denotes its cardinality; "$A \rightleftharpoons \mathcal{A}$" means "$A$ is a designation for $\mathcal{A}$"or "A equals $\mathcal{A}$ by definition"; and $\exp(n) \rightleftharpoons \exp_1(n)$. If a is a real number, then $\lceil a \rceil$ will denote the least integer that is no less than a. We will not use logarithms on different bases, so $\log x$ further denotes $\log_2 x$.

The variable n always designates the length of the input string in the denotations of any complexity classes. We recall that r-$DTIME[t(n)]$ and r-$DSPACE[s(n)]$ are accordingly the classes of languages recognized by the r-tape deterministic Turing machines (r-DTM) with the time and space complexity $\mathcal{O}(t(n))$ and $\mathcal{O}(s(n))$. We regard that r-$DTIME[t(n)] \subseteq r$-$DTIME[t_1(n)]$ and r-$DSPACE[s(n)] \subseteq r$-$DSPACE[s_1(n)]$, if $t(n) < t_1(n)$ and $s(n) < s_1(n)$ for almost all n. When the number of tapes is inessential, we will omit the label "r-".

A set (or a language) is identified with the related decision (recognition) problem. The language $TQBF$ consists of the true quantified Boolean formulae. A Turing machine is identified with its program (or *transition function*).

The goals and objectives, the paper's structure. The computation definability by quantified Boolean formulae was well-known for a long time. For example, back in 1971, S.A. Cook modeled with such formulae the actions of nondeterministic machines in polynomial time [3]; this is probably their first application for modeling computations. Cook's method gives rhe polynomially bounded simulating ∃-formulae that model very cumbersome computations since a nondeterministic machine potentially is able to produce an exponential amount of configurations (or *instantaneous descriptions* [1, 3, 10]) in polynomial bounded time; and the description of each of these potential configurations can be deduced from Cook's formula.

Then, A.R. Meyer and L.J. Stockmeyer showed in 1973 that a language $TQBF$ is complete in the class **PSPACE** under polynomial reduction (Theorem 4.3 in [10]). This implies, in particular, that there is an algorithm, which produces a quantified Boolean formula for every Turing machine P and for any input string X in polynomial time, and all these sentences model the computations of the machine P in polynomial space; namely, each of these sentences is true if and only if P accepts the given input. The long enough computations are simulated at that, as the polynomial constraint on memory allows the machine to run during the exponential time [1, 2, 4, 6].

Next, the author, having modified an elegant construction by Stockmeyer and Meyer, proved the following

Theorem 1 ([7])**.** *For each one-tape deterministic Turing machine P and every input string X, one can construct a closed formula (sentence) $\Omega(X,P)$ of the signature of the two-element Boolean algebra $\mathcal{B}$ with the following properties:*
(i) $\Omega(X,P)$ can be written within polynomial time of $|X|$ and $|P|$;
(ii) $\Omega(X,P)$ is true over the algebra $\mathcal{B}$ if and only if the Turing machine P accepts input X within time $\exp(|X|)$;
(iii) for every $\varepsilon>0$, there is a constant $D>0$ such that the inequalities $|X|<|\Omega(X,P)|\leqslant D\cdot|P|\cdot|X|^{2+\varepsilon}$ hold for all sufficiently long X.

It follws from this that the theory $Th(\mathcal{B})$, which is equivalent to the language $TQBF$ under polynomial reduction, has a subexponential lower bound of the recognition complexity (Corollary 3.1 in [7]). Similar subexponential lower bounds on the recognition complexity are valid for all equivalent-nontrivial theories (Corollary 7.1 in [7]). This will be discussed in more detail in Section 3. Furthermore, using these estimates of the cpmputation complexity, we odtain that the class **P** is a proper subclass of **PSPACE** (Corollary 3.2 in [7]), since both the theory $Th(\mathcal{B})$ and the language $TQBF$ not only belong to the class **PSPACE** but are also polynomially complete in this class [10].

We will often cite the article [7] in Sections 1–3; nevertheless, the given paper is fully self-contained, and no previous knowledge of that work is needed. Moreover, the paper [7] is not even mentioned in Sections 4–7 during the proof of the main theorem on modeling (Theorem 2), although the method of its proof differs from that of Theorem 1 only in details, but these details are essential.

In this paper, we will generalize and strengthen Theorem 1, as well as other results from [7]; this we will consider in detail in Section 3. Namely, the time restriction (it will be named a *simulation period*) in the second item of this theorem can be extended to $\exp(m(|X|)$, where $m(n)$ is any space-constructible function.

This extension of the simulation period (see Theorem 2 below) is a fairly easy task concerning its proof, although it results in an impressive consequence that **PSPACE** $=$ **EXP** (Theorem 5). It is more interesting to learn to write the modeling formulae $\Omega^{(m)}(X,P)$ for the Turing machines having two tapes because such machines allow us to define the space complexity that is less than the length of an input string. More precisely, we will use the one-tape Turing machines equipped with a read-only input tape.

In addition, we also will reduce (not much) the upper bound of the length of these modeling sentences for $m(n)=\mathcal{O}(n)$ in comparison with ones from [7] by modifying some fragments of the simulation formulae (both these purposes will be achieved in Sections 3,5–7; see also Section 9). This will allow us to obtain a more precise lower bound of the recognition complexity for the many decidable theories, specifically for $TQBF$ and $Th\mathcal{B}$ from Theorem 1. We will discuss this in detail at the end of Subsection 3.1.

Certainly, even this refined assessment of the recognition complexity is very far from the real one for many decidable theories; numerous examples confirming this are contained in [8], and the theory described in [11] generally has fantastic complexity. Nevertheless, the estimate obtained in Corollary 4 is close to exact for theories of one equivalence relation, pure equality, and the language $TQBF$. Apparently, it is also almost exact for the theory of any finite algebra having at least two elements.

Our goal is also to prove that any problem from class **P** can be solved using the logarithmic memory. This goal will be achieved in Section 8 (Theorem 6), not only by limiting the simulation period but mainly by simulating the computations of two-tape Turing machines. To understand the proof of this theorem, one needs a detailed description of the formulae that model the actions of two-tape machines, and which cannot be obtained from the formulation of Theorem 2. For this reason, a complete proof of the main theorem on modeling is given in Sections 4–7.

We will discuss the *possibility of relativization* and *generalization to the non-deterministic case* in the last nineth section.

2 Used methods and the main idea.

The method, which is employed for the implementation of definability in the prove of Theorem 4.3 from [10], permits writing a polynomially bounded formula for the modeling of the exponential quantity of the Turing machine steps, provided that one step is described by the formula, whose length is polynomial. There, one running step of the machine is described by a Boolean ∃-formula. It corresponds to the subformula of the formula, which has been applied for the modeling of the actions of the nondeterministic Turing machine in the proof of **NP**-completeness of the problem SAT (the problem on the satisfiability of quantifier-free Boolean formulas) [3]; see also the proof of Theorem 10.3 in [1]. Therefore this ∃-formula will be termed *the Cook's method formula.*

The construct of Meyer and Stockmeyer is modified as follows in [7] and here. One running step of a machine is described by a more complicated formula that is universal in essence (see Remark 3 in Section 5). This complication is caused because Cook's method formula is very long for our goal — it is far longer than the amount of memory used. It has a subformula consisting of one propositional (or Boolean) variable $C_{i,j,t}$; for instance, see the proof of Theorem 10.3 in [1]. This variable is true if and only if the ith cell contains the symbol X_j of the tape alphabet at the instant t. But suppose that each of the first $k+1$ squares of tape contains the symbol X_0 at time t, and the remaining part of the tape is empty. This simple tape configuration is described by the formula that has a fragment $C_{0,0,t} \wedge C_{1,0,t} \wedge \ldots \wedge C_{k,0,t}$. Only this subformula is $2k+1$ in length without taking the indices into account. It is impossible to abridge this record since applying the universal quantifier to the indices is not allowed within the confines of the first-order theory. Thus, to describe the machine actions on the exponential fragment of tape, Cook's method formula has to be lengthened still more.

In [7], the binary notation of the cell number is encoded by a value set of special variables $x_{t,0}, \ldots, x_{t,m}$, where $m = \lceil \log T \rceil$, and T is a simulation period, while t denotes a step number. A similar notation will also be used in this paper when identifying naturally the value of the variable "true"here with the number one in [7] and the value "false"with zero. However, the length of the set of variables encoding cell numbers on different tapes will be distinct. This length will be equal to $s = \lceil \log n \rceil$ for the configurations on the first (input) tape and $m(n) = \lceil \log T \rceil$ for the ones on the second (work) tape, where n is the length of an input string (see Subsections 5.1 and 7.1 and also Section 8 for further details), provided that the function $m(n)$ is space-constructible. We will also apply various types of variables for the description of configurations on different tapes: the capital letters with subscripts will be used for the first tape, and the lowercase ones for the second. So, $\mathcal{O}(s\lceil \log_s \rceil)$ various symbols (together with indices) are sufficient for describing one cell on the first tape, and $\mathcal{O}(n \cdot s \cdot \lceil \log s \rceil)$ ones are doing for the representation of the whole input.

Certainly, our description of one tape cell is appreciably longer in comparison with one of Cook's method. Nevertheless, we will obtain a more brief description of many other fragments of the modeling formula by sacrificing the brevity of the description of one cell.

For example, one can describe an instruction action of the machine, which uses $T = \exp(m(n))$ memory cells, with the aid of a formula that is $\mathcal{O}(m(n) \cdot \lceil \log m(n) \rceil)$ in length. The main idea of such a brief description consists of the following. Firstly, a record can change in the only square of the second tape for each running step, while all the entries remain unchanged on the first (input) tape. Both the heads can only shift by one position, although the whole computation can use a significant amount of memory. Secondly, the actions executed by the Turing machine on a current step exclusively depend on the records in two cells, at which the machine heads are aimed. Therefore, it is enough to describe "the inspection"of two squares, the shifts of heads, and the possible change in the only cell on the second tape in explicit form. In contrast, the contents of the remaining ones can be "copied"by applying the universal quantifier (see the construction of the formula $\Delta^{cop}(\widehat{u})$ in Subsection 5.2). So, we need to explicitly describe only three simple actions and not more than five squares on the tapes for every running step (in fact, we explicitly will describe only two or three cells — see Subsections 4.1 and 5.2) and the proof of Proposition 2.

The denoted locality of the actions of deterministic machines has long been used in the modeling of machine computation with the help of formulae, e.g., Lemma 2.14 in [9] or Lemma 7 in [11].

The description of the initial configuration has a length of $\mathcal{O}(n \cdot s \cdot \lceil \log_2 s \rceil + m(n) \cdot \lceil \log m(n) \rceil)$ if the quantifiers are anew used. The condition for the successful termination of computations is very short. Hence, the entire formula, which simulates the first $\exp(m(n))$ steps of the computations of the machine P, will be no more than

$$\mathcal{O}(n \cdot s \cdot \lceil \log s)\rceil + m(n) \cdot \lceil \log m(n)\rceil) + \mathcal{O}(|P| \cdot m(n) + m^2(n)) \cdot \lceil \log m(n)\rceil])$$

in length, together with the indices.

Remark 1. *Since s is equal to $\lceil \log n\rceil$, we need to select $\lceil \log n\rceil \cdot \lceil \log\lceil \log n\rceil\rceil$ squares for the record of the variables $X_0, \dots, X_s$ for describing one cell at the first tape. But the function $\lceil \log\lceil \log n\rceil\rceil$ is not fully space-constructible since it grows as $o(\log n)$. However, we do not need to measure out $\lceil \log\lceil \log n\rceil\rceil$ cells using the exact same amount of memory; it is enough only to fill s cells with symbols X (this we can do even utilizing exactly s cells because the function $\lceil \log n\rceil$ is fully space-constructible); after that, we can number these cells using a little more ones than $\lceil \log n\rceil \cdot \lceil \log\lceil \log n\rceil\rceil$. It is described in Subsection 7.3 (Lemma 6) how one can do this in a more general case.*

In the beginning, all variables will be introduced in great redundancy in order to facilitate the proof; namely, the variables will have the first indices t from 0 to T. Next, many of the variables will be eliminated using the methods from [10] and [7], see Remark 4 in Subsections 5.3.2 and 7.2 for further details.

3 The main results

Here, we will state the main theorem (Theorem 2, on simulation) and prove some of its consequences, which are the most important in the author's opinion. However, the application of this theorem to languages of class **P** will be discussed separately in Section 8. The proof of the theorem on modeling is given in sections 4–7.

It is assumed that the first tape of all 2-DTMs mentioned in this paper is the input tape. This means that the machine can read the records on the first tape, but it is forbidden to erase and write any symbols on this tape.

We recall that a function $f(n)$ is termed *fully space-constructible*, if there is a 2-DTM, which puts a marker exactly in the $f(n)$th square on the second (working) tape, and the machine head on this tape does not visit $(f(n)+1)$th cell for every input having length n [4]. Such functions are sometimes named *space-constructible* [1].

Theorem 2 (on modeling)**.** *Let $m(n) \geqslant \lceil \log n\rceil$ be a fully space-constructible function increasing almost everywhere, P be a two-tape deterministic Turing machine (2-DTM), and X be an input string on the first tape of machine P. Then one can construct a closed Boolean formula (sentence) $\Omega^{(m)}(X, P)$ with the following properties:*

(i) $\Omega^{(m)}(X, P)$ is true if and only if the 2-DTM P accepts input X within time $\exp(m(|X|))$;

(ii) there exist constants D_1, D_2, and D_w such that the inequalities ($n = |X|$ here and everywhere below)

$$m(n) \ < \ |\Omega^{(m)}(X,P)| \ \leqslant \ D_1 \cdot n \cdot \lceil \log n \rceil \cdot \lceil \log \lceil \log n \rceil \rceil + \\ + D_2 \cdot m(n) \cdot \lceil \log m(n) \rceil + D_w \cdot \lceil (|P| \cdot m(n) + m^2(n)) \cdot \lceil \log m(n) \rceil \rceil \qquad (1)$$

hold for every sufficiently long X;

(iii) there is a polynomial $g(k)$ such that for all X and P, the construction time of the sentence $\Omega^{(m)}(X,P)$ is not greater than $g(|X|+|P|+m(|X|))$, while required space for this is $\mathcal{O}(|\Omega^{(m)}(X,P)|)$.

See the proof in Sections 4–7. At first, the simulation formulae will have a huge number of "redundant"variables. The constructed formulae will be rewritten in the brief form after the correctness of this modeling has been ascertained (see Propositions 2(ii), 3, and 4(ii) below); the Stockmeyer and Meyer method is substantially used at that.

Remark 2. *a) The function $g(|X|+|P|+m(|X|))$ may depend on $|X|$ in a non-polynomial way, despite $g(k)$ being polynomial, because the function $m(n)$ may not be polynomial; e.g., $m(n)$ can be k-fold exponential.*

b) The upper estimate of the length of the modeling formula has three summands in the right part of Inequality (1). The first of them, $D_1 \cdot n \cdot \lceil \log n \rceil \cdot \lceil \log \lceil \log n \rceil \rceil$, evaluates the length of the formula that describes an input string on the first tape. The next $D_2 \cdot m(n) \cdot \log m(n)$ does it for the formula, which asserts that the second tape is empty before the beginning of the machine work. The third summand does it for the subformula that describes the machine's actions P. The summand corresponding to the condition for the successful termination of computations is absent here, as this subformula is very short; its length is not more than $c \cdot m(n)$ for an appropriate constant c.

Corollary 1 (of the main theorem)**.** *Under the conditions of the theorem, if $n \leqslant m(n) \leqslant Cn$ for some constant C, then for every $\varepsilon > 0$, there is a constant $D_C > 0$ such that the inequality $|\Omega^{(m)}(X,P)| \leqslant D_C \cdot |X|^{2+\varepsilon}$ is valid for all long enough X; more precisely, $|\Omega^{(m)}(X,P)| \leqslant D_C \cdot |X|^2 \cdot \lceil \log |X| \rceil$.*

Proof. The latter inequality follows from Inequality (1) and the condition $m(n) \leqslant Cn$; it is valid for all X such that $|X| \geqslant |P|$. For each $\varepsilon > 0$, there is a number k such that an inequality $(\lceil \log n \rceil \leqslant n^{\varepsilon}$ holds for $n \geqslant k$. The first inequality is obtained from this and the second one. □

3.1. The computational complexity of decidable theories. The following consequences of the main theorem require some concepts and material from [7]; we will briefly remind the reader of this information. The function $F(n)$, which is monotone increasing on all sufficiently large n, is called a *limit upper bound for the class of all polynomials* (LUBP) if, for any polynomial p, there is a number k such that the inequality $F(n) > p(n)$ holds for $n \geqslant k$, i.e., each polynomial is asymptotically smaller than F. An obvious example of the limit upper bound for all polynomials is a s-iterated exponential for every $s \geqslant 1$.

The following statement is a certain generalization of the corresponding assertion from Subsection 4.1 in [8]; it is formulated here to estimate the computational complexity of decidable theories. However, it is valid for arbitrary languages — see Proposition 2.1 in [7].

Proposition 1 ([7], the generalized Rabin and Fischer method)**.** *Let F be a limit upper bound for all polynomials and $\mathcal{T}$ be a theory in the signature (or underlying language) σ. Suppose that for any k-tape DTM P and every string X on the input tape of this machine, one can effectively construct a sentence $S(P,X)$ of the signature σ, possessing the following properties:*

(i) $S(P,X)$ can be built within time $g(|X|+|P|)$, where g is a polynomial fixed for all X and P;

(ii) $S(P,X)$ belongs to $\mathcal{T}$ if and only if the machine P accepts the input X within $F(|X|)$ steps;

(iii) there exist constants $D, b, s > 0$ such that the inequalities $|X| \leqslant |S(P,X)| \leqslant D \cdot |P|^b \cdot |X|^s$ hold for all sufficiently long X, and these constants do not depend on P.

Then $\mathcal{T} \notin k$-DTIME$[F(D^{-\zeta} \cdot n^{\zeta})]$ holds for $\zeta = s^{-1}$.

The following statement improves the lower bound on the recognition complexity of the theory of the most straightforward Boolean algebra and the language $TQBF$.

Corollary 2. *Let $\mathcal{B}$ be a two-element Boolean algebra. For every $\varepsilon > 0$, neither $TQBF$ nor $Th\mathcal{B}$ belong to 2-DTIME$[\exp(D \cdot n^{\rho})]$ for some constant $D < 1$, where $\rho = (2+\varepsilon)^{-1}$ and $n^{\varepsilon} \geqslant \log n$.*

Proof. It straightforwardly follows from the proposition and Corollary 1, because the language $TQBF$ and theory $Th\mathcal{B}$ are not only polynomially equivalent, but also each sentence of $TQBF$ can be rewritten as an equivalent sentence of $Th\mathcal{B}$ with linear lengthening, and vice versa. □

Corollary 3 ([7])**.** *The class* **P** *is a proper subclass of the class* **PSPACE**.

Proof. The theory $Th\mathcal{B}$ and the language $TQBF$ do not belong to the class **P** under the previous corollary. But both belong to the class **PSPACE**; moreover, they are polynomially complete for this class [6, 10]. □

Let $\mathfrak{K}$ be a class of algebraic systems, whose signature σ contains the symbol of the binary predicate $\sim$, and this predicate is interpreted as an equivalence relation on every structure of the class; in particular, $\sim$ may be an equality relation. These relations are denoted by the same symbol $\sim$ in all systems of the class.

Definition 1 ([7])**.** *Let us assume that there exists a* $\sim$-nontrivial *system $\mathcal{E}$ in a class $\mathfrak{K}$, i.e., this structure contains at least two $\sim$-nonequivalent elements. Then the class $\mathfrak{K}$ is also termed* $\sim$-nontrivial. *A theory $\mathcal{T}$ is named*

$\sim$-nontrivial *if it has a $\sim$-nontrivial model. When the relation $\sim$ is either equality or there is a formula $N(x,y)$ of the signature σ such that the sentence $\exists x, y N(x,y)$ is consistent with the theory $Th\mathfrak{K}$ (or $\mathcal{T}$, or belongs to $Th\mathcal{E}$), and this formula means that the elements x and y are not equal, then the term "$\sim$-nontrivial"will be replaced with* "equational-nontrivial". *If it does not matter which equivalence relation is meant, we will discuss* equivalency-nontrivial *theory, class, or structure.*

Prominent examples of such theories are the theories of pure equality and one equivalence relation. Furthermore, nearly all decidable theories mentioned in the surveys [5, 8] are nontrivial regarding equality or equivalence.

Theorem 3. *Let $\mathcal{E}$, $\mathfrak{K}$, and $\mathcal{T}$ accordingly be an equivalency-nontrivial system, class, and theory of the signature σ; in particular, they may be equational-nontrivial. Then there exists an algorithm that builds the sentence $\Omega^{(T)}(X,P)$ of σ for every 2-DTM P and any of its input X, where $T \in \{Th\mathcal{E}, Th\mathfrak{K}, \mathcal{T}\}$; this formula possesses the properties (i) and (ii) of the sentence $S(P,X)$ from the formulation of Proposition 1 for $F(|X|) = \exp(|X|)$. Moreover, for each $\varepsilon > 0$, there is a constant $E_{T,\sigma}$ such that the inequalities $|X| < |\Omega^{(T)}(X,P)| \leqslant E_{T,\sigma} \cdot |P| \cdot |X|^{2+\varepsilon}$ hold for any long enough X and such that $|X|^{\varepsilon} \geqslant \log |X|$.*

Proof. At first, given X and P, one can write a simulating Boolean sentence $\Omega^{(|X|)}(X,P)$ applying Theorem 2. Then, this formula is transformed into the required sentence $\Omega^{(T)}(X,P)$ within polynomial time as this does in the proof of Theorem 7.1 in [7]; there, it is shown that the length of $\Omega^{(T)}(X,P)$ increases linearly in so doing. □

Corollary 4. *The recognition complexity of each equivalency-nontrivial decidable theory $\mathcal{T}$, in particular, equational-nontrivial, has the non-polynomial lower bound; more precisely, $\mathcal{T} \notin$ 2-DTIME$[\exp(D_{T,\sigma} \cdot n^{\delta})]$, where $\delta = (2+\varepsilon)^{-1}$, $D_{T,\sigma} = (E_{T,\sigma})^{-\delta}$, and $\varepsilon > 0$ is preassigned.*

Proof. It straightforwardly follows from this theorem and Proposition 1. □

The lower bounds for the computational complexity of the theories in Corollaries 2 and 4 are somewhat higher than those obtained in [7], although this is difficult to see from their formulations. In Lemma 3.1 from [7], the factor $n^{2+\varepsilon}$ arises as an upper bound for the quantity $(n \cdot \log n)^2 = n^2 \cdot \log^2 n$ for the estimation of the length of the modeling formula. In contrast, this factor evaluates $n \cdot (n \log n)$ in Corollary 1 and Lemma 7 when $m(n) = n$. Moreover, these evaluations of the computational complexity are now valid for two-tape machines; therefore, they are even higher for one-tape machines.

3.2. Complexity classes and polynomially complete problems. In the previous subsection and [7], it has been proved that the class **P** is a proper subclass of **PSPACE**; this is based on the non-polynomial lower bound on the computational complexity of all the equational-nontrivial theories. Another immediate consequence of Theorem 2 is the following.

Theorem 4. *The language $TQBF$ and the theory $Th\mathcal{B}$ are polynomially complete for the class* **EXP***.*

Proof. Theorem 2 explicitly ensures a polynomial reduction of every problem from the class **EXP** to appropriate Boolean formulae when $m(n)$ is polynomial. □

Corollary 5. *The class* **PSPACE** *equals the class* **EXP***.*

Proof. The language $TQBF$ is polynomially complete for both classes according to the theorem and [10]. □

Let us recall some concepts to formulate the following result. The letter c denotes some constant in the following definitions:

$$^{\mathbf{k}}\mathbf{EXP} \rightleftharpoons \bigcup_{c>0} \bigcup_{j>0} DTIME[\exp_k(cn^j)]$$

is k-fold Deterministic Exponential Time for $k \geqslant 1$ and $\mathbf{EXP} \rightleftharpoons {}^{\mathbf{1}}\mathbf{EXP}$;

$$^{\mathbf{k}}\mathbf{EXPSPACE} \rightleftharpoons \bigcup_{c>0} \bigcup_{j>0} DSPACE[\exp_k(cn^j)]$$

is k-fold Deterministic Exponential Space for $k \geqslant 1$;

$$\mathbf{Log} \rightleftharpoons \bigcup_{j>0} DSPACE[\log^j n]$$

is Polylogarithmic Space (or sometimes Polylog Space).

One can define also that $^{\mathbf{0}}\mathbf{EXP}$ is **P**, $^{\mathbf{0}}\mathbf{EXPSPACE} = \mathbf{PSPACE}$, and $^{-\mathbf{1}}\mathbf{EXPSPACE} = \mathbf{Log}$.

Theorem 5. *For every $k \geqslant 0$, the classes k-fold Deterministic Exponential Space and $(k+1)$-fold Deterministic Exponential Time coincide, i.e.,*

$$^{\mathbf{k}}\mathbf{EXPSPACE} = {}^{(\mathbf{k+1})}\mathbf{EXP}.$$

Proof. It is known that if a deterministic 2-DTM M works within space bounded by the function $s(n)$, then it cannot have more than $s(n)\cdot\exp(Cs(n))$ various configurations on the working tape for suitable constant C [1, 2, 4, 6], so its running time is less than $s(n) \cdot \exp(Cs(n))$, provided that M halts on a given input. Hence, the inclusion $^{\mathbf{k}}\mathbf{EXPSPACE} \subseteq {}^{(\mathbf{k+1})}\mathbf{EXP}$ holds for every $k \geqslant 0$.

Let $\mathcal{L}$ be a language in $^{(\mathbf{k+1})}\mathbf{EXP}$, and P_0 be a 2-DTM that recognizes this language in time $\exp m(n)$, where $m(n) = \exp_k(cn^d)$ for some constants $c, d > 0$. According to Theorem 2, one can write down the Boolean sentence $\Omega^{(m)}(X, P_0)$ such that this formula simulates the actions of the machine P_0 on each input X; at that, Inequalities (1) are valid for some constants D_1, D_2, and D_w, and this formula can be built within space bounded a linear function on its length. So, the ascertainment of the $\Omega^{(m)}(X, P_0)$ truth replaces the computations of the machine P_0 on the input X; this substitution is realized within space that has the order of $|\Omega^{(m)}(X, P_0)|$, i.e., within space

$C(\exp_k(cn^d))^{2+\varepsilon}$ for the proper constant C and $n^\varepsilon \geqslant \log n$. The question of whether the sentence $\Omega^{(m)}(X, P_0)$ is true is decidable within a space of the same order. $\square$

4 Necessary preparations for the proof of the main theorem

In this section, we specify the restrictions on the Turing machines used, the characteristics of their actions, and the methods of recording their instructions and Boolean formulae. These agreements are essential in proving the main theorem on simulation (Theorem 2).

4.1. On the Turing machines. It is implied that simulated deterministic machines have two tapes and a fixed arbitrary finite tape alphabet $\mathcal{A}$, which contains at least four symbols: the first of them is a designating "blank"symbol, denoted Λ; the second is a designating "start"symbol, denoted $\rhd$; and the last two are the numerals 0,1 (almost as in Section 1.2 of [2]). The machine cannot erase or write down the $\rhd$ symbol. The machine tapes are infinite only to the right, as in almost all works cited here, because such machines can simulate the computations, which is T steps in length on the machines with two-sided tapes, in linear time of T [2].

Before the beginning of the machine work, the first tape contains the start symbol $\rhd$ at the most left square (at the zeroth cell), a finite non-blank input string X alongside $\rhd$, and the blank symbol Λ on the rest of its cells, i.e., these squares are empty. The cells of the second tape do not contain any symbols except the only $\rhd$ located at the zeroth cell. Both heads are aimed at the left ends of the tapes, and the machine is in the special starting state $q_{start} = q_0$. When the machine recognizes an input, it enters the accepting state $q_1 = q_{acc}$ or the rejecting state $q_2 = q_{rej}$.

The first tape serves only to store the input string; the machine cannot erase or write anything in the cells of this tape. Therefore, the simulated machines have the *three-operand* instructions of the kind: $q_i\alpha_1, \alpha_2 \to q_j\beta_1, \gamma, \beta_2$, where $\alpha_1, \alpha_2, \gamma \in \mathcal{A}$, $\beta_1, \beta_2 \in \{L, R, S\}$ (naturally, it is assumed that $\mathcal{A} \cap \{L, R, S\} = \varnothing$); α_1 and α_2 are the symbols visible by the heads, respectively, on the first and second tapes; β_1 and β_2 are the commands to shift the corresponding head to the left, or the right, or stay in the same place; γ is a symbol, which should be written down on the second tape instead of α_2; q_i and q_j are the inner states of the machine accordingly before and after the execution of this instruction.

The Turing machines do not fall into a situation where the machine stops, but its answer remains undefined. Namely, they do not try to go beyond the left edges of the tapes, i.e., there are no instructions of a form $q_i\rhd, \alpha_2 \to q_kL, \gamma, \beta_2$ or $q_i\alpha_1, \rhd \to q_k\beta_1, \rhd, L$. The programs also do not contain the *hanging* (or *pending*) internal states q_j, for which $j \neq 0, 1, 2$, and there exist instructions of a kind $\ldots \to q_j\beta_1, \gamma, \beta_2$, but no instructions are beginning with $q_j\alpha_1, \alpha_2 \to \ldots$ at least for one pair of symbols $\langle\alpha_1, \alpha_2\rangle \in \mathcal{A}^2$. The hanging states are eliminated by adding the instructions of a kind $q_j\alpha_1, \alpha_2 \to q_jS, \alpha_2, S$ for each of the missing pair $\langle\alpha_1, \alpha_2\rangle$ of alphabet symbols.

4.2. On the recording of Boolean formulae. We reserve the following *natural language alphabet* $\mathcal{A}_1$ for writing the Boolean formulae (it can be different from the alphabet of simulated machines $\mathcal{A}$): a) the capital and lowercase Latin letters for the indication of the types of propositional (Boolean) variables; b) Arabic numerals and commas for the writing of indices; c) basic Logical connectives $\neg, \wedge, \vee, \to$; d) the signs of quantifiers $\forall, \exists$; e) auxiliary symbols: (,).

If necessary (for example, if the alphabet $\mathcal{A}$ does not contain $\mathcal{A}_1$), our formulae can be rewritten in the alphabet $\mathcal{A}$ (or even $\mathcal{A}_0 \rightleftharpoons \{0, 1\}$) in polynomial time and with a linear increase in length. We always suppose this when we say that some machine writes a formula.

We also introduce the following abbreviations and agreements for the improvement of perception. The transformation of the formula written by applying these abbreviations to the one in natural language increases its length linearly.

Firstly, (1) in long formulae, the square brackets and (curly) braces are equally applied with the ordinary parentheses; (2) the connective $\wedge$ is sometimes written as '$\cdot$' (as a rule, between the variables or their negations) or as '&' (between the long enough fragments of formulae); (3) if x is propositional variable, then $\overline{x} \rightleftharpoons \neg x$.

Secondly, we will be useful in the additional (auxiliary) logical connectives, "constants", and "relations":
(4) $x \equiv y \ \rightleftharpoons\ (x \to y) \wedge (y \to x)$;
(5) $0 \ \rightleftharpoons\ x \wedge \neg x = x \cdot \overline{x}$, $\qquad 1 \ \rightleftharpoons\ x \vee \neg x = x \vee \overline{x}$ [1];
(6) $x \oplus y \ \rightleftharpoons\ \overline{x} \cdot y \vee x \cdot \overline{y}$; obviously, that $0 \oplus 0 \equiv 0$, $0 \oplus 1 \equiv 1$, $1 \oplus 0 \equiv 1$, and $1 \oplus 1 \equiv 0$, i.e., this is the modulo 2 addition;
(7) $x{<}y \ \rightleftharpoons\ x{\equiv}0 \wedge y{\equiv}1$.

Thirdly, the priority of connectives or its absence is inessential, as a difference in the length of formulae is linear in these cases; therefore, the following priority of the connectives and their versions is assumed (in descending order of closeness of the connection): $\neg$ (in both forms), $\cdot$, $\oplus$, $\equiv$, $\wedge$, &, $\to$, $\vee$.

With some liberties in notation, we use the signs 0,1, and $<$ for both numbers and formulae. However, this should not lead to confusion since what is meant will always be clear from the context.

The record $\widehat{x}$ signifies an ordered set $\langle x_0, \dots, x_s\rangle$, whose length is fixed. Naturally, the "formula" $\widehat{x} \equiv \widehat{\alpha}$ denotes the system of equivalences $x_0 \equiv \alpha_0 \wedge \ldots \wedge x_s \equiv \alpha_s$. The record $\widehat{\alpha} < \widehat{\beta} \rightleftharpoons \langle\alpha_0, \dots, \alpha_s\rangle < \langle\beta_0, \dots, \beta_s\rangle$ denotes the comparison of tuples in lexicographic ordering, i.e., it is the following

[1]The usage of the signs $\equiv$, 0, and 1 seems more natural than $\sim$ (or $\leftrightarrow$), $\perp$, and $\top$ instead of them while simultaneously applying the signs of inequality and addition modulo two.

formula:

$$\alpha_0 < \beta_0 \vee \Big\{\alpha_0 \equiv \beta_0 \wedge \Big[\alpha_1 < \beta_1 \vee \Big(\alpha_1 \equiv \beta_1 \wedge \big\{\alpha_2 < \beta_2 \vee \\ \vee \big[\alpha_2 \equiv \beta_2 \wedge \big(\alpha_3 < \beta_3 \vee \{\alpha_3 \equiv \beta_3 \wedge \ldots\}\big)\big]\big\}\Big)\Big]\Big\}.$$

The tuples of variables with two subscripts will occur only in the form where the first index is fixed, for instance, $\langle u_{k,0}, \ldots, u_{k,s}\rangle$, and we will denote it by $\widehat{u}_k$.

A binary representation of a natural number t is denoted by $(t)_2$. On the other hand, if $\widehat{\gamma}$ is a tuple consisting of 0 and 1, then $(\widehat{\gamma})$ will denote the binary representation of some natural number t. So, $((t)_2) = t$ and $((\widehat{\gamma}))_2 = (\widehat{\gamma})$ according to our dessignations. Here, 0 and 1 can denote the above-defined formulae or sometimes the numerals.

It is known that if $t = (\widehat{\gamma}) \rightleftharpoons \langle \gamma_0 \ldots \gamma_s \rangle_2$ is a binary representation of a natural number t, then the numbers $t \pm 1$ for $0 \leqslant t-1 < t+1 \leqslant \exp(2, s)$ will be expressed as

$$((\widehat{\gamma}) \pm 1)_2 = (\widehat{\gamma} \oplus \widehat{v}) \ \rightleftharpoons\ \langle \gamma_0 \oplus v_0, \ldots, \gamma_{s-2} \oplus v_{s-2}, \gamma_{s-1} \oplus v_{s-1},\ \gamma_s \oplus v_s \rangle_2,$$

where the corrections v_i are calculated in descending order of indices according to the rules defined by the following formulae:

$$\kappa_R(\widehat{\gamma}, \widehat{v}) \ \rightleftharpoons\ (v_s \equiv 1) \ \wedge\ (v_{s-1} \equiv \gamma_s) \ \wedge\ (v_{s-2} \equiv \gamma_{s-1} \cdot v_{s-1}) \ \wedge \ldots \wedge \\ \wedge\ (v_1 \equiv \gamma_2 \cdot v_2) \ \wedge\ (v_0 \equiv \gamma_1 \cdot v_1) \qquad (2)$$

for writing $t+1$; and

$$\kappa_L(\widehat{\gamma}, \widehat{v}) \ \rightleftharpoons\ (v_s \equiv 1) \ \wedge\ (v_{s-1} \equiv \overline{\gamma}_s) \ \wedge\ (v_{s-2} \equiv \overline{\gamma}_{s-1} \cdot v_{s-1}) \ \wedge \ldots \wedge \\ \wedge\ (v_1 \equiv \overline{\gamma}_2 \cdot v_2) \ \wedge\ (v_0 \equiv \overline{\gamma}_1 \cdot v_1) \qquad (3)$$

for writing $t-1$.

Lemma 1. *(i)* $|\widehat{\alpha} < \widehat{\beta}| = \mathcal{O}(|\widehat{\alpha}| + |\widehat{\beta}|)$.

(ii) The binary representations of the numbers $t \pm 1$ can be written in linear time on $|(t)_2|$; hence, their lengths (together with the formula $\kappa_R(\widehat{\gamma}, \widehat{v})$ or $\kappa_L(\widehat{\gamma}, \widehat{v})$) are $\mathcal{O}(|(t)_2|)$.

Proof. It is obtained by direct calculation. □

5 The beginning of the construction of the modeling formulae

We recall that $\mathcal{A}$ is the alphabet of simulated machines. Before writing a formula $\Omega^{(m)}(X, P)$, we add $2|\mathcal{A}|^2$ *instructions of the idle run* to a program P; they have the form $q_k\alpha_1, \alpha_2 \to q_k S, \alpha_2, S$, where $k \in \{1(\mathit{accept}), 2(\mathit{reject})\}$, $\alpha_1, \alpha_2 \in \mathcal{A}$. While the machine executes them, the tape configurations do not change.

5.1. The primary (basic) and auxiliary variables. To simulate the operations of a Turing machine P on an input X within the first $T = \exp(m(|X|))$ steps, it is enough to describe its actions on a zone, of the second tape, which is $T+1$ squares in width, as if P starts its run in the zeroth cell, it can finish a computation at most in the Tth one. Since the record of the number $(T)_2$ has $m+1 = m(|X|)+1$ bits, the cell numbers of the second tape are encoded by the values of the ordered sets of the variables of a kind "x": $\widehat{x}_t = \langle x_{t,0}, \dots, x_{t,m}\rangle$, which have a length of $m+1$. The first index t, i.e., *the color* of the record, denotes the step number, after which a configuration under study appeared on the second tape. So the formula $\widehat{x}_t \equiv \widehat{\alpha} \rightleftharpoons x_{t,0} \equiv \alpha_0 \wedge \dots \wedge x_{t,m} \equiv \alpha_m$ assigns the number $(\widehat{\alpha})$ of the needed cell of this tape in the binary notation at the instant t.

The formula $\widehat{X} \equiv \widehat{\gamma} \rightleftharpoons X_0 \equiv \gamma_0 \wedge \dots \wedge X_s \equiv \gamma_s$ assigns the number $(\widehat{\gamma})$ of the cell of the first tape in the binary notation, where $s = \lceil \log |X| \rceil$. It is clear that $\exp(s+1) > |X|+1$, which is normal since the first head can go beyond the right edge of the input string in the computation process; on the other hand, we propose that this head does not visit the $(|X|+2)$th cell. Because nothing can be erased or written in the cells of the first tape, one does not need to indicate the step number here.

Let r be so great a number that one can simultaneously write down all the inner state numbers of P and encode all the symbols of the alphabet $\mathcal{A}$ through the bit combinations in the length $r+1$. Thus, $\exp(r+1) \geqslant |\mathcal{A}|+U$, where U is the maximal number of internal states of P; and if $\beta \in \mathcal{A}$, then $c\beta \rightleftharpoons \langle c\beta_0, \dots, c\beta_r\rangle$ will be the $(r+1)$-tuple, which consists of 0 and 1 and encodes β.

The formula $\widehat{f_t} \equiv c\varepsilon$ represents a record of the symbol $\varepsilon \in \mathcal{A}$ in some cell of the second tape after step t, where $\widehat{f_t}$ is the $(r+1)$-tuple of variables. When the $(\widehat{\mu})$th cell of the second tape contains the ε symbol after step t, then this fact is associated with the *quasi-equation*[2] (or *the clause*) of color t:

$$\psi_{2,t}(\widehat{\mu} \to \varepsilon) \rightleftharpoons [\widehat{x}_t \equiv \widehat{\mu} \to \widehat{f_t} \equiv c\varepsilon] \rightleftharpoons$$
$$\rightleftharpoons (x_{t,0} \equiv \mu_0 \wedge \dots \wedge x_{t,m} \equiv \mu_m) \to (f_{t,0} \equiv c\varepsilon_0 \wedge \dots \wedge f_{t,r} \equiv c\varepsilon_r).$$

The clause

$$\psi_1(\widehat{\zeta} \to \varepsilon) \rightleftharpoons [\widehat{F} \equiv c\varepsilon] \rightleftharpoons$$
$$\rightleftharpoons (X_0 \equiv \zeta_0 \wedge \dots \wedge X_s \equiv \zeta_s) \to (F_0 \equiv c\varepsilon_0 \wedge \dots \wedge F_r \equiv c\varepsilon_r)$$

corresponds to the entry of the ε symbol at (ζ)th cell on the first tape.

The tuples of variables $\widehat{q}_t$, $\widehat{D}_t$, and $\widehat{d}_t$ are accordingly used to indicate the number of the machine's internal state and the codes of the symbols visible by the heads on the first and second tapes at the instant t; the tuples of variables $\widehat{Z}_t$ and $\widehat{z}_t$ store the scanned cell's numbers on the first and second tapes. For every step t, a number $i = (\widehat{\delta})$ of the machine state q_i and the

[2]by analogy with quasi-identities

scanned square's numbers $(\widehat{\zeta})$ and $(\widehat{\xi})$ together with the symbols α and β, which are contained there, are represented by a united *π-formula* of color t:

$$\pi_t((i)_2, \alpha, \beta, \widehat{\zeta}, \widehat{\xi}) \rightleftharpoons [\widehat{q_t} \equiv \widehat{\delta} \wedge \widehat{D}_t \equiv c\alpha \wedge \widehat{d_t} \equiv \widehat{\beta} \wedge \widehat{Z}_t \equiv \widehat{\zeta} \wedge \widehat{z_t} \equiv \widehat{\xi}] \rightleftharpoons$$
$$\rightleftharpoons (q_{t,0} \equiv \delta_0 \wedge \ldots \wedge q_{t,r} \equiv \delta_r) \wedge (D_{t,0} \equiv c\alpha_0 \wedge \ldots \wedge D_{t,r} \equiv c\alpha_r) \wedge$$
$$\wedge (d_{t,0} \equiv c\beta_0 \wedge \ldots \wedge d_{t,r} \equiv c\beta_r) \wedge (Z_{t,0} \equiv \zeta_0 \wedge \ldots \wedge Z_{t,s} \equiv \zeta_s) \wedge (z_{t,0} \equiv \xi_0 \wedge \ldots \wedge z_{t,m} \equiv \xi_m).$$

This formula expresses a condition for the applicability of instruction $q_i\alpha, \beta \to \ldots$; in other words, this is a *timer* (or *trigger*) that "activates"exactly this instruction, provided that the heads scan the $(\widehat{\zeta})$th and $(\widehat{\xi})$th cells.

The tuples of basic variables $\widehat{x_t}$, $\widehat{z_t}$, $\widehat{X}$, $\widehat{Z}_t$, $\widehat{q_t}$, $\widehat{F}$, $\widehat{f_t}$, $\widehat{D}_t$, and $\widehat{d_t}$ are introduced in great abundance to facilitate the proof. However, a final modeling formula will contain only those for which $t = 0$ or $t = T \rightleftharpoons \exp(m(n))$.

The sets of basic variables have different lengths. However, this will not lead to confusion since the tuples of the first two types (x and z) will always be $m+1$ in length without indices, the second two types (X and Z) will always be $s+1$ in length, whereas the last five ones (q, F, f, D, and d) will have a length of $r+1$. The sets of "constants"or other variables may also be different in length. But such tuples will always be unambiguously associated with some of the primary ones by using the connective $\equiv$.

The other variables are auxiliary. They will be described as needed. Their task consists of determining the values of the basic variables of the color $t+1$, provided that the primary ones of the color t have the "correct"values; this "attribution"should adequately correspond to that instruction employed at step $t+1$.

Lemma 2. *If the indices are left out of the account, then* $|\psi_{2,t}(\widehat{u} \to \beta)| = \mathcal{O}(m+r)$, $|\psi_1(\widehat{V} \to \alpha)| = \mathcal{O}(s+r)$, *and a timer ($\pi$-formula) will be* $\mathcal{O}(m+r+s)$ *in length.*

Proof. It is obtained by direct counting. □

5.2. The description of the instruction actions. To improve perception and to facilitate proof, formulae describing the actions of one instruction and the entire Turing machine in one step are constructed with an alternation of universal and existential quantifiers in the subformula $\Delta^{cop}(\widehat{u})$. Still, this alternation is not essential, as noted in Subsection 5.3 — see Remark 3.

The following formula $\varphi(k)$ describes the actions of the kth instruction $M(k) = q_i\alpha_1, \alpha_2 \to q_j\beta_1, \gamma, \beta_2$, including the idle run's instructions (see the beginning of this section) at some step $t+1$[3], where $\alpha_1, \alpha_2, \gamma \in \mathcal{A}$ and $\beta_1, \beta_2 \in$

[3]see Remark 4 below

$\{R, L, S\}$:

$$\begin{aligned}\varphi(k) \quad &\rightleftharpoons \quad (\forall\,\widehat{u},\widehat{U}\,)(\forall\,\widehat{v},\widehat{V}\,)(\forall\,\widehat{h},\widehat{H})\ \Big\{\Big[\pi_t((i)_2,\alpha_1,\alpha_2,\widehat{U},\widehat{u})\ \ \&\\ &\quad \&\ \ \Big(\kappa_1(\beta_1)(\widehat{U},\widehat{V})\ \&\ \kappa_2(\beta_2)(\widehat{u},\widehat{v})\Big)\ \&\ \ \Gamma^{ret}(\beta_1,\beta_2)\Big]\ \rightarrow \\ &\rightarrow\ \ \big[\Delta^{cop}(\widehat{u})\ \&\ \Delta^{wr}(\widehat{u})\ \&\ \pi_{t+1}\big((j)_2,\widehat{H},\widehat{h},\widehat{U}(\beta_1),\widehat{u}(\beta_2)\big]\Big\}.\end{aligned} \tag{4}$$

We will describe the subformulae of $\varphi(k)$ with the free main variables $\widehat{q}_t, \widehat{X},\ \widehat{x}_t, \widehat{Z}_t, \widehat{z}_t, \widehat{D}_t, \widehat{d}_t, \widehat{F}, \widehat{f}_t, \widehat{q}_{t+1}, \widehat{x}_{t+1}, \widehat{Z}_{t+1}, \widehat{z}_{t+1}, \widehat{D}_{t+1},\ \widehat{d}_{t+1}$, and $\widehat{f}_{t+1}$ and the sense of these formulae.

The first π-formula of color t plays the role of *trigger* and "starts up the fulfillment"of the instruction with the prefix "$q_i\alpha_1, \alpha_2 \rightarrow \ldots$" provided that the heads scan the $(\widehat{U})$th cell on the first tape and the $(\widehat{u})$th square on the second. In this case, the numbers $(\widehat{U}(\beta_1))$ and $(\widehat{u}(\beta_2))$ of the cells, which the heads will view after the execution of the instruction $M(k)$, can accordingly be found as $(\widehat{U}(\beta_1)) \rightleftharpoons (\widehat{U} \oplus \widehat{V})$ and $(\widehat{u}(\beta_2)) \rightleftharpoons (\widehat{u} \oplus \widehat{v})$, where the tuples of corrections $\widehat{V}$ and $\widehat{v}$ satisfy the conditions $\kappa_1(\beta_1)(\widehat{U},\widehat{V})$ and $\kappa_2(\beta_2)(\widehat{u},\widehat{v})$ for given meta-symbols β_1, β_2. These conditions almost coincide with the formulae 2 and 3 in Subsection 4.2 ; more precisely:

$$\kappa_1(\beta_1)(\widehat{U},\widehat{V}) \rightleftharpoons \begin{cases}\kappa_{\beta_1}(\widehat{U},\widehat{V}), & \text{if}\ \ \beta_1 \in \{L,R\};\\ \widehat{V} \equiv \widehat{0}, & \text{if}\ \beta_1 = S;\end{cases} \tag{5}$$

and

$$\kappa_2(\beta_2)(\widehat{u},\widehat{v}) \rightleftharpoons \begin{cases}\kappa_{\beta_2}(\widehat{u},\widehat{v}), & \text{if}\ \ \beta_2 \in \{L,R\};\\ \widehat{v} \equiv \widehat{0}, & \text{if}\ \beta_2 = S.\end{cases} \tag{6}$$

The informal sense of subformula $\Gamma^{ret}(\beta_1,\beta_2)$ is the following: it "seeks"the codes $\widehat{H}$ and $\widehat{h}$ of the symbols λ_1 and λ_2, which will be scanned after the current step $t{+}1$ (for this reason, it is named "retrieval"). For this purpose, it "inspects"the squares that are to the right or left of the cells $(\widehat{U})$ and $(\widehat{u})$ (if $\beta_1,\beta_2 \in \{R,L\}$) or it "seeks"nothing on the lth tape (when $\beta_l = S$):

$$\Gamma^{ret}(\beta_1,\beta_2) \rightleftharpoons \Gamma^{ret}_1(\beta_1) \wedge \Gamma^{ret}_2(\beta_2),$$

where $\Gamma^{ret}_1(\beta_1) \rightleftharpoons \psi_1(\widehat{U}(\beta_1) \rightarrow \widehat{H})$ for each β_1, $\Gamma^{ret}_2(\beta_2) \rightleftharpoons \psi_{2,t}(\widehat{u}(\beta_2) \rightarrow \widehat{h})$ for $\beta_2 \in \{R,L\}$, and $\Gamma^{ret}_2(S) \rightleftharpoons \widehat{h} \equiv c\gamma$, as the symbol $\lambda_2 = \gamma$ will be scanned by the second head in this case. Indeed, when $\beta_1 = S$, then $(\widehat{U}(\beta_1)) = (\widehat{U} \oplus \widehat{V}) = (\widehat{U})$ and $\widehat{H} = \widehat{\alpha}_1$, while if $\beta_2 = S$, then $(\widehat{u}(\beta_2)) = (\widehat{u} \oplus \widehat{v}) = (\widehat{u})$ and $\widehat{h} = c\gamma = \lambda_2$.

The formula $\Delta^{cop}(\widehat{u})$ changes the color of records in all the cells of the second tape, whose numbers are different from $(\widehat{u})$; in other words, it "copies"the majority of records on this tape:

$$\Delta^{cop}(\widehat{u}) \ \rightleftharpoons\ \forall\,\widehat{w}\,[\neg(\widehat{w} \equiv \widehat{u}) \rightarrow \exists\,\widehat{g}(\psi_{2,t}(\widehat{w} \rightarrow \widehat{g}) \wedge \psi_{2,t+1}(\widehat{w} \rightarrow \widehat{g}))].$$

The formula $\Delta^{wr}(\widehat{u})$ "puts"the symbol γ of color $t{+}1$ in the $(\widehat{u})$th square on the second tape: $\Delta^{wr}(\widehat{u}) \rightleftharpoons \psi_{2,t+1}(\widehat{u} \to \gamma)$.

The second π-formula of the color $t+1$ "aims"the heads at the $(\widehat{U} \oplus \widehat{V})$th and $(\widehat{u} \oplus \widehat{v})$th cells; "places"the symbols $\widehat{H}$ and $\widehat{h}$ in these locations; and "changes"the machine state's number on j:

$$\widehat{Z}_{t+1} \equiv \widehat{U} \oplus \widehat{V} \ \wedge\ \widehat{z}_{t+1} \equiv \widehat{u} \oplus \widehat{v} \ \wedge\ \widehat{D}_{t+1} \equiv \widehat{H} \ \wedge\ \widehat{d}_{t+1} \equiv \widehat{h} \ \wedge\ \widehat{q}_{t+1} \equiv (j)_2.$$

Lemma 3. *If the indices are left out of account, then* $|\varphi(k)| = \mathcal{O}(m+r+s)$.

Proof. This follows from Lemmata 1 and 2 by immediate calculation. □

5.3. The description of the running steps and configurations. The Meyer and Stockmeyer technique will be applied to describe the machine's actions over an exponential period.

5.3.1. One step. Let N be the number of the machine's instructions P, together with $2|\mathcal{A}|^2$ instructions of the idle run (see the beginning of this section). The formula $\Phi^{(0)}(P)$ that describes one step (whose number is $t{+}1$) of the machine P run is of the form:

$$\Phi^{(0)}(P)(\widehat{y}_t, \widehat{y}_{t+1}) \ \rightleftharpoons \ \bigwedge_{0< k\leqslant N} \varphi(k)(\widehat{y}_t, \widehat{y}_{t+1}), \tag{7}$$

where $\widehat{y}_t \rightleftharpoons \langle \widehat{q}_t, \widehat{X}, \widehat{x}_t, \widehat{Z}_t, \widehat{z}_t, \widehat{D}_t, \widehat{d}_t, \widehat{F}, \widehat{f}_t \rangle$ and $\widehat{y}_{t+1} \rightleftharpoons \langle \widehat{q}_{t+1}, \widehat{X}, \widehat{x}_{t+1}, \widehat{Z}_{t+1}, \widehat{z}_{t+1}, \widehat{D}_{t+1}, \widehat{d}_{t+1}, \widehat{F}, \widehat{f}_{t+1} \rangle$ are two tuples of its $2{\cdot}(2m{+}5r{+}2s)$ free variables.

Remark 3. ***a)***. *Let $\langle\chi\rangle$ be a quantifier-free part of a formula χ. Using the well-known Tarski and Kuratowski algorithm, we obtain that the prenex-normal form of $\varphi(k)$ is $\forall\, \widehat{U}, \widehat{u}, \widehat{V}, \widehat{v}\, \forall \widehat{H}, \widehat{h}\, \forall\, \widehat{w}\, \exists\, \widehat{g} \langle \varphi(k) \rangle$. Recall that the variables $\widehat{U}, \widehat{u}, \widehat{V}, \widehat{v}$, $\widehat{H}, \widehat{h}, \widehat{w}$, and $\widehat{g}$ are auxiliary and "attached"to the corresponding basic variables (see Subsection 5.1); at that, the variables $\widehat{g}$ "service"only $\widehat{f}_t$ and $\widehat{f}_{t+1}$. Hence, the tuple $\widehat{g}$ has a length of $r+1$, but not $m+1 = m(n)+1$ and not $s+1 = \lceil \log n \rceil + 1$, i.e., its length does not depend on $n = |X|$. Therefore, we can substitute the subformula $\Delta^{cop}(\widehat{u})$ with $\Delta_1^{cop}(\widehat{u}) \rightleftharpoons \forall \widehat{w} \bigvee_{\rho\in\mathcal{A}} \langle \Delta^{cop}(\widehat{u})(\rho) \rangle$, where all possible tuples $c\rho$ for $\rho \in A$ are placed instead of the tuples $\widehat{g}$, since the sense of the basic variables $\widehat{f}_t$ and $\widehat{f}_{t+1}$ is "storing"information about the alphabet symbols. It is clear that $|\Delta_1^{cop}(\widehat{u})| \leqslant |\mathcal{A}| \cdot |\Delta^{cop}(\widehat{u})|$ because one does not need to write a prefix $\exists \widehat{g}$. Thus, the formulae $\varphi(k)$ and $\Phi^{(0)}(P)(\widehat{y}_t, \widehat{y}_{t+1})$ are universal in essence.*

b). *The formula $\varphi(k)$ does not depend on concrete entries on the tapes at instant t but depends only on the length of the input string n (since $s+1 = \lceil \log n \rceil + 1$ is the number of variables $Z_{t,i}$, X_i, and $Z_{t+1,i}$) and the value of function $m(n)$ (which equals to the number of variables $x_{t,j}$, $z_{t,j}$, $z_{t+1,j}$, and $x_{t+1,j}$) at this concrete value n. Therefore, the whole formula $\Phi^{(0)}(P)$ also depends only on the parameter n and the function m for a fixed program P. There is a full analogy with programs for real computers, which can contain parameters depending on the input length, but not the input string itself.*

For example, a program for multiplying square matrices in programming languages Algol, BASIC, Pascal, C#, C^{++}, etc., can contain a parameter n, which is the number of rows and entered additionally as a rule (certainly, this program is a learning exercise). However, a competently written program should not depend on specific values of elements of matrices to be multiplied.

Lemma 4. *(i) If $\widehat{x}_t \neq \widehat{\mu}$, then a clause $\psi_{2,t}(\widehat{\mu} \to \varepsilon)$ will be true independently of the values of variables $\widehat{f}_t$. In particular, any quasi-equation, which is contained in the record of $\langle \Delta^{cop}(\widehat{u}) \rangle$, will be true if its color is t or $t+1$, and at the same time $\widehat{x}_t \neq \widehat{w}$ or $\widehat{x}_{t+1} \neq \widehat{w}$, respectively.*

(ii) When $\widehat{X} \neq \widehat{\mu}$, then a clause $\psi_1(\widehat{\mu} \to \varepsilon)$ will be true independently of the values of variables $\widehat{F}$.

(iii) For some constant C_1, the inequality $|\Phi^{(0)}(P)(\widehat{y}_t, \widehat{y}_{t+1})| \leqslant C_1 \cdot |P| \cdot |\varphi(N)|$ holds, provided that the program P is not empty.

Proof. (i),(ii) The premises of the clauses are false in these cases.
(iii) If $N - 2|\mathcal{A}|^2 \neq 0$ (see the definition-equation (7) and the beginning of this section), then $N \cdot \lceil \log N \rceil < C_2 \cdot |P|$; here $\log N$ locations are required to number (using indices k for $1 \leqslant k \leqslant N$) the tuples of auxiliary variables $\widehat{u}, \widehat{U}, \widehat{v}, \widehat{V}, \widehat{h}, \widehat{H}, \ldots$ entered in the record of each formula $\varphi(k)$. This implies the lemma assertion. □

5.3.2. The exponential quantity of steps. The following formulae $\Phi^{(k)}(P)(\widehat{y}_t, \widehat{y}_{t+e(k)})$ conform to the actions of machine P over the period of time $e(k) \rightleftharpoons \exp(k)$. They are defined by induction on k:

$$\Phi^{(k+1)}(P) \rightleftharpoons \exists \widehat{Y} \forall \widehat{a} \forall \widehat{b} \big\{ \big[(\widehat{y}_t \equiv \widehat{a} \wedge \widehat{Y} \equiv \widehat{b}) \vee (\widehat{Y} \equiv \widehat{a} \wedge \widehat{b} \equiv \widehat{y}_{t+e(k+1)}) \big] \to \\ \to \Phi^{(k)}(P)(\widehat{a}, \widehat{b}) \big\},$$

where $\widehat{Y}, \widehat{a}, \widehat{b}$ are the tuples of the new auxiliary variables $(2m+5r+2s)$ in length ignoring the indices.

Remark 4. *The definition of the formula $\varphi(k)$, which describes the actions of the k-th instruction by the equation (4), includes as indices the parameters t and $t+1$, which denote the numbers of configurations that arise before and after the $t+1$-th step, respectively. They are included for definiteness; their presence makes it easier to perceive. Now we see that new variables gradually replace the ones that contained these indices. In the end, only those with $t=0$ and $t+1 = T = \exp(m)$ will remain. At the same time, the total number of variables decreases as the parameter k grows.*

5.3.3. The description of configurations. Let $L_2(t)$ be a configuration recorded on the second tape after step t (it may be unrealizable). Namely, every cell, whose number is $(\widehat{\mu})$, contains a symbol $\varepsilon(\widehat{\mu})$; the scanned squares have the numbers $(\widehat{\zeta}_1)$ and $(\widehat{\zeta}_2)$; and a machine is ready to execute an instruction $q_i\alpha_1, \alpha_2 \to \ldots$. The following formula corresponds to this configuration:

$$\Psi L_2(t)(\widehat{y}_t) \rightleftharpoons \pi_t((i)_2, \alpha_1, \alpha_2, \widehat{\zeta}_1, \widehat{\zeta}_2) \mathbin{\&} \bigwedge_{0 \leqslant (\widehat{\mu}) \leqslant T} \psi_{2,t}(\widehat{\mu} \to \varepsilon(\widehat{\mu})),$$

where $T = \exp(m)$. Let us notice that this formula also indicates the location of the head on the first tape and describes that symbol, which is in this cell.

It remains only to describe the entries in the cells of the first tape:

$$\Psi K_1(\widehat{X}, \widehat{F}) \rightleftharpoons \psi_1(\widehat{0} \to \triangleright) \mathbin{\&} \bigwedge_{1 \leqslant (\widehat{\xi}) \leqslant n} \psi_1(\widehat{\xi} \to \chi(\widehat{\xi})) \mathbin{\&}$$
$$\mathbin{\&} \forall \widehat{u}_0 \left(\widehat{u}_0 > (n)_2 \ \to \ \psi_1(\widehat{u}_0 \to \Lambda)\right),$$

here the letter "L"is replaced by the letter "K"because these records on the first tape are undoubtedly real. The meaning of this entire formula is as follows: the $\triangleright$ symbol is located in the zeroth cell, the input string $X = \langle \chi_1, \ldots, \chi_n \rangle$ is written next to $\triangleright$, and the remaining cells of the tape are empty.

The configuration $C(t) = K_1 \cup L_2(t)$ of both tapes together at instant t is described by the formula

$$\Psi(C)(\widehat{y}_t) \rightleftharpoons \Psi K_1(\widehat{X}, \widehat{F}) \mathbin{\&} \Psi L_2(t)(\widehat{y}_t).$$

6 The simulation of the computations

So far, we have only associated the previously constructed formulae with some program components or processes. However, it cannot be argued that these formulae model something, i.e., they will not always become true, when the described events occur in reality.

Let $C(t)$ and $C(t+1)$ be some adjacent configurations, we definite

$$\Omega^{(0)}(X, P)(\widehat{y}_t, \widehat{y}_{t+1}) \rightleftharpoons [\Psi C(t)(\widehat{y}_t) \mathbin{\&} \Phi^{(0)}(P)(\widehat{y}_t, \widehat{y}_{t+1})] \to \Psi C(t+1)(\widehat{y}_{t+1}).$$

We will prove that the sentence $\forall \widehat{y}_t \forall \widehat{y}_{t+1} \Omega^{(0)}(X, P)(\widehat{y}_t, \widehat{y}_{t+1})$ is true if and only if the machine P transforms the configuration $C(t)$ to $C(t+1)$ in one step, i.e., this formula models the machine actions at the step $t+1$.

Remark 5. *Since the subformula $\Psi K_1(\widehat{X}, \widehat{F})$ does not depend on the step number, we can focus only on the subformulae $\Psi K_2(t)$ and $\Psi K_2(t+1)$ of formulae $\Psi C(t)$ and $\Psi C(t+1)$ in this section.*

6.1. The single-valuedness of the modeling of one step. Let $C(t+1) = K_1 \cup K_2(t+1)$ be a configuration that has arisen from a configuration $C(t) = K_1 \cup K_2(t)$ as a result of the machine P's action at the step $t+1$.

Proposition 2. *(i) There exist special values of variables $\widehat{y}_t$ such that the formula $\Psi C(t)(\widehat{y}_t)$ is true, and the truth of $\Phi^{(0)}(P)(\widehat{y}_t, \widehat{y}_{t+1})$ follows from the truth of $\Psi C(t+1)(\widehat{y}_{t+1})$ for every tuple $\widehat{y}_{t+1}$.*

(ii) If a sentence $\forall\widehat{y}_t\forall\widehat{y}_{t+1}\Omega^{(0)}(X,P)(\widehat{y}_t,\widehat{y}_{t+1})$ is true, then the machine P cannot convert the configuration $C(t)$ into a configuration which differs from $C(t+1)$, at the step $t+1$.

Proof. We will prove these assertions simultaneously. Namely, we will select the values of the variables $\widehat{y}_t$ and $\widehat{y}_{t+1}$ such that a formula

$$\Upsilon_{t+1}(\widehat{y}_t,\widehat{y}_{t+1})\rightleftharpoons[\Psi K_2(t)(\widehat{y}_t)\ \&\ \Phi^{(0)}(P)(\widehat{y}_t,\widehat{y}_{t+1})]\rightarrow\Psi L_2(t+1)(\widehat{y}_{t+1})$$

will be false if the configuration on the second tape $L_2(t+1)$ differs from the real $K_2(t+1)$. This implies Item (ii) of the proposition. However, in the beginning, we will select the special values of the variables of the tuple $\widehat{y}_t$. After that, when we pick out the suitable values of the corresponding variables of the color $t+1$, the formulae $\Psi K_2(t+1)(\widehat{y}_{t+1})$ and $\Phi^{(0)}(P)(\widehat{y}_t,\widehat{y}_{t+1})$ will become true or false at the same time depending on the values of the variables $\widehat{y}_{t+1}$.

Let $M(k)=q_i\alpha_1,\alpha_2\rightarrow\ \dots$ be an instruction that is applicable to the configuration $C(t)$; $(\widehat{\zeta}_1)$ and $(\widehat{\zeta}_2)$ be the numbers of the scanned squares. We define $\widehat{q}_t=(i)_2$, $\widehat{D}_t=c\alpha_1$, $\widehat{d}_t=c\alpha_2$, $\widehat{Z}_t=\widehat{\zeta}_1$, $\widehat{z}_t=\widehat{\zeta}_2$. Then π-formula $\pi_t((i)_2,\alpha_1,\alpha_2,\widehat{\zeta}_1,\widehat{\zeta}_2)$, which is in the record of the formula $\Psi C(t)(\widehat{y}_t))$, becomes true.

Now we consider a formula $\varphi(l)$ that conforms to some instruction $M(l)=q_b\theta_1,\theta_2\rightarrow\dots$ that differs from $M(k)$. This formula has a timer $\pi_t((b)_2,\theta_1,\theta_2,\widehat{U},\widehat{u})$ as the first premise. For the selected values of the variables $\widehat{q}_t$, $\widehat{D}_t$, and $\widehat{d}_t$, this timer takes the form of $(i)_2\equiv(b)_2\wedge c\alpha_1\equiv c\theta_1\wedge c\alpha_2\equiv c\theta_2\wedge\dots$. It is obvious that if $i\neq b$, or $\alpha_1\neq\theta_1$, or $\alpha_2\neq\theta_2$, then this π-formula will be false, and the whole $\varphi(l)$ will be true.

Thus, let $\varphi(k)$ be a formula corresponding to the instruction $M(k)=q_i\alpha_1,\alpha_2\rightarrow q_j\beta_1,\gamma,\beta_2$. The quantifier-free part of this formula can become false only for $\widehat{U}=\widehat{\zeta}_1$ and $\widehat{u}=\widehat{\zeta}_2$ since the values of variables $\widehat{Z}_t$ and $\widehat{z}_t$ are such. Therefore, we will consider the case when $\widehat{U}=\widehat{\zeta}_1$ and $\widehat{u}=\widehat{\zeta}_2$. For the same reason, we pick out $\widehat{V}=\widehat{\eta}_1$ and $\widehat{v}=\widehat{\eta}_2$, where the tuple $\widehat{\eta}_l$ satisfies the condition $\kappa_l(\beta_l)(\widehat{\zeta}_l,\widehat{\eta}_l)$ for $l=1,2$ (see the formulae (5) and (6) in Subsection 5.2). So, the number of the cell, which a head will scan after step $t+1$ on the lth tape, equals $(\widehat{\zeta}_l(\beta_l))=(\widehat{\zeta}_l\oplus\widehat{\eta}_l)$.

Let us assign $\widehat{x}_t=\widehat{\zeta}_2(\beta_2)$ and $\widehat{X}=\widehat{\zeta}_1(\beta_1)$. Since $\widehat{u}(\beta_2)=\widehat{\zeta}_2(\beta_2)$, the quasi-equation, of the color t, which enters in $\langle\Delta^{cop}(\widehat{u})\rangle$ (i.e., in the quantifier-free part of $\Delta^{cop}(\widehat{u})$), is true for all $\widehat{w}\neq\widehat{\zeta}_2(\beta_2)$ irrespective of the values of the tuples $\widehat{f}_t$ and $\widehat{g}$ according to Lemma 4(i,ii); while when $\widehat{w}=\widehat{\zeta}_2$, the premise of $\langle\Delta^{cop}(\widehat{u})\rangle$ is false. For the same reason, all clauses that are included in $\Psi K_2(t)$ and ΨK_1 are true, except the clauses $\psi_{2,t}(\widehat{\zeta}_2(\beta_2)\rightarrow\lambda_2)$ and $\psi_1(\widehat{\zeta}_1(\beta_1)\rightarrow\lambda_1)$, where λ_l is that symbol, which a head will view after step $t+1$ on the lth tape for $l=1,2$. We set the values of two tuples, $\widehat{F}$ and $\widehat{f}_t$, as $c\lambda_1$ and $c\lambda_2$, respectively. Now, the questionable clauses from $\Psi C(t)$ become true because their premises and conclusions are such. All clauses of color t from $\langle\varphi(k)\rangle$ also become true.

So, we have selected the values of the variables $\widehat{y}_t$, i.e., of all basic ones of color t. It remains only to assign the values of the variables of color $t+1$.

We recall that λ_l is the symbol that will be in the $(\widehat{\zeta}_l(\beta_l))$th cell on lth tape for $l=1,2$ at the instant $t+1$. We define $\widehat{D}_{t+1}=c\lambda_1$, $\widehat{d}_{t+1}=c\lambda_2$, $\widehat{Z}_{t+1}=\widehat{\zeta}_1(\beta_1)=\widehat{\zeta}_1\oplus\widehat{\eta}_1$, $\widehat{z}_{t+1}=\widehat{\zeta}_2(\beta_2)=\widehat{\zeta}_2\oplus\widehat{\eta}_2$, and $\widehat{q}_{t+1}=(j)_2$.

With this assignment and the chosen values of variables $\widehat{U}$, $\widehat{u}$, $\widehat{V}$, and $\widehat{v}$, the timer $\pi_{t+1}((j)_2,\lambda_1,\lambda_2,\widehat{\zeta}_1(\beta_1),\widehat{\zeta}_2(\beta_2))$, which enters into the record of $\Psi K_2(t+1)$, becomes true; while the π-formula contained in $\Psi L_2(t+1)$ is false if it differs from "proper". However, the conclusion of the quantifier-free part $\langle\varphi(k)\rangle$ of the formula $\varphi(k)$ contains a slightly different timer $\pi_{t+1}((j)_2,\widehat{H},\widehat{h},\widehat{U}(\beta_1),\widehat{u}(\beta)_2)$. In this trigger, the equivalencies $\widehat{D}_{t+1}\equiv\widehat{H}$ and $\widehat{d}_{t+1}\equiv\widehat{h}$ raise doubts for the time being.

If $\widehat{H}\neq c\lambda_1$ or $\widehat{h}\neq c\lambda_2$, then the formula $\Gamma^{ret}(\beta_1,\beta_2)$ will be false for the assigned values of $\widehat{X}$, $\widehat{x}_t$, $\widehat{F}$, and $\widehat{f}_t$ and the calculated $\widehat{\zeta}_1(\beta_1)$ and $\widehat{\zeta}_2(\beta_2)$. Hence, the whole formula $\langle\varphi(k)\rangle$ will be true in this case because its third premise, $\Gamma^{ret}(\beta_1,\beta_2)$, is false. When $\widehat{H}=c\lambda_1$ and $\widehat{h}=c\lambda_2$, the terminal π-formula in $\langle\varphi(k)\rangle$ becomes true.

When the "incorrect"formula $\Psi L_2(t+1)$ has a mistake in the record of the clause $\psi_{2,t+1}(\widehat{\zeta}_2\to\gamma)$ (see the definition of the formula $\Delta^{cop}(\widehat{u})$ in Subsection 5.2), we will assign $\widehat{x}_{t+1}=\widehat{\zeta}_2$ and $\widehat{f}_{t+1}=c\,\gamma$. If this fragment is as it should be, but there is another "incorrect"clause $\psi_{t+1}(\widehat{\mu}\to\rho)$, where ρ is different from "real"δ, we will define $\widehat{x}_{2,t+1}=\widehat{\mu}$ and $\widehat{f}_{t+1}=\widehat{g}=c\delta$ (we note that 1) this is the only case when we need to set the values of the variables $\widehat{g}$; 2) in this case, we essentially use the formula $\Delta^{cop}(\widehat{u})$ in the form of disjunction — see Remark 3). Now, the quasi-equations of the color $t+1$ in the formulae $\langle\Delta^{cop}(\widehat{u})\rangle$ and $\Delta^{wr}(\widehat{u})$ are valid in both of these cases on the grounds of Lemma 4(i,ii) or because their premises and conclusions are true. Therefore, the whole formula $\langle\varphi(k)\rangle$ is true. All the clauses contained in $\Psi K_2(t+1)$ are true for the same reasons.

We obtain as a result that any formula $\varphi(l)$ is true for the above-selected values of the primary variables, so the entire conjunction $\Phi^{(0)}(P)$ is true. Since the premise and conclusion of the $\Omega^{(0)}(X,P)(\widehat{y}_t,\widehat{y}_{t+1})$ are true, and the configurations $K_2(t+1)$ and $L_2(t+1)$ are different, the "incorrect"formula Υ_{t+1} is false because of the choice of the values of basic variables.

As the configuration $L_2(t+1)$ may differ from the real $K_2(t+1)$ in any place, Item (i) is established too. □

6.2. The sufficiency of the modeling of one step. We will now prove a converse to Proposition 2(ii).

Proposition 3. *Let* $C(t+1)=K_1\cup K_2(t+1)$ *be a configuration that has arisen from a configuration* $C(t)=K_1\cup K_2(t)$ *as a result of an action of the* *machine P at the step $t{+}1$. Then the formula $\Omega^{(0)}(X,P)(\widehat{y}_t,\widehat{y}_{t+1})$ is true for all the values variables $\widehat{y}_t$ and $\widehat{y}_{t+1}$.*

Proof. Let $M(k)=q_i\alpha_1,\alpha_2\to q_j,\beta_1,\gamma,\beta_2$ be the instruction that transforms the configuration $C(t)$ into $C(t{+}1)$; and $\varphi(k)(\widehat{y}_t,\widehat{y}_{t+1})$ be a formula, which is written for this instruction. This formula is a consequence of $\Phi^{(0)}(P)(\widehat{y}_t,\widehat{y}_{t+1})$.

Let us replace $\varphi(k)$ by a conjunction of formulae $\varphi(k)(\widehat{\mu}_1,\widehat{\mu}_2)$; they are each obtained as the result of the substitution of the various values of the universal variables $\widehat{U}$ and $\widehat{u}$ for the variables themselves. Every formula $\varphi(k)(\widehat{\mu}_1,\mu_2)$ contains the subformula $\widehat{q}_t\equiv(i)_2\wedge\widehat{D}_t\equiv c\alpha_1\wedge\widehat{d}_t\equiv c\alpha_2\wedge\widehat{Z}_t\equiv\widehat{\mu}_1\wedge\widehat{z}_t\equiv\widehat{\mu}_2$ as the first premise. One of these subformulae coincides with the only timer $\pi_t((i)_2,\alpha_1,\alpha_2,\widehat{\zeta}_2,\widehat{\zeta}_2)$ included in $\Psi K_2(t)$ for $\widehat{U}=\widehat{\mu}_1=\widehat{\zeta}_1$ and $\widehat{u}=\widehat{\mu}_2=\widehat{\zeta}_2$, as the instruction $M(k)$ is applicable to the configuration $C(t)$.

For these $\widehat{U}=\widehat{\zeta}_1$ and $\widehat{u}=\widehat{\zeta}_2$, there are tuples $\widehat{\eta}_1$ and $\widehat{\eta}_2$ such that the conditions $\kappa_1(\beta_1)(\widehat{U},\widehat{V})$ and $\kappa_2(\beta_2)(\widehat{u},\widehat{v})$ are valid for $\widehat{V}=\widehat{\eta}_1$ and $\widehat{v}=\widehat{\eta}_2$. So, the second premise of $\varphi(k)$ is true for suitable values of variables $\widehat{U}$, $\widehat{u}$, $\widehat{V}$, and $\widehat{v}$.

Using these $\widehat{\eta}_1$ and $\widehat{\eta}_2$, one can find $\widehat{U}(\beta_1)$ and $\widehat{u}(\beta_2)$ correspondently as $\widehat{\zeta}_1\oplus\widehat{\eta}_1$ and $\widehat{\zeta}_2\oplus\widehat{\eta}_2$. We recall that the machine P cannot go beyond the left edges of tapes, hence $0\leqslant(\widehat{\zeta}_1(\beta_1))\leqslant n{+}1$ and $0\leqslant(\widehat{\zeta}_2(\beta_2))\leqslant T=\exp(m)$ (see the second paragraph in Subsection 5.1); therefore the formulae ΨK_1 and $\Psi K_2(t)$ accordingly contain the quasi-equations $\psi_1(\widehat{\zeta}_1(\beta_1)\to\lambda_1)$ and $\psi_{2,t}(\widehat{\zeta}_2(\beta_2)\to\lambda_2)$ for some symbols λ_1 and λ_2. The conjunction of these clauses coincides with $\Gamma^{ret}(\beta_1,\beta_2)$, when $\beta_1\in\{L,R,S\}$ and $\beta_2\in\{L,R\}$ for $\widehat{H}=c\lambda_1$ and $\widehat{h}=c\lambda_2$; while if $\beta_2=S$, then $\Gamma_2^{ret}(\beta_2)$ becomes true equivance for $\widehat{h}=c\,\gamma$.

So, each of the three premises in the formula $\varphi(k)$ coincides with the appropriate subformula of $\Psi C(t)$ or is true for some values of universal auxiliary variables from the list at the beginning of $\varphi(k)$. It follows from this that

$$\Psi K_2(t)\ \ \&\ \ \Delta^{cop}(\widehat{\zeta}_2)\ \ \&\ \ \Delta^{wr}(\widehat{\zeta}_2)\ \ \&\ \ \pi_{t+1}((j)_2,\lambda_1,\lambda_2,\widehat{\zeta}_1(\beta_1),\widehat{\zeta}_2(\beta_2))$$

is the consequence of the formulae $\Psi K_2(t)$ and $\Phi^{(0)}(P)(\widehat{y}_t,\widehat{y}_{t+1})$.

The formula $\Delta^{cop}(\widehat{\zeta}_2)$ begins with the quantifiers $\forall\,\widehat{w}$. Let us replace this formula with a conjunction that is equivalent to it; we substitute all possible values of variables $\widehat{w}$ into this formula to this effect. For every value of $\widehat{w}$, there is a unique value of the tuple $\widehat{g}$ such that the clause $\psi_{2,t}(\widehat{w}\to\widehat{g})$ enters into the formula $\Psi K_2(t)$. When these values of $\widehat{g}$ are substituted in their places, we will obtain all the quasi-equations from $\Psi K_2(t{+}1)$, except one, which coincides with $\Delta^{wr}(\widehat{\zeta}_2)$. □

6.3. The simulation of the exponential computations. We define the formulae that model $e(k)\rightleftharpoons\exp(k)$ running steps of a machine P when it applies to a configuration $C(t)$:

$$\Omega^{(k)}(X,P)(\widehat{y}_t,\widehat{y}_{t+e(k)}) \rightleftharpoons [\Psi C(t)(\widehat{y}_t)\ \&\ \Phi^{(k)}(P)(\widehat{y}_t,\widehat{y}_{t+e(k)})] \to \\ \to \Psi C(t+e(k))(\widehat{y}_{t+e(k)}).$$

Proposition 4. *Let $t,k\geqslant 0$ be the integers such that $t+e(k)\leqslant T=\exp(m)$.*

(i) If the machine P transforms the configuration $C(t)$ to $C(t+e(k))$ within $e(k)$ steps, then there are special values of variables $\widehat{y}_t$ such that the formula $\Psi C(t)(\widehat{y}_t)$ is true, and for all $\widehat{y}_{t+e(k)}$, whenever the formula $\Psi C(t+e(k))(\widehat{y}_{t+e(k)})$ is true, $\Phi^{(k)}(P)(\widehat{y}_t,\widehat{y}_{t+e(k)})$ is also true.

(ii) The formula $\Omega^{(k)}(X,P)(\widehat{y}_t,\widehat{y}_{t+e(k)})$ is valid for all values of variables $\widehat{y}_t,\widehat{y}_{t+e(k)}$ (i.e., identically true) if and only if the machine P converts the configuration $C(t)$ into $C(t+e(k))$ within $e(k)$ steps.

Proof. Induction on the parameter k. For $k=0$, Item (i) is Proposition 2(i), and Item (ii) follows from Propositions 2(ii) and 3.

We start the proof of the inductive step by rewriting the formula $\Phi^{(k+1)}(P)(\widehat{y}_t,\widehat{y}_{t+e(k+1)})$ in the equivalent, but longer form:

$$\exists\widehat{Y}\big\{\forall\widehat{a}\forall\widehat{b}\big[(\widehat{y}_t\equiv\widehat{a}\ \wedge\ \widehat{Y}\equiv\widehat{b})\to\Phi^{(k)}(P)(\widehat{a},\widehat{b})\big]\quad\& \\ \&\quad \forall\widehat{a}\forall\widehat{b}\big[(\widehat{Y}\equiv\widehat{a}\ \wedge\ \widehat{b}\equiv\widehat{y}_{t+e(k+1)})\to\Phi^{(k)}(P)(\widehat{a},\widehat{b})\big]\big\}.$$

The following formula results from this immediately:

$$\Xi_{k+1}(\widehat{y}_t,\widehat{y}_{t+e(k+1)}) \rightleftharpoons \exists\widehat{Y}\big\{\Phi^{(k)}(P)(\widehat{y}_t,\widehat{Y})\ \&\ \Phi^{(k)}(P)(\widehat{Y},\widehat{y}_{t+e(r+1)})\big\}.$$

On the other hand, each of the two implications which are included in the long form of the formula $\Phi^{(k+1)}(P)(\widehat{y}_t,\widehat{y}_{t+e(k+1)})$ can be false only when the equivalences existing in its premise are valid. Hence, this formula is tantamount to Ξ_{k+1}.

Let the machine P transform the configuration $C(t)$ into $C(t+e(k))$ within $e(k)$ steps and convert the latter into $C(t+e(k+1))$ within the same time.

By the inductive hypothesis of Item (ii) (we recall that the induction is carried out over a single parameter k), the formula $\Omega^{(k)}(P)(\widehat{y}_t,\widehat{y}_{t+e(k)})$ is identically true for any t such that $t+e(k)\leqslant T$, and hence, it is identically true for an arbitrarily chosen t_0 and $t_1=t_0+e(k)$ provided that $t_0+e(k+1)=t_1+e(k)\leqslant T$. Thus, the formulae

$$[\Psi C(t_0)(\widehat{y}_{t_0})\ \&\ \Phi^{(k)}(P)(\widehat{y}_{t_0},\widehat{y}_{t_0+e(k)})] \to \Psi C(t_0+e(k))(\widehat{y}_{t_0+e(k)})$$

and

$$[\Psi C(t_0+e(k))(\widehat{y}_{t_0+e(k)})\ \&\ \Phi^{(r)}(P)(\widehat{y}_{t_0+e(k)},\widehat{y}_{t_0+e(k+1)})] \to \\ \to \Psi C(t_0+e(k+1))(\widehat{y}_{t_0+e(k+1)})$$

are identically true. Therefore, when we change the variables $\widehat{y}_{t_0+e(k)}$ under the sign of the quantifier after partial universalization, we obtain that the following formula

$$\forall\widehat{Y}\{[\Psi C(t_0)(\widehat{y}_{t_0})\ \&\ \Phi^{(k)}(P)(\widehat{y}_{t_0},\widehat{Y})\ \&\ \Phi^{(k)}(P)(\widehat{Y},\widehat{y}_{t_0+e(r+1)})]\ \rightarrow \\ \rightarrow\ \ \Psi C(t_0{+}e(k{+}1))(\widehat{y}_{t_0+e(k+1)})\}$$

is also identically true. This formula is equivalent to

$$[\Psi C(t_0)(\widehat{y}_{t_0})\ \&\ \Xi_{k+1}(\widehat{y}_t,\widehat{y}_{t+e(k+1)})]\ \rightarrow\ \Psi C(t{+}e(k{+}1)$$

because the universal quantifiers will be interchanged with the quantifiers of existence when they are introduced into the premise of the implication. Since the premise of the formula $\Omega^{(k+1)}(X,P)(\widehat{y}_t,\widehat{y}_{t+e(k+1)})$ is equivalent to $\Psi C(t)\ \&\ \Xi_{k+1}$ under the preceding argument, the inductive step of Item (ii) is proven in one direction.

Now, let the configurations $L(t{+}e(k{+}1))$ and $C(t{+}e(k{+}1))$ be different. For some values $\widehat{Y}_1$ and $\widehat{Y}_2$ of variables $\widehat{y}_{t+e(k)}$ and $\widehat{y}_{t+e(k+1)}$, respectively, the formula $\{[\Psi C(t{+}e(k))\ \&\ \Phi^{(k)}(P)] \rightarrow \Psi L(t{+}e(k{+}1))\}(\widehat{Y}_1,\widehat{Y}_2)$ is false by the inductive assumption of Item (ii). Therefore, its conclusion $\Psi L(t{+}e(k{+}1))(\widehat{Y}_2)$ is false, but both its premises $\Phi^{(k)}(P)(\widehat{Y}_1,\widehat{Y}_2)$ and $\Psi C(t{+}e(k))(\widehat{Y}_1)$ are true. Since the last formula and $\Psi C(t)(\widehat{Y}_0)$ are true for some special $\widehat{Y}_0$, which exists due to the induction proposition of Item (i), the formula $\Phi^{(k)}(P)(\widehat{Y}_0,\widehat{Y}_1)$ is true. So, the implication

$$\{[\Psi C(t)\ \&\ \Phi^{(k+1)}(P)] \rightarrow \Psi L(t{+}e(k{+}1))\}(\widehat{Y}_0,\widehat{Y}_2)$$

has a true premise, and a false conclusion; therefore, it is not identically true. Item (ii) is proven.

Since the configuration $L(t{+}e(k{+}1))$ may differ from the proper one at any position, to finish the proof of Item (i), we set the values $\widehat{Y}_1$ of the variables $\widehat{y}_{t+e(k)}$ in a specific manner, using the inductive hypothesis of Item (ii) and then Item (i) for the configurations $C(t+e(k))$ and $C(t+e(k+1))$, as P transforms the first of them to the second. □

7 End of the proof of the main theorem

7.1. The short recording of the initial configuration and the condition of the successful run termination. We cannot fully describe the final configuration on the second tape when the machine arrives at one of the two states, q_{acc} or q_{rej}. Firstly, this final configuration may be very long. Secondly, and most importantly, it is unknown to us — we only know the input string X on the first tape and the machine's program P. However, we do not need the entire final configuration.

Since we have the instructions for the machine's run at idle (see the beginning of Section 5), the statement that the machine P accepts an input string X within $T{=}\exp(m(n))$ steps can be written rather briefly — utilizing one quantifier-free formula of the color T: $\chi(\omega)\ \rightleftharpoons\ \widehat{q}_T\ \equiv\ (1)_2$ because the accepting state has the number one. This formula is a full analog of subformula I of Cook's formula from [3], and it has $\mathcal{O}(r)$ symbols in length (see Section 5.1), without considering the indices. The writing of the first index T (in the tuple $\widehat{q}_T$) occupies $m{+}1 = m(n){+}1$ bits. The maximum length of the second indices is $\mathcal{O}\lceil\log r\rceil$, so we have $|\chi(\omega)| = \mathcal{O}(r\cdot(m(n){+}\lceil\log r\rceil))$.

The formula $\Psi C(t)(\widehat{y}_t)$ was introduced in Subsection 5.3 to describe a configuration arising after step t. It is a very lengthy: $|\Psi C(t)| > m\cdot\exp(m)$. However, the initial configuration is simple enough and allows us to give a relatively short description. We have already described the entries on the first tape in Subsection 5.3.3 since they remain unchanged throughout the work. This is done using the following formula $\chi_1(X)$ with the free variables $\widehat{X} = \langle X_0,\dots,X_s\rangle$ and $\widehat{F} = \langle F_0,\dots,F_r\rangle$:

$$\chi_1(X) \rightleftharpoons \Psi K_1(\widehat{X},\widehat{F}) \rightleftharpoons \psi_1(\widehat{0}\to\rhd) \mathbin{\&} \bigwedge_{1\leqslant\,(\widehat{\xi})\leqslant n} \psi_1(\widehat{\xi}\to\chi(\widehat{\xi})) \mathbin{\&} \tag{8}$$
$$\mathbin{\&} \quad \forall\widehat{u}_0\,\big(\widehat{u}_0 > (n)_2 \;\to\; \psi_1(\widehat{u}_0\to\Lambda)\big).$$

Let us pay attention that the variable X denotes the input string, but not the tuple of the variables $\widehat{X}$. The sense of $\chi_1(X)$ is as follows: the $\rhd$ symbol is located in the zeroth cell, the input string $X = \langle\chi_1,\dots,\chi_n\rangle$ is written next to $\rhd$ in the squares $1,2,\dots,n$, and the remaining cells of the tape are empty.

The entries on the second tape before starting work are described even more simply since all its cells are empty, except for the zeroth one, which contains the $\rhd$ symbol:

$$\chi_2(0) \rightleftharpoons \psi_{2,0}((0)_2\to\rhd) \mathbin{\&} \forall\widehat{u}_0[\widehat{u}_0 \geqslant (1)_2 \to \psi_{2,0}(\widehat{u}_0\to\Lambda)].$$

It remains to record that the machine is in the start state q_0; both of its heads are pointed at zeroth cells and see the $\rhd$ symbols there. The single timer of color 0 describes all this $\pi_0((0)_2,\rhd,\rhd,(0)_2,(0)_2)$ — see Section 5.1.

The whole initial configuration is described by the formula

$$\chi(0)(\widehat{y}_0) \;\rightleftharpoons\; \chi_1(X) \;\;\&\;\; \chi_2(0) \;\;\&\;\; \pi_0((0)_2,\rhd,\rhd,(0)_2,(0)_2), \tag{9}$$

It is easy to see that the formula $\chi(0)(\widehat{y}_0)$ is the brief form for $\Psi C(t)(\widehat{y}_0)$ introduced in Subsection 5.3.3 for $t{=}0$.

We recall that r positions are reserved for recording the codes of the symbols of the working alphabet of simulated machines and the indication of the machine state's numbers in Subsection 5.1; and if $|X| = n$, then $|\widehat{X}| = \mathcal{O}(\lceil\log n\rceil\cdot\lceil\log\lceil\log n\rceil\rceil)$.

Lemma 5. *$|\chi(0)(\widehat{y}_0)| \leqslant D_1\cdot n\cdot\lceil\log n\rceil\cdot\lceil\log\lceil\log n\rceil\rceil + D_2\cdot m(n)\cdot\lceil\log m(n)\rceil$ for the suitable constants D_1, D_2 and $|X| = n > r$.*

Proof. This follows from Lemmata 1 and 2 by immediate calculation. □

7.2. The simulating formula $\Omega^{(m)}(X,P)$. Let us define

$$\Omega^{(m)}(X,P) \rightleftharpoons \forall \widehat{y}_0, \widehat{y}_T \Big\{ \Big[\chi(0)(\widehat{y}_0) \ \& \ \exists \widehat{Y}_m \forall \widehat{a}_m \forall \widehat{b}_m \ldots \exists \widehat{Y}_1 \forall \widehat{a}_1 \forall \widehat{b}_1$$
$$\Big\{ \bigwedge_{1 \leqslant k \leqslant m} \big[(\widehat{a}_{k+1} \equiv \widehat{a}_k \wedge \widehat{Y}_k \equiv \widehat{b}_k) \vee (\widehat{Y}_k \equiv \widehat{a}_k \wedge \widehat{b}_k \equiv \widehat{b}_{k+1}) \big] \to \tag{10}$$
$$\to \Phi^{(0)}(P)(\widehat{a}_1, \widehat{b}_1) \Big\} \Big] \to \quad \chi(\omega)(\widehat{y}_T) \Big\};$$

here, we have designated $\widehat{a}_{m+1} = \widehat{y}_0$, $\widehat{b}_{m+1} = \widehat{y}_T$ in the record of the "big"conjunction for the sake of brevity.

Proposition 5. *The formula $\Omega^{(m)}(X,P)$ has the property (i) from the statement of Teorema 2. In other words, this sentence is true if and only if the machine P accepts the input X within $T = \exp(m(n))$ steps.*

Proof. Let $\Theta_k = \Theta_k(\widehat{a}_k, \widehat{a}_{k+1}, \widehat{Y}_k, \widehat{b}_k, \widehat{b}_{k+1})$ be the denotation for the expression $(\widehat{a}_{k+1} \equiv \widehat{a}_k \wedge \widehat{Y}_k \equiv \widehat{b}_k) \vee (\widehat{Y}_k \equiv \widehat{a}_k \wedge \widehat{b}_k \equiv \widehat{b}_{k+1})$. The part of the formula $\Omega^{(m)}(X,P)$, which is located in the big square brackets in Equation (10), almost coincides with the first of the two following formulae and is equavalent to the second (according to the agreement of Subsection 4.2 that conjunction connects more intimately than an implication):

1) $\chi(0)(\widehat{y}_0) \ \& \ \exists \widehat{Y}_m \forall \widehat{a}_m \forall \widehat{b}_m \ldots \exists \widehat{Y}_1 \forall \widehat{a}_1 \forall \widehat{b}_1 \ \{ \bigwedge_{1 \leqslant k \leqslant m} \Theta_k \to \Phi^{(0)}(P)(\widehat{a}_1, \widehat{b}_1) \};$

2) $\chi(0)(\widehat{y}_0) \ \& \ \exists \widehat{Y}_m \forall \widehat{a}_m \forall \widehat{b}_m \ldots \exists \widehat{Y}_1 \forall \widehat{a}_1 \forall \widehat{b}_1 \Big(\Theta_m \to \big(\Theta_{m-1} \to (\ldots \to (\Theta_1 \to$
$\to \Phi^{(0)}(P)(\widehat{a}_1, \widehat{b}_1)) \ldots)) \Big).$

If we carry the quantifiers through the subformulae, which do not contain the corresponding variables, then we will conclude that the second formula is tantamount to the third one:

3) $\chi(0)(\widehat{y}_0) \ \& \ \exists \widehat{Y}_m \forall \widehat{a}_m \forall \widehat{b}_m \Big(\Theta_m \ \to \ \exists \widehat{Y}_{m-1} \forall \widehat{a}_{m-1} \forall \widehat{b}_{m-1} \big(\Theta_{m-1} \to (\ldots \to$
$\to \exists \widehat{Y}_1 \forall \widehat{a}_1 \forall \widehat{b}_1 (\Theta_1 \to \Phi^{(0)}(P)(\widehat{a}_1, \widehat{b}_1)) \ldots)) \Big).$

According to the definition, the formula

$$\exists \widehat{Y}_k \forall \widehat{a}_k \forall \widehat{b}_k (\Theta_k \to \Phi^{(k-1)}(P)(\widehat{a}_k, \widehat{b}_k))$$

contracts into $\Phi^{(k)}(P)(\widehat{a}_{k+1}, \widehat{b}_{k+1})$. Therefore, the whole sentence $\Omega^{(m)}(X,P)$ is equivalent to $\forall \widehat{y}_0, \widehat{y}_T \big[\big(\chi(0) \ \& \ \Phi^{(m)}(P) \big) \to \chi(\omega) \big]$. Notice that the formula

$$\forall \widehat{y}_0, \widehat{y}_T \big[\big(\Psi C(0)(\widehat{y}_0) \ \& \ \Phi^{(m)}(P)(\widehat{y}_0, \widehat{y}_T) \big) \to \Psi(T)(\widehat{y}_T) \big]$$

is true by Proposition 4(ii) if and only if the machine P accepts an input string X within T steps, and $\chi(\omega)$ is a conjunctive term of the formula $\Psi(T)(\widehat{y}_T)$. Consequently, based on the fact that formulae $\chi(0)(\widehat{y}_0)$ and $\Psi C(0)(\widehat{y}_0)$ are tantamount, one could say that the formula (10) is modeling. □

7.3. The time of writing of $\Omega^{(m)}(X,P)$ and its length. Let us discuss the possibility of writing this formula for the various functions $m(n)$.

Lemma 6. *Let $g(n)$ be a fully space-constructible function. Then for each input string having length n on the first tape, one can fill exactly $g(n)+1$ square on the second tape and then number the filled cells in the binary representation using $\mathcal{O}(g(n)\cdot\lceil\log g(n)\rceil)$ memory cells within time bounded the value of some polynomial on $\max\{n,g(n)\}$, provided that the working alphabet has at least five symbols: $\rhd,\Lambda,0,1$, and one more, for instance, X.*

Proof. At first, we fill exactly $g(n)+1$ squares on the second tape with the X symbols. One can make this using exactly $g(n)+1$ memory cells since the function $g(n)$ is fully space-constructible. Then, we count the filled cells beginning with the zero number and apply a reverse binary representation, e.g., the number 6 will be written down as 011, but not 110. For this purpose, we write the tuple Y consisting of 0 and 1 to the right of $(g(n)+1)$-tuple $(X,\ldots,X)$. The tuple Y is $\lceil\log(g(n)+1)\rceil$ symbols in length. One can fill these $g(n)+|Y|+1$ squares utilizing the same amount of memory. Next, we fill the zone Y with zeros. After that, we move apart the adjacent symbols X by $|Y|$ positions and fill these gaps with zeros. Possibly, we will use a few more cells than $(g(n)+1)\cdot|Y|$ for this expansion. It remains only to number the X symbols, using gaps filled with zeros for this; however, now we write the binary representation of the numbers in the usual way. If $g(n)=\lceil\log n\rceil$, then as a result, we will obtain the tuple $\widehat{X}$ involved in the record of the simulating formula — see Subsection 5.1.

All this is feasible in polynomial time on $\max\{n,g(n)\}$. □

Remark 6. *(1) It is quite probable that this lemma can be noticeably strengthened. In particular, one can most likely get by with the alphabet $\{\rhd,\Lambda,0,1\}$.*

(2) All functions, interesting for us, are fully space-constructible: $\lceil\log n\rceil$, n, cn^d, and $\exp_k(n)$. The functions $n\cdot\lceil\log n\rceil$ and $n\cdot\lceil\log n\rceil\cdot\lceil\log\lceil\log n\rceil\rceil$ are uninteresting for us concerning the property "be space-constructible" because the summand $D_1\cdot n\cdot\lceil\log n\rceil\cdot\lceil\log\lceil\log n\rceil\rceil$ arises in the upper estimate of the length of the modeling formula (see inequalities (1) and Lemma 5) because the input string is described by n clauses, each of which is $\mathcal{O}(\lceil\log n\rceil\cdot\lceil\log\lceil\log n\rceil\rceil)$ in length.

We recall that the disjunction of equivalencies $(\widehat{a}_{k+1}\equiv\widehat{a}_k\wedge\widehat{Y}_k\equiv\widehat{b}_k)\vee(\widehat{Y}_k\equiv\widehat{a}_k\wedge\widehat{b}_k\equiv\widehat{b}_{k+1})$ entered in the formula $\Omega^{(m)}(X,P)$ was denoted as Θ_k in the proof of Proposition 5; and the lengths of the formulae are calculated in the natural language — see Subsection 4.2.

Lemma 7. *The length of the subformula*

$$\Upsilon(m) \;\rightleftharpoons\; \exists\,\widehat{Y}_m\forall\,\widehat{a}_m\forall\,\widehat{b}_m\ldots\exists\,\widehat{Y}_1\forall\,\widehat{a}_1\forall\,\widehat{b}_1\;\Big\{\bigwedge_{1\leqslant k\leqslant m}\Theta_k\to\Phi^{(0)}(P)(\widehat{a}_1,\widehat{b}_1)\Big\}$$

is $\mathcal{O}((|P|\cdot m(n)+m^2(n))\cdot\lceil\log m(n)\rceil)$ for sufficiently long X and $m=m(n)\geqslant\lceil\log n\rceil$.

Proof. According to Lemmata 3 and 4(iii),

$$|\Phi^{(0)}(P)(\widehat{y}_t, \widehat{y}_{t+1})| \leqslant C_1 \cdot |P| \cdot |\varphi(N)| \leqslant C_2 \cdot |P| \cdot (m+r+s) \cdot \max\{\log m, \log r, \log s\}$$

for appropriate constants C_1 and C_2. The tuple $\widehat{y}_t$ consists of $2m+5r+2s$ various variables; therefore, its length is not more than $9m \cdot \lceil \log m \rceil$ for all X such that $|X| = n$ and $m = m(n) \geqslant \lceil \log n \rceil = s \geqslant r$. For $1 \leqslant k \leqslant m$, the tuples $\widehat{Y}_k$, $\widehat{a}_k$, and $\widehat{b}_k$ also have $2m+5r+2s$ variables; and each of these tuples has the variables of the identical type, but not of nine ones; hence, their maximal second index equals $\lceil \log(2m+5r+2s) \rceil$.

Nevertheless, $|\exists \widehat{Y}_k| = |\forall \widehat{a}_k| = |\forall \widehat{b}_k| = \mathcal{O}(m \cdot \lceil \log m \rceil)$ for all k according to our suggestions about X and m. Hence,

$$|\exists\, \widehat{Y}_m \forall\, \widehat{a}_m \forall\, \widehat{b}_m \ldots \exists\, \widehat{Y}_1 \forall\, \widehat{a}_1 \forall\, \widehat{b}_1 \{ \bigwedge_{1 \leqslant k \leqslant m} \Theta_k \}| \;=\; \mathcal{O}(m^2(n) \cdot \lceil \log m(n) \rceil).$$

□

Corollary 6. *There exist constants D_1, D_2, and D_w such that Inequalities (1) hold for every sufficiently long X and $m(n) \geqslant \lceil \log n \rceil$, i.e.,*

$$m(n) < |\Omega^{(m)}(X,P)| \leqslant D_1 \cdot n \cdot \lceil \log n \rceil \cdot \lceil \log \lceil \log n \rceil \rceil + D_2 \cdot m(n) \cdot \lceil \log m(n) \rceil +$$
$$+ D_w \cdot [(|P| \cdot m(n) + m^2(n)) \cdot \lceil \log m(n) \rceil.$$

Proof. It follows from this lemma, Lemma 5, and the description of $\chi(0)$ in Subsection 7.1 because $|\forall \widehat{y}_0| < |\forall \widehat{y}_T| = \mathcal{O}(m \cdot \lceil \log m \rceil)$. □

Thus, we have proven that the formula $\Omega^{(m)}(X,P)$ of form (10) possesses the properties (i) (see Proposition 5) and (ii) from Theorem 2, i.e., it is simulating and has a relatively short length. The modeling formula is described by Equation (10) in an explicit form; this description allows us to design a polynomial algorithm relative to $\max\{n, m(n)\}$ for its construction based on Lemma 6.

8 The space complexity of languages from the class P

A simple application of the modeling theorem to languages of class **P** (as is done in the second proof of Theorem 5 for languages from class ${}^{(k+1)}$**EXP** in Subsection 3.2) yields the uninteresting fact that these languages can be recognized in an almost linear space, more precisely, with memory $\mathcal{O}(n \cdot \lceil \log n \rceil^2 \cdot \lceil \log \lceil \log n \rceil \rceil)$; this follows from Corollary 6 for $m(n) = C \lceil \log n \rceil$. We will try to strengthen this statement.

When $m(n) = \lceil \log n^d \rceil$, then the longest component of the modeling formula $\Omega^{(m)}(X,P)$ is the subformula $\chi_1(X)$ corresponding to the unchanged entries on the first tape (see Subsection 7.1) since it contains the subformula $\bigwedge_{1 \leqslant (\widehat{\xi}) \leqslant n} \psi_1(\widehat{\xi} \to \chi(\widehat{\xi}))$ describing the input string $X = \langle \chi_1, \ldots, \chi_n \rangle$ and having length $\mathcal{O}(n \cdot \lceil \log n \rceil \cdot \lceil \log \lceil \log n \rceil \rceil)$. How this description can essentially be abridged is perfectly incomprehensible. The universal quantifier cannot be applied here, as the input string may contain various symbols.

Should we not try to remove this description entirely because we have a record of the input string on the first tape?

Let us notice that the programs of all Turing machines depend only on the working alphabet, recognized language, and the number of their tapes, but not on an input string; though, the program's execution depends on the input, as a rule, essentially, see Remark 3b).

Definition 2. *Let the formula $\Omega^{(m)}(X,P)$ is written for the simulation of the actions of 2-DTM P for some input X during the first $\exp m(n)$ steps. In that case, we denote its subformula, which is obtained if the component "$\chi_1(X)$ &"is removed from the formula $\chi(0)(\widehat{y}_0)$ (see Eq. (8) and (9)), as $\Delta^{(m)}(n,P)$, and will name it a conditionally modeling formula.*

The reduced formula $\Delta^{(m)}(n,P)$ depends only on the program P, the function m, and the length n of input according to the description of the simulating formula — see Remark 3 **b)**. Its length is

$$\mathcal{O}((|P| \cdot d \cdot \lceil \log n \rceil + d^{\,2} \cdot \lceil \log n) \rceil^2 \cdot \lceil \log \lceil \log n \rceil \rceil)$$

for $m(n) = \mathcal{O}(\lceil \log n^d \rceil)$. We note that formula $\chi(0)$ is only shortened to $\chi_2(0)$ & $\pi_0((0)_2, \triangleright, \triangleright, (0)_2, (0)_2)$ (see Eq. (9) in Subsection 7.1), but is not removed at all.

Theorem 6. *Every language of class* **P** *can be recognized in the polylogarithmical space* **Log***; more precisely, if a language $\mathcal{L}$ belongs to 2-$DTIME[n^d]$, then there is a 2-DTM $P(\mathcal{L})$ that recognizes $\mathcal{L}$ with memory bounded by $C \cdot d^{\,2} \cdot \lceil \log^2 n \rceil \cdot \lceil \log \lceil \log n \rceil \rceil)$ for the appropriate constant C.*

Proof. We recall that the working alphabet of all 2-DTMs is $\mathcal{A}$, and $\mathcal{A}_1$ is the alphabet of the natural language described in Subsection 4.2. Both these alphabets are finite. We regard that the programs of the machines are also written in the alphabet $\mathcal{B} \rightleftharpoons \mathcal{A} \cup \mathcal{A}_1$ with a natural identification of the implication and the arrow, which is involved in the record of the instructions. In Subsection 5.1, a uniform length binary coding $c = c(\mathcal{A}, P)$ of all symbols of the working alphabet $\mathcal{A}$ and simultaneously all internal states of the given machine P was applied. Now, we extend this coding to $c_1 = c_1(\mathcal{B}, P)$ to include the whole alphabet $\mathcal{A}_1$. We will continue to write down the formulae $\Omega^{(m)}(X,P)$ and $\Delta^{(m)}(n,P)$, applying the alphabet $\mathcal{A}_1$ and the encoding c_1 as described in Section 5 for the encoding c.

However, one can do without such an extension if one immediately assumes that the alphabet $\mathcal{A}$ includes the whole alphabet $\mathcal{A}_1$ since only its finiteness is used in all constructions. On the other hand, we can proceed wholly to the codes of alphabetic symbols from $\mathcal{A}_1$ using the symbols of the alphabet $\mathcal{A}$, but this will only overload the proof with irrelevant details.

Lemma 8. *For every program P, there is a 2-DTM $M(P)$ that writes P when $M(P)$ gets any input; by that, the $\triangleright$ symbol is written as $(c_1 \triangleright)$, i.e., as an r-tuple consisting of $r-1$ zeros and one 1 (see Subsection 5.1).*

Proof. Executing its instructions, the machine $M(P)$ writes down all the symbols involved in the record of P one by one in the second tape. The first head remains in the zeroth place during the entire work, so one may say that $M(P)$ is running in the *one-tape mode*.

Using an example, let us explain how $M(P)$ works. Let $q_7 \rhd, B \to q_4 R, C, L$ be some instruction in the program P. The sense of this instruction is as follows. If, in the seventh state, the machine P sees the $\rhd$ symbol on the first tape and the B symbol on the second at some instant, then P moves the first head to the right, writes C instead of B on the second tape, moves the second head to the left, and enters the state q_4.

This instruction must be written on the second tape of machine $M(P)$ in the following form (the comma belongs to the natural language's alphabet): $q7, (c_1 \rhd), B \to q4, R, C, L$. The first six instructions of $M(P)$ for writing this instruction are (here we regard that $M(P)$ has entered the state q_{101} after executing the current instruction; the ultimate non-empty symbol on the second tape is D, and the second head aims at this D):

$$\begin{array}{llll} (1) & q_{101} \rhd, D \to q_{102} S, D, R & (2) & q_{102} \rhd, \Lambda \to q_{103}\, S, q, R \\ (3) & q_{103} \rhd, \Lambda \to q_{104} S, 7, R & (4) & q_{104} \rhd, \Lambda \to q_{105} S, ,, R \\ (5) & q_{105} \rhd, \Lambda \to q_{106} S, (, R & (6) & q_{106} \rhd, \Lambda \to q_{107} S, 0, R \end{array}$$

The result of the actions of these six instructions is the string $q7, (0$. □

Lemma 9. *For every program P and each space-constructible function m, there is a 2-DTM $M_1(P)$ that writes the conditionally modeling formula $\Delta^{(m)}(n, P)$ when $M_1(P)$ gets any input X having length n. For this purpose, $M_1(P)$ uses $\mathcal{O}(|\Delta^{(m)}(n, P)|)$, i.e., $\mathcal{O}((|P| \cdot m(n) + m^2(n)) \cdot \lceil \log m(n) \rceil)$, the memory cells for the sufficiently long X.*

Proof. At first, the machine $M_1(P)$ activates its subprogram $M(P)$, which writes down the program P. Then, $M_1(P)$ calculates the value $m(n) = m|X|)$ and writes it in the binary representation on the second tape to the right of the string P. Next, it writes the formula $\Delta^{(m)}(n, P)$ using the strings P and $(m(n))_2$, i.e., the binary representation of the number $m(n)$, following Equation (10). The machine $M(P)$ again runs in the one-tape mode in this and the last stage. In the last stage, $M_1(P)$ erases the strings P and $(m(n))_2$ and shifts left the formula $\Delta^{(m)}(n, P)$. □

Let us return to the proof of the theorem.

In Sections 4-7, it has been implicitly assumed the presence of 2-DTM, which writes the code of any symbol of the working alphabet on the second tape when the first head is aimed at this symbol. This was not stated due to the obviousness. It is not difficult to add several instructions to this program that check whether the clause of a kind $\psi_1(\widehat{\zeta} \to \alpha) = \widehat{X} \equiv \widehat{\zeta} \to \widehat{F} \equiv c\widehat{\alpha}$ is true, i.e., whether the α symbol is located in the $(\widehat{\zeta})$th cell of the first tape. The actions of this subroutine $M_2(\widehat{\zeta}, \alpha)$ can be such: it shifts the head to the $(\widehat{\zeta})$th position on the first tape, reads some symbol β there, writes the code $c\beta$ on the second tape, and then compares it with the $c\alpha$ code. Similarly, $M_2(\zeta_1, \alpha_1)$ can check the truth of the fragments of the timers $\pi_t((i)_2, \alpha_1, \alpha_2, \zeta_1, \zeta_2)$ of any color t.

The last procedure we need is $M_3(\phi)$, which determines whether a given Boolean sentence ϕ is true. This machine uses a complete enumeration (or almost complete when using, for example, the branch and bound method) of possible values of the variables included in the record ϕ. After generating the next set of variable values, it substitutes these values into ϕ and checks the truth of all subformulas from ϕ. This algorithm requires exponential time but is feasible in a linearly bounded (relative to the length of ϕ) space.

Thus, suppose some 2-DTM P_0 recognizes a language $\mathcal{L}$ within time n^d. One can regard the number d as an integer. We will describe the actions of the machine $\mathcal{M}(\mathcal{L})$, which recognizes $\mathcal{L}$ in space $\mathcal{O}(\lceil\log^2 n)\rceil \cdot \lceil\log\lceil\log n\rceil\rceil)$.

Let the machine $\mathcal{M}(\mathcal{L})$ get an input X. In the first stage, it turns on subroutine $M_1(P_0)$, which writes the conditionally modeling formula $\Delta^{(m)}(n, P_0)$ for $m(n) = d \cdot \lceil\log n\rceil$. Notice that the formulae $\Delta^{(m)}(n, P_0)$ and $\Omega^{(m)}(X, P_0)$ contain the same variables. Grounding on this, the recognizing machine $\mathcal{M}(\mathcal{L})$ activates the modified procedure $M_3(\Omega^{(m)}(X, P_0))$ to check the validness of the sentence $\Omega^{(m)}(X, P_0)$. This modification lies in that the check of the truth of the triggers $\pi_t((i)_2, \alpha, \alpha_2, \zeta, \zeta_2)$ and clauses $\psi_1(\widehat{\zeta} \to \alpha)$ is carried out by applying subroutine $M_2(\widehat{\zeta}, \alpha)$. If the sentence $\Omega^{(m)}(X, P_0)$ occurs be true, then this means that the machine P_0 accepts the input X by Theorem 2; hence, the string X belongs to $\mathcal{L}$; therefore $\mathcal{M}(\mathcal{L})$ also accepts X. If not, $\mathcal{M}(\mathcal{L})$ rejects X since the word X does not belong to $\mathcal{L}$. $\square$

We recall that in Subsection 3.2, the designations for some complexity classes have been defined: $^{\mathbf{0}}\mathbf{EXP} = \mathbf{P}$, $^{\mathbf{0}}\mathbf{EXPSPACE} = \mathbf{PSPACE}$, and $^{\mathbf{-1}}\mathbf{EXPSPACE} = \mathbf{Log}$. So, based on Theorems 5 and 6, we obtained the following statement.

Theorem 7. *For every integer $k \geqslant -1$, the complexity classes $^{\mathbf{k}}\mathbf{EXPSPACE}$ and $^{\mathbf{k+1}}\mathbf{EXP}$ coincide.*

9 Relativization and non-deterministic computations

9.1. Motivation. The basic motivation for writing this work was the author's desire to significantly improve the lower estimate of the computational complexity of equivalency-nontrivial decidable theories from [7]. For this purpose, the possibility of reducing the formula, which simulates lengthy computations, was analyzed. Three fragments were the longest for the modeling period $\exp n$, i.e., when $m(n) = n$.

The first long component corresponds to the representation of the number $t \pm 1$ in binary notation. If the binary representation $(t)_2$ of the number t has length $n\lceil\log n\rceil$, then in [7], the binary representations of the numbers

$t \pm 1$ have length $\mathcal{O}((n\lceil\log n\rceil)^2)$ (Lemma 3.1 in Subsection 3.3). Introducing in corrective additives, it was possible to significantly reduce the length of recording these numbers Subsection 4.2 (compare Lemmata 3.1 in [7] and 1 in this paper).

The second long fragment in [7], the formula $\chi(0)$, describes the initial configuration, including the input string X. It was $\mathcal{O}(n \cdot n\lceil\log n\rceil)$ symbols in length. Now, it has the length $\mathcal{O}(n \cdot \lceil\log n\rceil \cdot \lceil\log\lceil\log n\rceil\rceil)$ through the usage of an input tape on which nothing can be written or erased. This is mainly why the author began to describe the modeling of the actions of two-tape machines in this paper. He has only seen the possibility of proving Theorem 6 in the writing process of the paper.

Nevertheless, the other long component, the $\Upsilon(m)$ formula (see Lemma 7), remains here of the same length. Apparently, it is possible to significantly reduce the size of the description of the exponential (relative to $m(n)$) number of steps only by abandoning the usage of the elegant construction of Stockmeyer and Meyer, but that has not yet been possible to do accurately enough.

9.2. On the possibility of relativization. Relativization of the main theorem on modeling (Theorem 2) is impossible. Indeed, in an exponential number of steps, the machine can record on the oracle's tape a string, whose length is not bounded by the value of the polynomial. Certainly, the recording process can be described by formulae similar to $\Phi^{(0)}(\widehat{y}_t, \widehat{y}_{t+1})$ (see Subsections 5.2 and 5.3). However, to describe the query to the oracle, it will be necessary to use a formula of exponential length since the query chain may be heterogeneous in structure, and as a result, the universal quantifier is not applicable here. In addition, this query string will only be known to us at the instant of the query, in contrast to the input X and the program of the machine P, which are given to us immediately, while it is desirable for us to write the modeling formula before starting the computations.

9.3. Impossibility of the definability of non-deterministic computations. Recall that Lemma 4(i) asserts *if* $\widehat{x}_t \neq \widehat{\mu}$, *then a clause* $\psi_{2,t}(\widehat{\mu} \to \varepsilon)$ *will be true independently of the values of variables* $\widehat{f}_t$. The reason for this is that this formula has the form of an implication (see Subsection 5.1). This means that, for any assignment of variables included in the modeling formulae, only one clause of the form $\psi_{2,t}(\widehat{\mu} \to \varepsilon)$ (the formula describing the contents of cell number $(\widehat{\mu})$ on the working tape at time t) can become false. Thus, this modeling method is *locally pointwise*. In contrast, the description method used in [3] and [10] is also pointwise, but there, any number of subformulas describing entries in specific cells (including all of them) can become false. Therefore, that technique can be called a *global pointwise* method.

On the one hand, using a locally pointwise method for describing individual cells allows for the application of the universal quantifier, thereby modeling a computational domain of exponential width, as described in Subsections 5.2 and 5.3. On the other hand, this is also the main obstacle to simulating nondeterministic computations. Specifically, in this case, it is impossible to adequately model even a single step of a nondeterministic machine. More precisely, it is impossible to prove an analogue of Proposition 2(ii).

Let us explain this in more detail. Recall that the difference between non-deterministic and deterministic Turing machines is as follows. A deterministic machine can have at most one instruction with a fixed beginning of the form $q_i\alpha \to \ldots$ for all possible internal states q_i and any symbols α the machine head is pointing at. In contrast, a non-deterministic machine can have several instructions with the same beginning of this form. We will call an instruction $q_i\alpha \to \ldots$ of the machine P *ordinary* if there are no other instructions with such a beginning in the program of the machine P. A series of instructions with the same beginning will be considered one *branching* (or *alternative*) instruction.

Suppose we have learned to describe the actions of alternative instructions of non-deterministic machines at any step by formulae, while preserving the locally pointwise method of encoding the record in individual cells. We note that this can be done by slightly changing the formulae $\varphi(k)$ from Section 5.2, which describe the action of the kth instruction. However, the specific way this is done is of no importance. Let the formula $\Phi_1^{(0)}(P)$ describe the application of the non-deterministic machine P in one step; again, it is completely unimportant how this formula is composed of $\varphi(k)$.

Nevertheless, even under these assumptions, we will not be able to prove the necessary full analogue of Proposition 2(ii):

Assertion 1. *Let us assume that all the configurations, which can arise after t steps of the non-deterministic machine P, are $K_1(t), K_2(t), \ldots, K_l(t)$. Suppose the formula*

$$\Omega_1^{(0)}(P) \rightleftharpoons \{[\bigvee_{1\leqslant j\leqslant l} \Psi K_j(t)] \ \&\ \Phi_1^{(0)}(P)\} \to [\bigvee_{1\leqslant i\leqslant s} \Psi K_i(t+1)]$$

is identically true. In this case, each configuration $K_1(t+1), K_2(t+1), \ldots, K_s(t+1)$ can be obtained in the $(t+1)$th step of the machine P only from some configurations $K_1(t), K_2(t), \ldots, K_l(l)$.

Naturally, $s = l = 1$ for deterministic machines; this is Proposition 2(ii). However, the converse statement, the analogue of Proposition 3, can probably still be proven, but this is inessential.

Remark 7. *This statement is formulated for the case of single-tape machines, since the entries on the first tape cannot be changed. In the following counterexample, we also consider only one working tape. Thus, all configurations of K_j and L_i should be treated as configurations of the second tape.*

To see that Assertion 1 is unprovable, consider the following

Counterexample. Let a non-deterministic machine P can create some configurations $K_1(t)$ and $K_2(t)$ after step t. At step $t+1$, some ordinary instruction converts these configurations into $K_1(t{+}1)$ and $K_2(t{+}1)$, respectively.

Suppose that configurations $K_1(t+1)$ and $K_2(t+1)$ differ only in the entries in two cells: $K_1(t+1)$ has the symbol γ in the square $(\widehat{\zeta})_2$, while $K_2(t+1)$ has the symbol δ in this cell; and in the cell $(\widehat{\theta})_2$, it is the other way around: $K_1(t+1)$ has the δ symbol, while $K_2(t+1)$ has the symbol γ in these squares. The head position in both configurations is the same, namely, the cell $(\widehat{\eta})_2 \neq (\widehat{\zeta})_2, (\widehat{\theta})_2$, the internal state of both is j, and the symbol being observed is κ. Therefore the description of the configurations $K_1(t+1)$ and $K_2(t+1)$ contain the same π-formulae and clauses $\psi_{2,t+1}(\widehat{\mu} \to \varepsilon(\mu))$ for all $\widehat{\mu} \neq \widehat{\zeta}, \widehat{\theta}$.

Now consider the configurations $L_1(t+1)$ and $L_2(t+1)$, which have the same head position as $K_1(t+1)$ and $K_2(t+1)$, the symbol it is observing, the internal state, and all records on the tape except for the cells $(\widehat{\zeta})_2$ and $(\widehat{\theta})_2$: $L_1(t+1)$ has the symbol γ written in both cells, while $L_2(t+1)$ has the symbol δ in these squares.

Above, the following simple property of clauses was noted: when $\widehat{x}_{t+1} \neq \widehat{\mu}$, the clause $\psi_{2,t+1}(\widehat{\mu} \to \varepsilon)$ is true regardless of the value of the variables $\widehat{f}_{t+1}$. This means that for any values of the variables $\widehat{x}_{t+1}$ and $\widehat{f}_{t+1}$, only one clause can be false in one of the formulae $\Psi K_1(t+1)$, $\Psi K_2(t+1)$, $\Psi L_1(t+1)$, and $\Psi L_2(t+1)$, while the rest automatically become true.

From this and based on the fact that all these four formulae $\Psi K_1(t+1)$, $\Psi K_2(t+1)$, $\Psi L_1(t+1)$, and $\Psi L_2(t+1)$ include the same timer (i.e., formula, which describes the positions of the heads, the symbols it views, and the internal state), it is easy to understand that for any values of the variables, the disjunctions $\Psi K_1(t+1) \vee \Psi K_2(t+1)$ and $\Psi L_1(t+1) \vee \Psi L_2(t+1)$ will become true or false simultaneously. Thus, the assumption that the formula

$$\bigl\{\bigl[\Psi K_1(t) \ \vee \ \Psi K_2(t)\bigr] \ \& \ \Phi^{(0)}(P)\bigr\} \ \to \ \bigl[\Psi K_1(t{+}1) \ \vee \ \Psi K_2(t{+}1)\bigr]$$

is identically true causes the identically truth of the formula

$$\bigl\{\bigl[\Psi K_1(t) \ \vee \ \Psi K_2(t)\bigr] \ \& \ \Phi^{(0)}(P)\bigr\} \ \to \ \bigl[\Psi L_1(t{+}1) \ \vee \ \Psi L_2(t{+}1)\bigr].$$

Consequently, if we want to prove Assertion 1 for this case, we need both clauses with the beginning $\widehat{x}_t \equiv \widehat{\zeta} \to \ldots$ to become false or true independently of the truth values of the quasi-equations with the beginning $\widehat{x}_t \equiv \widehat{\theta} \to \ldots$. Therefore, the description of the cell number $(\widehat{\zeta})_2$ must include at least one variable that is not in the description of the cell number $(\widehat{\theta})_2$. By symmetry, we also obtain the opposite, that the encoding of the cell number $(\widehat{\theta})_2$ must include at least one variable that is not in the description of the cell number $(\widehat{\zeta})_2$. And since we have not made any specific assumptions about the values of $(\widehat{\zeta})_2$, $(\widehat{\theta})_2$, and $(\widehat{\eta})_2$, other than the fact that they are different, it turns out that the description of each cell number must contain a variable that is not present in the encoding of any other cell.

So, the quantity of variables must be no less than the width of the simulated computation zone, when we want to model actions of a non-deterministic machine.

References

[1] [4] A.V. Aho, J.E. Hopcroft, J.D. Ullman, *The design and analysis of computer algorithms*, Addison-Wesley Publishing Company, Reading, Massachusetts, 1976.

[2] S. Arora, B. Barak, *Computational Complexity: A Modern Approach*, Published by Cambridge University Press, 2009.

[3] S.A. Cook, *The complexity of theorem-proving procedures*, in: Proc. of the 3rd Annual ACM Symposium on the Theory of Computing, Shaker Heights, Ohio, (1971), 151–159.

[4] D. Du, K. Ko, *Theory of computational complexity. Second edition.* Published by John Wiley&Sons, Inc., Hoboken, New Jersey, 2014.

[5] Y.L. Ershov, I.A. Lavrov, A.D. Taimanov, M.A. Taitslin, *Elementary theories*, in: Russian Math. Surveys, **20** (1965), 35–100 (English translation).

[6] M.R. Garey, D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, New York, 1979.

[7] I.V. Latkin, *The recognition complexity of decidable theories.* Eurasian Math. J., (2022), **13**:1, 44–68.

[8] M.O. Rabin, *Decidable theories*, in: Handbook of mathematical logic, Part C., Ch III, Ed. J. Barwise, Elsevier, Amsterdam, New York, Eighth impression, 1999, 596–629.

[9] L.J. Stockmeyer, *The complexity of decision problems in automata theory and logic*, PhD thesis, MIT Lab for Computer Science, 1974; also /MIT/LCS TechRep 133.

[10] L.J. Stockmeyer, A.R. Meyer, *Word problems requiring exponential time*, in Proc. 5th Ann. ACM Symp. on the Theory of Computing, Association for Computing Machinery, New York, 1973, 1–9.

[11] S. Vorobyov, *The most nonelementary theory*, Information and Computation, **190** (2004), 196–219.

Ivan Vasil'evich Latkin
East Kazakhstan technical university,
D. Serikbayev str., 19,
070004, Ust-Kamenogors, Kazakhstan
E-mail address: lativan@yandex.kz

[4]We quote from this book and [6] and from review [5] by their versions in Russian.