The complexity of the first-order theory of pure equality*

Ivan V. Latkin†

Abstract

We will find a lower bound on the recognition complexity of the theories that are nontrivial relative to equality (or equational-nontrivial), namely, each of these theories is consistent with the formula, whose sense is that there exist two various elements at least. However, at first, we will obtain a lower bound on the computational complexity for the first-order theory of Boolean algebra that has only two elements. For this purpose, we will code the long-continued deterministic Turing machine computations by the relatively short-length quantified Boolean formulae; the modified Stockmeyer and Meyer method will appreciably be used for this simulation. Then, we will construct a polynomial reduction of the theory of this Boolean algebra to the first-order theory of the pure equality.

Key words: Computational complexity, the theory of equality, the coding of computations, simulation by means formulae, polynomial time, polynomial space, lower complexity bound

1 Introduction

At the beginning, we recall some designations. A function \(\exp_k(n)\) is called \(k\)-iterated or \(k\)-story exponential, if, for every natural \(k\), it is calculated in the following way: \(\exp_0(n) = 2^n\), \(\exp_{k+1}(n) = 2^{\exp_k(n)}\). The length of a word \(X\) is denoted by \(|X|\), i.e., \(|X|\) is the number of symbols in \(X\). If \(A\) is a set, then \(|A|\) denotes its cardinality; ”\(A \equiv A\)” means ”\(A\) is a designation for \(A\)”; and \(\exp(n) \equiv \exp_1(n)\).

*Mathematics Subject Classification. 03C40, 03C07, 03D15.
†lativan@yandex.ru
1.1 Problem statement

The results on the complexity of recognition (or computational complexity) for many of the decidable theories are well-known \[7, 12, 13, 14, 15, 16, 17\]. We recall only some of these results concerning first-order theories.

Any decision procedure has more than an exponential complexity for the theory $\text{ThRLC}$ of the field $\mathbb{R}$ of real numbers, and even for $\text{Th}(\mathbb{R}, +)$ \[7\], namely, there exists a rational constant $d_1 > 0$, such that if $P$ is a deterministic Turing machine which recognizes the theory $\text{ThRLC}$ (or $\text{Th}(\mathbb{R}, +)$), then the $P$ runs for at least $2^{d_1|\varphi|}$ steps when started on input $\varphi$, for infinitely many sentences $\varphi$; and so, the complexity of recognition for these theories (which corresponds to the concept of inner complexity as defined in \[15\]) is more than $\exp(d_1n)$, here and below, the variable $n$ is the length of the input string; and the letter $d$ with subscripts denotes a suitable constant. In other words, $\text{ThRLC}$ and $\text{Th}(\mathbb{R}, +)$ do not belong to $\text{DTIME}(\exp(d_1n))$.

For Presburger arithmetic $\text{PAR}$ (the theory of natural numbers with addition) and for Skolem arithmetic $\text{SAR}$ (the theory of natural numbers with multiplication), the recognition complexity is more than a double exponential: $\text{PAR}, \text{SAR} \notin \text{DTIME}(\exp_2(d_2n))$. For the theory of linearly ordered sets $\text{ThOR}$, the computational complexity is very great \[13\]: $\text{ThOR} \notin \text{DTIME}(\exp_{[d_3n]}(n))$, where $\lfloor y \rfloor$ is the integer part of a number $y$.

It is quite natural to expect that if we go beyond the confines of logical theories of the first order, then we can see more impressive lower bounds on the recognition complexity of theories. An example of such an estimate is the lower bound for the weak monadic second-order theory of one successor $\text{WSIS}$, other examples can be found in \[4, 12, 15, 17\]. However, according to the author, the most impressive estimate of this kind was obtained by Vorobyev S.G. \[19\] for the type theory $\Omega$, which is a rudimentary fragment of the theory of propositional types due to Henkin: $\Omega \notin \text{DSPACE}(\exp_\infty(\exp(d_4n)))$, hence $\Omega \notin \text{DTIME}(\exp_\infty(\exp(d_4n)))$, where the function $\exp_\infty$ is recursively defined by $\exp_\infty(0) = 1$ and $\exp_\infty(k + 1) = 2^{\exp_\infty(k)}$, i.e., this lower bound has the exponentially growing stack of twos.

And what is the recognition complexity of the simplest (in the semantic and syntactical sense), but non-trivial theories? Should it be polynomial? In other words, shall such theories be quickly decidable?

It is clear that one of the simplest theory is the first-order theory of the algebraic structure of two elements with a unique equality predicate. We will see in Section \[7\] that even this theory does not have a polynomial upper bound.
of computational complexity. We will in passing obtain the lower bounds on the recognition complexity of the theories that are nontrivial relatively to some equivalence relation $\sim$, namely, these theories have models with at least two elements that are not $\sim$-equivalent. Obvious examples of such theories are the theories of pure equality and of one equivalence relation.

Since the lower bound on the computational complexity of these theories is not polynomial, we obtain that the class $P$ is a proper subclass of $PSPACE$.

1.2 Used methods and the main idea

The lower bounds on the computational complexity for the theories mentioned in previous subsection and some others were yielded by the techniques of the efficient reducibility of the machines to the formulae in [7, 12, 13, 14, 15, 16, 17, 19], or more precisely, by methods of the immediate codings of the machine actions. The essence of these methods is as follows [15]. Let $T$ be the theory, under study, written in the signature (or underlying language [15]) $\sigma$. Assume that, for any input string $X$ and every program $P$ of the Turing machine, one can write a sentence $S(P,X)$, of the $\sigma$, satisfying the following conditions. There exist a constant $d > 0$ and a function $f$ such that: (i) $|S(P,X)| < d(|X| + |P|)$; (ii) $S(P,X) \in T$ if and only if a computation by the program $P$ accepts the input $X$ in fewer than $f(|X|)$ steps; (iii) the formula $S(P,X)$ can be effectively constructed from $X$ and $P$ in fewer than $g(|X| + |P|)$ steps, where $g(k)$ is a fixed polynomial. If $f(k)$ is a function growing at least at exponential rate, then under the above conditions, there exist a constant $C > 0$ and infinitely many sentences $\varphi$ of $\sigma$, for which every Turing machine requires at least $f(C|\varphi|)$ steps to decide whether $\varphi \in T$, i.e., $T \notin DTIME(f(Cn))$.

The proof of the last statement is based on a well-known diagonal argument, though we will below scrutinize this method in more detail and in a somewhat more general form than this was done in the previous paragraph or in Subsection 4.1 in [15]. We need the more general form of this technique for the following reason.

Our main purpose is to evaluate the computational complexity of an equality theory $ThE$ (Section 7). However, at first, we will obtain a lower

---

1We will call it the Rabin and Fischer method or the technique for modeling of computations by means formulae.
bound on recognition complexity for the first-order theory of Boolean algebra $\mathcal{B}$ that has only two elements, using the Rabin and Fischer method. Then, we will construct a polynomial reduction of the $\text{Th}\mathcal{B}$ to $\text{Th}\mathcal{E}$. In Subsection 8.1 we will explain why such a succession of actions is applied.

But the first-order theory of two-element Boolean algebra has a very weak expressive ability. Therefore, the modeling sentence for this theory, i.e., the formula possessing property (ii) from the method described above, does not turn out to be very short, it may have not a linear restriction on its length (see Subsection 6.4 for more details). Furthermore, the $\text{Th}\mathcal{B}$ is so poor and meager that there can, in general, be a doubt about the very possibility of the simulation of the sufficiently long computations by means of the relatively short formulae of this theory.

Nevertheless, such a modeling was well-known a long time ago. Stockmeyer L.J. and Meyer A.R. showed in 1973 that a language $\text{TQBF}_2$ consisting of the true quantified Boolean formulae is polynomially complete in the class $\text{PSPACE}$ [9, 18]. This implies in particular that for every language $\mathcal{L}$ in this class, there is an algorithm, which produces a quantified Boolean formula for any input string in polynomial time; and all these sentences model the computations that recognize the $\mathcal{L}$ and use the polynomial amount of space. Namely, each of theirs is true if and only if the given input string belongs to the language under study; at that, the long enough computations are simulated, seeing that the polynomial constraint on memory allows the machine to run during the exponential-long time [11, 2, 9].

Stockmeyer L.J. and Meyer A.R. have employed the highly ingenious technique for the implementation of this simulation (see the proof of Theorem 4.3 in [18]). Their approach permits writing down 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, the length of which is polynomially bounded. One running step of machine is described in [18] by the Cook’s method formula; this is a formula of the propositional calculus, and one can construct it just as the sentences, which were applied for the modeling of the polynomial quantity of steps of the nondeterministic Turing machine in the proof of $\text{NP}$-completeness of the problem $\text{SAT}$ [5], (see also the proof of Theorem 10.3 in [1]). There exists a Boolean $\exists$-formula, which corresponds to the Cook’s method formula. We will also name this $\exists$-formula.

\footnote{The problem corresponding to this language is designated as $\text{QBF}$, or sometimes $\text{QSAT}$.}
as Cook’s formula.

We intend to modernize the elegant construct of Stockmeyer and Meyer and to bring it into play for the obtaining our purpose. But we will model the running steps of a machine by the more complicated formulae that have an alternation of quantifiers. This complication is caused due to the fact that the Cook’s method formula is very long — it is far longer than an amount of the used memory. Really, it has a subformula that consists of one propositional variable \( C_{i,j,t} \) (see, for instance, the proof of Theorem 10.3 in [1] and also Section 8). This variable is true if the \( i \)th cell contains symbol \( X_j \) of the tape alphabet at the instant of time \( t \). But suppose that each of the first \( T + 1 \) squares of tape contains the symbol \( X_0 \) at time \( t \), the remaining part of the tape is empty. This simple tape configuration (or instantaneous description [1,18]) is described by the formula that has a fragment \( C_{0,0,t} \land C_{1,0,t} \land \ldots \land C_{T,0,t} \), and this subformula is \( 2T + 1 \) in length without taking the indices into account. It is impossible to abridge this record, even if we try to use the universal quantifier since its application to the indices is not allowed within the confines of the first-order theory. Thus, in order to describe the machine actions using the exponential amount of space, we need Cook’s formula, whose length is no less than exponential.

We propose to encode the binary notation of the cell number by a value set of special variables \( x_{t,0}, \ldots, x_{t,n} \), where \( n+1 \geq \log_2 T \) (see Subsections 4.1, 6.2, and 8.2 for further details). So we need \( \mathcal{O}(n) \) symbols for the describing of one cell, and \( \mathcal{O}(n^2) \) ones for the assignment of the whole input string \( X \), if \( n = |X| \). But then we can describe one running step of the machine, which uses \( T \equiv \exp(|X|) \) memory cells on input \( X \), with the aid a formula that is \( \mathcal{O}(n^3) \) in length. The main idea of so brief a describing consists of the following: merely one tape square can change on each of the running steps, although the whole computation can use the exponential amount of memory.

Therefore, it is enough for us to describe the changes in the only cell, and the contents of the remaining ones can be ”copied” by applying the universal quantifier (see the construction of the formula \( \Delta^{cop}(\tilde{u}) \) in Subsection 4.1).

At the beginning, we will introduce all variables in great abundance in order to facilitate the proof, namely, the variables will have the first indices \( t \) from 0 to \( T \). Next, we will eliminate many of the variables using the

\footnote{The denoted locality of the actions of deterministic machines has long been used in the modeling of the machine computation with the help of formulae, see, for example, Lemma 2.14 in [17] or Lemma 7 in [19].}
modified method of Stockmeyer and Meyer — see Subsections 4.3.2, 6.2, 8.2 for further details. A final modeling formula will only contain those of variables for which \(0 \leq t \leq n\) or \(t = T\) hold.

The description of the initial configuration and the condition of the successful termination of computations have a length of \(O(n^3)\), if we anew use the quantifiers; hence the entire formula, which simulates the first \(\exp(n)\) steps of the computation of the machine \(P\), will be \(O(|P| \cdot n^3)\) in length (taking into account the indices).

Therefore, we need to slightly strengthen the Rabin and Fischer method, so that it can also be applied in the case of a non-linear estimate for the length of the modeling formula.

1.3 The paper structure

The generalized Fisher and Rabin method is adduced in Section 2. The degree of its usefulness and novelty is discussed in Remark 1. Section 3 contains an exact formulation of the main theorem (Theorem 1), its primary corollaries, and some preparation for that and for the proof of this theorem. Sections 4–6 are devoted to the proof of the main theorem. The lower bound on the computational complexity of the theories, which are nontrivial relatively to some equivalence relation, in particular, equational-nontrivial, will be yielded in Section 7. In Section 8, we will discuss the obtained results and consider the used methods in greater detail, comparing theirs with other approaches to the simulation of computation.

2 The generalized Fischer and Rabin method

We will describe this method in the most general form.

2.1 Auxiliary notions

We will need some new concepts.

**Definition 1.** Let \(P\) be a program of the Turing machine; \(k\) be a number of its tapes; and \(q_b\alpha_1\alpha_2\ldots\alpha_k \rightarrow q_j\beta_1\beta_2\ldots\beta_k\) be an instruction of this program. We will call this instruction explicitly non-executable and the internal state \(b\) inaccessible (for the \(P\)), if the program \(P\) does not contain the instructions of the form \(q_r\gamma_1\gamma_2\ldots\gamma_k \rightarrow q_b\delta_1\delta_2\ldots\delta_k\).
It is clear that one can write such machine program that it contains some non-executable instructions, but all its internal states are accessible. It is evident too that one can easily find the explicitly non-executable instructions in any program, most precisely, all such instructions can be found in polynomial time on the program length. However, the detection of the non-executable instructions, whose internal states are accessible, maybe is the very difficult task in some cases.

Let us assume that we have removed all the explicitly non-executable instructions from a program \( P \). The elimination has resulted in some program \( P_1 \). This \( P_1 \) may again contain some explicitly non-executable instructions, for instance, if the instructions \( q_1 \gamma_1 \gamma_2 \ldots \gamma_k \rightarrow q_0 \delta_1 \delta_2 \ldots \delta_k \) and \( q_0 \alpha_1 \alpha_2 \ldots \alpha_k \rightarrow q_j \beta_1 \beta_2 \ldots \beta_k \) belong to \( P \), the first of them is explicitly non-executable for the \( P \), and the state \( b \) is not included in other instructions, then the second instruction is not such in full, although it is non-executable for the \( P \). However, it already is explicitly non-executable for the program \( P_1 \). We can continue this removing process of the explicitly non-executable instructions until we obtain the irreducible program \( r(P) \) that does not contain such instructions.

We name the programs \( T \) and \( P \) monoclonal if \( r(T) = r(P) \); at that the \( P \), \( T \), and \( r(P) \) are called the clones of each other. As usual, a Turing machine and its program are designated by a uniform sign. Therefore we will say that two Turing machines are monoclonal if their programs are so.

**Lemma 1.** (i) There exists a polynomial \( h(n) \) such that one can write the code of irreducible clone \( r(P) \) within \( h(|P|) \) steps for every program \( P \);

(ii) the question about the monoclonality of any two programs \( P \) and \( T \) is solvable in polynomial time from \( |P| \) and \( |T| \);

(iii) all the tape actions of monoclonal machines are identical with each other on the same inputs.

**Proof.** It straightforwardly follows from definitions. \( \square \)

**Definition 2.** Let \( F(n) \) be a function that is monotone increasing on all sufficiently large \( n \). The function \( F \) is called a limit upper bound for the class of all polynomials (LUBP) if, for any polynomial \( p \), there is a number \( n \) such that the inequality \( F(m) > p(m) \) holds for \( m \geq n \), 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 \( s \geq 1 \). It is easy to see that if \( F(x) \) is a LUBP,
then the functions $F(x^m)$ and $F(rx)$ are LUBPs also for positive constants $m$ and $r$, moreover, the function $F(x) - F(dx)$ is a LUBP for every constant $d$ such that $0 < d < 1$. It follows from this that if $T(n)$ is a LUBP, then it grows at least exponentially in the sense that is considered in [4], namely, $T(dn)/T(n)$ tends to 0 as $n$ tends to $\infty$. Inverse assertion seemingly is valid too.

2.2 The generalization

Let us suppose that we want to find a lower bound on the recognition complexity of a language $L$ over alphabet $\sigma$. We, first of all, fix a finite tape alphabet $A$ of Turing machines and the number $k$ of their tapes. We also fix a certain polynomial encoding of the strings over the alphabet $\sigma$ and of the programs of Turing machines by finite strings of symbols (words) over the alphabet $A$, i.e., it is implied that the encoding and unique decoding are realized in a polynomial time from the length of an object in a natural language. We presume also that the used encoding is composite, namely, the code of each instruction in any program is the constituent of the program code. The code of an object $E$ is denoted by $c_O E$, i.e., $c_O E \in A^*$, if $E \in \sigma^*$ or $E$ is a program.

**Proposition 1.** Let $F$ be a limit upper bound for all polynomials and $L$ be a language over some alphabet $\sigma$. Suppose that for any given program $P$ of a deterministic Turing machine and every string $X$ on the input tape of this machine, one can effectively construct a word $S(P,X)$ over the alphabet $\sigma$ with the following properties:

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

(ii) the word $S(P,X)$ belongs to $L$ if and only if the Turing machine $P$ accepts input $X$ within $F(|X|)$ steps;

(iii) there exist constants $D, b, s > 0$ such that either the inequalities

\[ (a) \quad |X| \leq |c_O S(P, X)| \leq D \cdot |c_O P|^b \cdot |X|^s \]

This language consists of all words over the alphabet $\sigma$ and all the Turing machines programs with $k$ the tapes and the tape alphabet $A$. An example of such natural language will be described in Subsection 3.1. It is implied here and below that the numbers of the internal states and other indices are written in decimal notation.
or the inequalities

\[(b) \quad |X| \leq |c_\text{O}S(P, X)| \leq D \cdot (|c_\text{O}P| + |X|)\]

hold true for all sufficiently long \(X\), and these constants do not depend on \(P\), but they depend on the applied encoding.

Then

1. for every constant \(\delta > 0\) and any program \(P\), there is a number \(t_0\) such that the inequality \(|c_\text{O}S(P, X)| \leq D_1 \cdot |X|^{s_1}\) holds true for all of the strings \(X\), which are longer than \(t_0\), where \(D_1 = D + \delta\) and \(s_1 = s + \delta\) in case (a) or \(D_1 = (D + \delta)\) and \(s_1 = 1\) in case (b);
2. for each \(a > 1\) and every deterministic Turing machine \(M\), which recognizes the language \(\mathcal{L}\), there exist infinitely many words \(Y\), on which \(M\) runs for more than \(F(D_2 \cdot |c_\text{O}Y|^{\rho})\) steps for \(D_2 = (aD_1)^{-\rho}\) and \(\rho = (s_1)^{-1}\).

Proof. (1) It is easy to see that the \(t_0\) is equal to \(|c_\text{O}P|^{b/\delta}\) in case (a); and it equals to \((D/\delta) \cdot |c_\text{O}P|\) in case (b).

(2) In accordance with condition (i), one can assume that a code for \(S(P, X)\) can be written by some machine \(M_1\) for all given strings \(X\) and \(c_\text{O}P\).

Let us suppose that there exist numbers \(a, t_1\) and a machine \(M_2\) such that the \(M_2\) determines whether \(Y \in \mathcal{L}\) within \(F(D_2 \cdot |c_\text{O}Y|^{\rho})\) steps for any string \(Y\) over the \(\sigma\), provided that \(|c_\text{O}Y| > t_1\) and \(a > 1\).

To proceed to an ordinary diagonal argument, we stage-by-stage construct the Turing machine \(M\). At the first stage, we write a machine \(M_0\), which for a given input \(X\), determines whether the string \(X\) is the code \(c_\text{O}P\) of some program \(P\). If not, then the \(M_0\), as well as the whole machine \(M\), rejects the \(X\); else it writes the code \(c_\text{O}r(P)\) of the irreducible clone \(r(P)\).

At the second stage, the \(M_1\) joins the running process and writes a word \(c_\text{O}S(r(P), c_\text{O}P)\). At the next stage, the procedure \(M_2\) determines whether the string \(S(r(P), c_\text{O}P)\) belongs to the language \(\mathcal{L}\). If it does not, then the \(M\) accepts the input \(X = c_\text{O}P\). When the \(M_2\) gives an affirmative answer, then \(M\) rejects the \(X\).

We estimate the running time of \(M\) on input \(X = c_\text{O}P\). Since \(c_\text{O}\) is a polynomial encoding and Lemma \([11i]\) is valid, there exists a polynomial \(h_1\) such that the running time of \(M_0\) does not exceed \(h_1(|X|)\). The machine \(M_1\) builds \(c_\text{O}S(r(P), X)\) within \(g(|X| + |c_\text{O}r(P)|) \leq g(2|c_\text{O}P|)\) steps, since \(|c_\text{O}r(P)| \leq |c_\text{O}P|\); the stage \(M_2\) lasts no longer than \(F(D_2 \cdot |c_\text{O}S(r(P), c_\text{O}P)|^{\rho}) \leq F((D_1 \cdot |c_\text{O}P|^{s_1} \cdot (aD_1)^{-\rho}) = F(|c_\text{O}P|/a^\rho)\) steps for \(|c_\text{O}S(r(P), c_\text{O}P)| \geq |c_\text{O}P| > t_1\) by our assumption. Hence, the entire \(M\) will execute its work no more than
\[ T(P) = h_1(|c_0P|) + g(2|c_0P|) + F(|c_0P|/a^\rho) < F(|c_0P|) \] steps for all sufficiently large \(|c_0P|\).

Let us look at the situation that obtains if as \(X\) we take the code of so lengthy a clone \(\hat{M}\) of the machine \(M\) that the inequalities \(|c_0\hat{M}| > \max\{t_0, t_1\}\) and \(T(\hat{M}) < F(|c_0\hat{M}|)\) hold true.

If the \(M\) rejects the input \(c_0\hat{M}\), then the \(M_2\) answers affirmatively, i.e., the string \(S(r(\hat{M}), c_0\hat{M})\) belongs to the language \(\mathcal{L}\). According to the condition (ii), this means that the \(r(\hat{M})\) accepts the input \(c_0\hat{M}\) within \(F(|c_0\hat{M}|)\) steps. Since the machines \(M, \hat{M}\), and \(r(\hat{M})\) are monoclonal, the \(M\) does it too. There is a contradiction.

If the \(M\) accepts the \(c_0\hat{M}\) as its input, then the procedure \(M_2\) answers negatively. In accordance with the sense of the formula \(S(r(\hat{M}), c_0\hat{M})\), this signifies that the machine \(r(\hat{M})\) either rejects the \(c_0\hat{M}\) or its running time on this input is more than \(F(|c_0\hat{M}|)\). By construction and our assumption, the clone \(r(\hat{M})\) cannot operate so long. We have again arrived at a contradiction.

\[ \square \]

**Remark 1.** Apparently, the generalization of Rabin and Fischer’s method has been in essence known in an implicit form for a long time. For example, it is said in the penultimate paragraph of the introduction of the article \[19\] (before the paragraph "Paper outline") that the quadratic increase in the length of the modeling formulae implies a lowering of the lower bound with \(F(n)\) to \(F(\sqrt{n})\) (in our notation), when Compton and Henson’s method is applied. But the author could not find an explicit formulation of the statement similar to Proposition 1 for a reference, although its analog for the space complexity is Lemma 3 in \[19\]. The proof of the proposition is given only for the sake of completeness of the proof of Corollary 3. In addition, Proposition 1 in such form is clearly redundant for the proof of this corollary. However, the author hopes to apply it in further researches.

**Corollary 1.** Under the conditions of the proposition \(\mathcal{L} \notin \text{DTIME}(F(D^{-\zeta} \cdot n^\zeta))\), where \(\zeta = s^{-1} (s = 1 \text{ in case (b)}).\)

**Proof.** Really, \(s_1 = s + \delta\) and \(aD_1 = a(D + \delta)\) tend to \(s\) and \(D\) respectively, when \(a\) tends to one and \(\delta\) tends to zero. Hence, \(\rho = (s + \delta)^{-1}, n^\rho\), and \(D_1^{-\rho}\) accordingly tend to \(s^{-1}, n^{s^{-1}},\) and \(D^{-s^{-1}}\) in this case. \(\square\)
3 Necessary agreements and the main result

In this section, we specify the restrictions on the used Turing machines, the characteristics of their actions, and the methods of recording their instructions and Boolean formulae. These agreements are very important in proving the main theorem. Although any of these restrictions can be omitted at the cost of a complication of proofs.

3.1 On the Turing machines and recording of Boolean formulae

We reserve the following alphabet for the formulae of the signature of the two-element Boolean algebra $\mathcal{B}$:

a) signature symbols $\cap, \cup, C, 0, 1$ and equality sign $\approx$;  

b) Latin letters for the indication of the types of the object variables;  
c) Arabic numerals and comma for the writing of indices;  
d) Logical connectives $\neg, \land, \lor, \rightarrow$;  
e) the signs of quantifiers $\forall, \exists$;  
f) auxiliary symbols: $(),$. All these symbols constitute the first part of a natural language.

Remark 2. Let us pay attention to that we use three different symbols for the denotation of equality. The first is the signature symbol ”$\approx$”. It applies only inside the formulae of a logical theory. The second is the ordinary sign ”$=$”. It denotes the real or assumed equality and is used in our discussions on the formal logical system. The third sign ”$\equiv$” designates the equality in accordance with a definition.

The priority of connectives and operations or its absence is inessential, as a difference in length of formulae is linear in these cases.

Hereinafter we consider only deterministic machines with the fixed tape alphabet $A$, which contains at least four symbols: the first of them is a designated ”blank” symbol, denoted $\Lambda$; the second is a designated ”start” symbol, denoted $\bigstar$; and the last two are the numerals 0,1 (almost as in Section 1.2 of [2]). As usual, the machine cannot write or erase $\bigstar$ symbol.

It is implied that the simulated machines have an only tape, seeing that the transformation of the machine program from a multi-tape variant to a single-tape version is feasible in the polynomial time on the length of the program, at that the running time increases polynomially too [1, 2, 9]. Although the auxiliary machines may be multi-tape.
The machine tape is infinite only to the right, because the Turing machines are often considered in this manner (e.g., [1, 2, 4, 5, 7, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19]). Moreover, such machines can simulate the computations, which is \( T \) steps in length on the two-sided tape machine, in linear time of \( T \). The tape contains initially the start symbol \( \rhd \) in the leftmost square, a finite non-blank input string \( X \), and the blank symbol \( \Lambda \) on the rest of its cells. The head is aimed at the left end of the tape, and the machine is in the special starting state \( q_{\text{start}} = q_0 \). When machine recognizes an input, it enters the accepting state \( q_1 = q_{\text{acc}} \) or the rejecting state \( q_2 = q_{\text{rej}} \).

Our machines have the single-operand instructions of a kind \( q_i \alpha \rightarrow q_j \beta \) as in [11], which differ from double-operand instructions of a form \( q_i \alpha \rightarrow q_j \beta \gamma \), where \( \alpha \in A; \beta, \gamma \in A \cup \{R, L\} \). Even if we regard the execution of a double-operand instruction as one step of computation, then the difference in length of the running time will be linear.

The Turing machines do not fall into a situation when the machine stopped, but its answer remained undefined. Namely, they do not try to go beyond the left edge of the tape; and besides, they 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 \( \ldots \rightarrow q_j \beta \), but there are no instructions beginning with \( q_j \alpha \rightarrow \ldots \) at least for one \( \alpha \in A \). The attempts to go beyond the left edge of the tape are blocked by the replacement of the instructions of a form \( q_i \rhd \rightarrow q_k L \) by \( q_i \rhd \rightarrow q_k \rhd \). The hanging states are eliminated by adding the instructions of a kind \( q_j \alpha \rightarrow q_j \alpha \) for each of the missing alphabet symbol \( \alpha \).

The programs of the single-tape Turing machines with the tape alphabet \( A \) are written by the symbols of this alphabet, as well with the application of the symbols \( q, R, L, \rightarrow, \) Arabic numerals, and comma. This is the second, last part of a natural language.

### 3.2 The main theorem and its corollary

Let \( c_\Omega M \) be a chosen polynomial code of an object \( M \) by a string over a tape alphabet \( A \) — see the beginning of Subsection [2,2]. We suppose that for this encoding, there exists a linear function \( l \) such that the inequalities \( |M| \leq |c_\Omega M| \leq l(|M|) \) hold for any object \( M \) of the natural language described in the previous subsection.

**Theorem 1.** For each deterministic Turing machine \( P \) and every input string \( X \), one can write a closed formula (sentence) \( \Omega(X, P) \) of the signature
of the two-element Boolean algebra $\mathcal{B}$ with the following properties:

(i) there exists a polynomial $g$ such that the code $c_O \Omega(X, P)$ is written within time $g(|X|, |c_O P|)$ for all $X$ and $P$;

(ii) $Th(\mathcal{B}) \vdash \Omega(X, P)$ if and only if the Turing machine $P$ accepts input $X$ within time $\exp(|X|)$;

(iii) for every $\varepsilon > 0$, there is a constant $D > 0$ (depending on the used encoding) such that the inequalities

$$|X| < |c_O \Omega(X, P)| \leq D \cdot |c_O P| \cdot |X|^{2+\varepsilon}$$

hold true for all sufficiently long $X$.

**Proof.** See Sections 4–6. Now we just note that according to the agreement in the beginning of this subsection, the calculation of the lengths of all components of the modeling formulae will be based on the estimate of the quantity of all the symbols, of the natural language of Subsection 3.1 involved in their recording.

At first, we will construct the very long formulae that simulate the computations. These formulae will have the huge number of the ”redundant” variables. We will take care of the brief record of the constructed formulae after we ascertain the correctness of our modeling (see Propositions 2 (ii), 3 and 4(ii) below). The modified Stockmeyer and Meyer method is substantially used at that.

**Corollary 2.** For every $\varepsilon > 0$, $Th(\mathcal{B}) \notin \text{DTIME}(\exp(D^{-\rho} \cdot n^\rho))$, where $\rho = (2+\varepsilon)^{-1}$

**Proof.** It straightforwardly follows from the theorem and Corollary 1.

**Corollary 3.** The class $\text{P}$ is a proper subclass of the class $\text{PSPACE}$.

**Proof.** Really, the theory $Th(\mathcal{B})$ does not belong to the class $\text{P}$ in accordance with the previous corollary, and this theory is equivalent to the language $TQBF$ relatively polynomial reduction. But the second language belongs to the class $\text{PSPACE}$, moreover, it is polynomially complete for this class [18].

**Remark 3.** This result is quite natural and expected for a long time. Its proof is yielded by one of the few possible ways. Indeed, since the language $TQBF$ is polynomially complete for the class $\text{PSPACE}$, the inequality $\text{P} \neq \text{PSPACE}$ implies the impossibility of the inclusion $Th(\mathcal{B}) \in \text{P}$ that is almost equivalent to $Th(\mathcal{B}) \notin \text{DTIME}(\exp(dn^\delta))$ for suitable $d, \delta > 0$, as it is clear that $Th(\mathcal{B}) \in \text{DTIME}(\exp(d_1 n))$ for some $d_1 > 0$. 

13
3.3 Supplementary denotations and arrangements

We introduce the following abbreviations and arrangements for the improvement in perception (recall that ":=\ A\ means\ \ "A\ is a designation for\ \ A"):
(1) the square brackets and (curly) braces are equally applied with the ordinary parentheses in long formulae; (2) the connective \(\land\) is sometimes written as \(\&\); (3) \(\cap\) and \(\land\ (\&)\ connect more closely than \(\cup\) and \(\lor\rightarrow\); (4) \(x<y\ \iff\ x\approx0\land y\approx1\); (5) \(\langle a_0,\ldots,a_n\rangle<\langle b_0,\ldots,b_n\rangle\) is the comparison of tuples in lexicographic ordering, i.e., it is the formula
\[
\alpha_0<\beta_0\lor\left\{\alpha_0\approx\beta_0\land\left[\alpha_1<\beta_1\lor\left(\alpha_2<\beta_2\lor\left(\alpha_3<\beta_3\ldots\right)\right)\right]\right\}.
\]

The symbol \(\hat{x}\) signifies an ordered set \(\langle x_0,\ldots,x_n\rangle\), whose length is fixed. It is natural that "the formula" \(\hat{x}\approx\hat{\alpha}\) denotes the system of equations \(x_0\approx\alpha_0\land\ldots\land x_n\approx\alpha_n\). The tuples of variables with two subscripts will occur only in the form where the first of these indices is fixed, for instance, \(\langle u_{k,0},\ldots,u_{k,n}\rangle\), and we will denote it by \(\hat{u}_k\).

Counting the length of a formula in the natural language, we are guided by the rule: a tuple \(\hat{x}\) has a length of \(n+1\) plus \(M\), which is the quantity of symbols involved in a record of the indices \(0,\ldots,n\). The inequality \(|\hat{x}\approx\hat{\alpha}|\leq M+3n+3\) will hold, if the \(\hat{\alpha}\) is a tuple of constants; and \(|\hat{x}\approx\hat{\alpha}|\leq2M+3n+3\), when it consists of variables.

A binary representation of a natural number \(t\) is denoted by \((t)_2\).

It is known that if \(t=(\hat{\gamma})=(\gamma_0,\ldots,\gamma_n)\) is a binary representation of a natural number \(t\leq\exp(2,\ n)\), then the numbers \(t+1\) and \(t-1\) will be expressed as \((\hat{\gamma}+1)_2=(\gamma_0\oplus\gamma_1\ldots\gamma_{n-1}\ldots\gamma_0,\gamma_1,\ldots,\gamma_{n-2}\oplus\gamma_{n-1}\gamma_n,\gamma_n\oplus\gamma_n\oplus1)\) and \((\hat{\gamma}-1)_2=(\gamma_0\oplus C_{\gamma_1}\ldots\gamma_{n-1}\gamma_n,\ldots,\gamma_{n-2}\oplus C_{\gamma_{n-1}}C_{\gamma_n},\gamma_{n-1}\oplus C_{\gamma_n},\gamma_n\oplus1)\), respectively, where the operation \(\cap\) is written in the form of multiplication \(x\cap y=x\cdot y\); and \(x\oplus y=x\cdot Co(y)\cup Co(x)\cdot y\).

**Lemma 2.** (i) \(|\langle a_0,\ldots,a_n\rangle<\langle b_0,\ldots,b_n\rangle|\leq O(\max\{|\langle a_0,\ldots,a_n\rangle|,|\langle b_0,\ldots,b_n\rangle|\})\).

(ii) If a tuple \((t)_2\) (together with the indices) is \(l\) symbols in length, then the binary representation of the numbers \(t\pm1\) will take up \(O(l^2)\) symbols.

**Proof.** It is obtained by direct calculation. \(\square\)
4 The beginning of the proof of Theorem \[ \text{I} \]

Prior to the writing of a formula $\Omega(X, P)$, we add $2|A|$ the instructions of the idle run to a program $P$, these instructions have the form $q_k\alpha \rightarrow q_k\alpha$, where $k \in \{1(\text{accept}), 2(\text{reject})\}$, $\alpha \in A$. While the machine executes them, the tape configuration does not change.

4.1 The primary and auxiliary variables

In order to simulate the operations of a Turing machine $P$ on an input $X$ within the first $T = \exp(|X|)$ steps, it is enough to describe its actions on a zone, which is $T+1$ squares in width, since if the $P$ starts its run in the zeroth cell, then it can finish a computation at most in the $T$th square. Because the record of the number $(T)_2$ has the $n+1 = |X|+1$ bit, the cell numbers are encoded by the values of the ordered sets of the variables of a kind "$x$: $\hat{x}_t = \langle x_{t,0}, \ldots, x_{t,n} \rangle$, which have a length of $n+1$. The first index $t$, i.e., the color of the record, denotes the step number, after which there appeared a configuration under study on the tape. So the formula $\hat{x}_t \approx \hat{\alpha} \Leftarrow x_{t,0} \approx \alpha_0 \land \ldots \land x_{t,n} \approx \alpha_n$ assigns the number $(\hat{\alpha})$ of the required tape cell in binary notation at the instant $t$.

Let us select so great a number $r$ in order that one can write down all the state numbers of the $P$ and encode all the symbols of the alphabet $A$ by means of the bit combinations of the same length $r+1$ at one time. Thus, $\exp(r+1) \geq |A|+U$, where $U$ is the maximal number of the internal states of the $P$, and if $\beta \in A$, then $c\beta \equiv \langle c\beta_0, \ldots, c\beta_r \rangle$ will be the $(r+1)$-tuple, which codes the $\beta$. So, the encoding $c_0$ applied in Sections 2 and 3 is "outside" (inherent a machine being simulated), and the encoding $c$ is "inner" (inherent a modeling formula).

The formula $\hat{f}_t \approx c\varepsilon$ represents an entry of symbol $\varepsilon$ in some cell after step $t$, where $\hat{f}_t$ is the $(r+1)$-tuple of variables. When the cell, whose number is $(\hat{\mu})$, i.e. the $(\mu)$th cell, contains the symbol $\varepsilon$ after step $t$, then this fact is associated with the quasi-equation (or the clause) of color $t$:

$$\psi_t(\hat{\mu} \rightarrow \varepsilon) \Rightarrow \hat{x}_t \approx \hat{\mu} \rightarrow \hat{f}_t \approx c\varepsilon \Rightarrow (x_{t,0} \approx \mu_0 \land \ldots \land x_{t,n} \approx \mu_n) \rightarrow (f_{t,0} \approx c\varepsilon_0 \land \ldots \land f_{t,r} \approx c\varepsilon_r).$$

The tuples of variables $\hat{q}_t$ and $\hat{d}_t$ are accordingly used to indicate the number of the machine’s internal state and the code of the symbol scanned.
by the head at the instant $t$. For every step $t$, a number $i = (\delta)$ of the 
machine state $q_i$ and a scanned square’s number ($\xi$) together with a symbol 
$\alpha$, which is contained there, are represented by a united $\pi$-formula of color 
t: 

$$
\pi_t(\alpha, (i)_2, \xi) \equiv \tilde{d}_t \approx c\alpha \land \tilde{q}_t \approx \delta \land \tilde{z}_t \approx \hat{\xi} \equiv (d_{t,0} \approx c\alpha_0 \land \ldots \land d_{t,r} \approx c\alpha_r) \land \ni \in (q_{t,0} \approx \delta_0 \land \ldots \land q_{t,r} \approx \delta_r) \land (z_{t,0} \approx \xi_0 \land \ldots \land z_{t,n} \approx \xi_n),
$$

where the ordered sets of variables $\tilde{d}_t$ and $\tilde{q}_t$ have a length of $r+1$; and $\tilde{z}_t$ is 
the $(n+1)$-tuple of variables and is assigned for the storage of the scanned 
cell’s number. The formula expresses a condition for the applicability of 
instruction $q_i\alpha \rightarrow \ldots$; in other words, this is a timer that activates exactly 
this instruction, provided that the head scans the $(\xi)$th cell.

The basic variables $\tilde{x}_t, \tilde{z}_t$, and $\tilde{q}_t, \tilde{f}_t, \tilde{d}_t$ are introduced in great abundance 
in order to facilitate the proof. But a final modeling formula will only contain 
those of them for which $t = 0, \ldots, n$ or $t = T \equiv \exp(n)$ holds. The sets of 
the basic variables have the different lengths. However, this will not lead to 
confusion, since the tuples of the first two types will always be $n+1$ in length, 
whereas the last ones will have a length of $r+1$. The sets of constants or 
other variables may also be different in length, but such tuple will always be 
identically associated to some of the above mentioned ones.

The other variables are auxiliary. They will be described as needed. Their task consists in a determination of the values of the basic variables 
of the color $t + 1$ provided that the primary ones of the color $t$ have the 
"correct" values. Moreover, this transfer has to adequately correspond to 
that instruction which is employed at the step $t + 1$.

**Lemma 3.** If the indices are left out of account, then a clause $\psi_t(\tilde{u} \rightarrow \beta)$ and 
a timer ($\pi$-formula) will be $O(n+r)$ in length.

**Proof.** It is obtained by direct calculation. \(\square\)

### 4.2 The description of an instruction action

The following formula $\varphi(k)$ describes an action of the $k$th instruction $M(k) = q_i\alpha \rightarrow q_j\beta$ (including the idle run’s instructions; see the beginning of this 
section) at some step, where $\alpha \in A, \beta \in A \cup \{R, L\}$:

$$
\varphi(k) \equiv \forall \tilde{u}_k \left\{ \pi_t(\alpha, (i)_2, \tilde{u}_k) \rightarrow \left[ \Delta^{\text{cop}}(\tilde{u}_k(\beta)) \land \forall \tilde{h}_k(\Gamma^{\text{ret}}(\beta) \rightarrow \Delta^{\text{wr}}(\beta) \land \pi_{t+1}(h_k, (j)_2, \tilde{u}_k(\beta))) \right] \right\}.
$$
For the sake of concreteness, we regard that this step has a number $t+1$, so we have placed such subscripts on both $\pi$-formulae. Now we will describe the subformulae of the $\varphi(k)$ with the free basic variables $\hat{x}_t, \hat{q}_t, \hat{z}_t, \hat{d}_t, \hat{f}_t, \hat{x}_{t+1}, \hat{q}_{t+1}, \hat{z}_{t+1}, \hat{d}_{t+1}$, and $\hat{f}_{t+1}$.

The first $\pi$-formula of color $t$ plays a role of a timer. It starts up the fulfillment of the instruction with the prefix $q_k \alpha \to \ldots$ provided that a head scans the $(\hat{u})_k$th square. For every $(n+1)$-tuple $\hat{u}_k$ and a given meta-symbol $\beta \in \{R, L\} \cup A$, the number of the cell that will be scanned by the head after the execution of the instruction $M(k)$ is specified as follows: $\hat{u}_k(R) \equiv ((\hat{u}_k)_{+1})_2; \hat{u}_k(L) \equiv ((\hat{u}_k)_{-1})_2$; and $\hat{u}_k(\beta) \equiv \hat{u}_k$ for $\beta \in A$.

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

$$\Delta^{\cop}(\hat{u}_k(\beta)) \equiv \forall \hat{w}_k [\neg \hat{u}_k \approx \hat{u}_k(\beta) \to \exists \hat{g}_k (\psi_t(\hat{w}_k \to \hat{g}_k) \land \psi_{t+1}(\hat{w}_k \to \hat{g}_k))]$$

If $\beta \in \{R, L\}$, then $\Gamma^{\text{ref}}(\beta) \equiv \psi_t(\hat{u}_k(\beta) \to \hat{h}_k) = \hat{x}_t \approx \hat{u}_k(\beta) \to \hat{f}_t \approx \hat{h}_k$

An informal sense of this formula is the following: it "seeks" a code $\hat{h}_k$ of the symbol, which will be scanned after the next step $t+1$ (by this reason it is named "retrieval"); for this purpose, it "inspects" the square that is to the right or left of the cell $(\hat{u}_k)$. When $\beta \in A$, there is no need to look for anything, so the formula $\Gamma^{\text{ref}}(\beta)$ will be very simple in this case: $\hat{h}_k \approx c \beta$.

The formula $\Delta^{\text{wr}}(\beta)$ "puts" the symbol, whose code is $\hat{h}_k$ and color is $t+1$, in the $(\hat{u}_k(\beta))$th square: $\Delta^{\text{wr}}(\beta) \equiv \psi_{t+1}(\hat{u}_k(\beta) \to \hat{h}_k)$. The second $\pi$-formula of the color $t+1$ aims the head at the $(\hat{u}_k(\beta))$th cell; places the symbol $\hat{h}_k$ in this location; and changes the number of the machine state for $j$: $\hat{z}_{t+1} \approx \hat{u}_k(\beta) \land \hat{d}_{t+1} \approx \hat{h}_k \land \hat{q}_{t+1} \approx j_2$

**Lemma 4.** (i) If $\beta \in A$, then the formulae $\Gamma^{\text{ref}}(\beta); \pi_{t+1}(\hat{h}_k, (j)_2, \hat{u}_k(\beta)); \Delta^{\text{wr}}(k, \beta); \Delta^{\cop}(\hat{u}_k(\beta))$; and $\varphi(k)$ will be $O(|\psi_{t+1}(\hat{w}_k \to \hat{g}_k)|)$ in length.

(ii) For $\beta \in \{R, L\}$, each of these formulae is $O(n \cdot |\psi_{t+1}(\hat{w}_k \to \hat{g}_k)|)$ in length.

**Proof.** This follows from Lemmata 2 and 3 by direct calculation.

### 4.3 The description of the running steps and configurations

At first, we will construct a formula $\Phi^{(0)}(P)$ describing one step of the machine run, when the $P$ is applied to a configuration that arose after some step
Next, we will describe by means of the formulae the machine actions over an exponential period of time; at that, the Stockmeyer and Meyer method will be used.

### 4.3.1 One step

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

\[
\Phi^{(0)}(P)(\hat{y}_t, \hat{y}_{t+1}) \equiv \bigwedge_{0 < k \leq N} \varphi(k)(\hat{y}_t, \hat{y}_{t+1}),
\]

where \( \hat{y}_t \equiv \langle \hat{x}_t, \hat{q}_t, \hat{z}_t, \hat{d}_t, \hat{f}_t \rangle \) and \( \hat{y}_{t+1} \equiv \langle \hat{x}_{t+1}, \hat{q}_{t+1}, \hat{z}_{t+1}, \hat{d}_{t+1}, \hat{f}_{t+1} \rangle \) are two \((2n+3r+5)\)-tuples of its free variables.

Let us denote the quantifier-free part of a formula \( \chi \) as \( \langle \chi \rangle \).

**Lemma 5.** (i) If \( \hat{x}_t \neq \hat{w}_k \), then a clause \( \psi_t(\hat{w} \rightarrow \varepsilon) \) will be true independently of the value of variables \( f_t \). In particular, a quasi-equation, which is contained into the record of \( \langle \Delta^{\text{cop}}(\hat{u}_k) \rangle \), will be true, if its color is \( t \) or \( t+1 \), and at the same time \( \hat{x}_t \neq \hat{w}_k \) or \( \hat{x}_{t+1} \neq \hat{w}_k \), respectively.

(ii) For some constant \( D_1 \), the inequality

\[
|\Phi^{(0)}(P)(\hat{y}_t, \hat{y}_{t+1})| \leq D_1 \cdot |c_O P| \cdot |\varphi(N)|
\]

holds.

**Proof.** (i) The premises of clauses are false in these cases.

(ii) If the quantity of the program \( P \) instructions is not equal to zero, i.e., \( N - 2|A| \neq 0 \), then \( N \cdot \lceil \lg N \rceil < D_2 \cdot |c_O P| \). This implies the assertion of the lemma.

### 4.3.2 The configurations and the exponential quantity of steps

The formulae \( \Phi^{(s+1)}(P)(\hat{y}_t, \hat{y}_{t+\epsilon(s+1)}) \) conform to the actions of machine \( P \) over a period of time \( \epsilon(s) \equiv \exp(s) \). They are defined by induction:

\[
\Phi^{(s+1)}(P) \equiv \exists \hat{v} \forall \hat{a} \forall \hat{b} \left\{ \left[ (\hat{y}_t \approx \hat{a} \wedge \hat{v} \approx \hat{b}) \lor (\hat{v} \approx \hat{a} \wedge \hat{b} \approx \hat{y}_{t+\epsilon(s+1)}) \right] \rightarrow \Phi^{(s)}(P)(\hat{a}, \hat{b}) \right\},
\]

18
where $\widehat{v}, \widehat{a}, \widehat{b}$ are the $(2n+3r+5)$-tuples of the new auxiliary variables.

Let $L(t)$ be a configuration, which is recorded on the tape after step $t$ (it may be unrealizable): namely, at the instant $t$, every cell, whose number is $\widehat{\mu}$, contains a symbol $\varepsilon(\widehat{\mu})$; the scanned square has the number $\widehat{\eta}$; and a machine is ready to execute an instruction $q_i \alpha \rightarrow \ldots$. Then the following formula corresponds to this configuration (we recall that $T = \exp(n)$):

$$
\Psi_{L(t)}(\widehat{y}_t) \equiv \pi_t(\alpha, (i)_2, \widehat{\eta}) \& \bigwedge_{0 \leq (\widehat{\mu})_2 \leq T} \psi_t(\widehat{\mu} \rightarrow \varepsilon(\widehat{\mu})).
$$

It has $2n+3r+5$ free variables $\widehat{y}_t = (\widehat{x}_t, \widehat{q}_t, \widehat{z}_t, d_t, f_t)$.

## 5 The simulation of one running step

We simply associated the formulae, which were constructed earlier, with the certain components of programs or with processes. However one cannot assert that these formulae simulate something, i.e., they will not always turn true, when the events, which are described by them, are real.

### 5.1 Simulating formula

Let us definite

$$
\Omega^{(0)}(X, P)(\widehat{y}_t, \widehat{y}_{t+1}) \equiv [\Psi K(t)(\widehat{y}_t) \& \Phi^{(0)}(P)(\widehat{y}_t, \widehat{y}_{t+1})] \rightarrow \Psi K(t+1)(\widehat{y}_{t+1}).
$$

We will prove in this section 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 on the Boolean algebra $\mathcal{B}$ if and only if the machine $P$ transforms the configuration $K(t)$ into $K(t+1)$ in one step. So we can say that this formula models the machine actions at the step $t+1$.

**Remark 4.** One can regard that the formula $\Omega^{(0)}(X, P)(\widehat{y}_t, \widehat{y}_{t+1})$ is the analog of the Cook’s method formula $A_{0, m}(\widehat{U}, \widehat{V})$, which was applied in the proof of Theorem 4.3 in [18], here $\widehat{U}$ and $\widehat{V}$ are the sequences $(u_1, \ldots, u_m)$ and $(v_1, \ldots, v_m)$ of the Boolean variables and $m = q(|X|)$ is the value of suitable polynomial $q$ on the length of input $X$. Indeed, the sentence $\exists \widehat{U} \exists \widehat{V} A_{0, m}(\widehat{U}, \widehat{V})$ is true if and only if the configuration encoded by $v_1 \ldots v_m$ follows from the configuration that corresponds to $u_1 \ldots u_m$ in at most one step of the $P$ (these $m$ and $P$ are $n$ and $\mathfrak{M}$ in [18]).
However, there are solid arguments to believe that the real analog of the formula $A_{0,m}(\tilde{U}, \tilde{V})$ is the $\Phi^{(0)}(P)(\tilde{y}_t, \tilde{y}_{t+e(s+1)})$ nevertheless. We will return to the discussion of this analogy in Subsection 8.2.

5.2 The single-valuedness of modeling and the special values of variables

Let $K(t+1)$ be a configuration that has arisen from a configuration $K(t)$ as a result of the machine $P$ action at the step $t+1$.

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

(ii) If a formula $\Omega^{(0)}(X, P)(\hat{y}_t, \hat{y}_{t+1})$ is identically true over algebra $\mathcal{B}$, then the machine $P$ cannot convert the configuration $K(t)$ into the configuration, which differs from $K(t+1)$, at the step $t+1$.

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

$$\Upsilon_{t+1}(\tilde{y}_t, \tilde{y}_{t+1}) \equiv [\Psi K(t)(\tilde{y}_t) \& \Phi^{(0)}(P)(\tilde{y}_t, \tilde{y}_{t+1})] \rightarrow \Psi L(t+1)(\tilde{y}_{t+1})$$

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

Let $M(k) = q_i \alpha \rightarrow \ldots$ be an instruction that is applicable to the configuration $K(t)$; and $(\tilde{\eta})$ be a number of the scanned square. We specify $\hat{d}_t = c\alpha$, $\hat{q}_t = (i)_2$, $\hat{z}_t = \tilde{\eta}$. Then $\pi$-formula $\pi_t(\alpha, (i)_2, \tilde{\eta})$, which is in the record of $\Psi K(t)$, is true.

Let us consider a formula $\varphi(l)$ that conforms to some instruction $M(l) = q_b \theta \rightarrow \ldots$ that differs from the $M(k)$. This formula has a timer $\pi_t(\theta, (b)_2, \tilde{u}_t)$ as the first premise. For the selected values of the variables $\hat{d}_t; \hat{q}_t$; and $\hat{z}_t$, the timer takes the form of $c\alpha \approx c\theta \land (i)_2 \approx (b)_2 \land \tilde{\eta} \approx \tilde{u}_t$. It is obvious that if $\alpha \neq \theta$; or $i \neq b$; or $\tilde{u}_t \neq \tilde{\eta}$, then this $\pi$-formula will be false, and the whole $\varphi(l)$ will be true.
Thus, let \( \varphi(k) \) be a formula that corresponds to the instruction \( M(k) = q_i \alpha \rightarrow q_j \beta \); and \( \hat{u}_k = \hat{\eta} \). Let us define \( \hat{d}_{t+1} = c \lambda \); \( \hat{q}_{t+1} = (j) \); \( \hat{z}_{t+1} = \hat{\eta}(\beta) \), where \( (\hat{\eta}(\beta)) \) is a number of the square, which will be scanned by the machine head after the fulfillment of the instruction \( M(k) \); and \( \lambda \) is the symbol, which the head will see there. For these \( \hat{u}_k \) and selected values of \( \hat{d}_{t+1}, \hat{q}_{t+1}, \hat{z}_{t+1} \), the \( \pi \)-formula, which enters into the record of \( \Psi K(t+1) \), becomes true. But the conclusion of the quantifier-free part \( \langle \varphi(k) \rangle \) contains a slightly different timer \( \pi_{t+1}(\hat{h}_k, (j), \hat{u}_k(\beta)) \); in this timer, the only equality \( \hat{d}_{t+1} \approx \hat{h}_k \) included in it raises doubts for the time being.

Let us assign \( \hat{x}_t = \hat{\eta}(\beta) \). Since we consider the case, when \( \hat{u}_k = \hat{\eta} \), the equality \( \hat{u}_k(\beta) = \hat{\eta}(\beta) \) holds true too. Therefore the quasi-equation, of the color \( t \), which enters into the \( \langle \Delta^{\text{cop}}(\hat{u}_k(\beta)) \rangle \), is true for all \( \hat{u}_k \neq \hat{\eta}(\beta) \) and irrespective of the values of the tuples \( \hat{f}_t \) and \( \hat{g}_k \) according to Lemma 5(i).

For the same reason, all the clauses that are included in the \( \Psi K(t) \) are true, except the clause \( \psi_t(\hat{\eta}(\beta) \rightarrow \lambda) \) for \( \beta \in \{R, L\} \) or \( \psi_t(\hat{\eta} \rightarrow \alpha) \) for \( \beta \in A \). We set the value of the tuple \( \hat{f}_t \) as \( c \lambda \), if \( \beta \in \{R, L\} \), or as \( c \alpha \), if not. Now, the questionable clause from the \( \Psi K(t) \) becomes true, because its premise and conclusion are true.

If \( \hat{h}_k \neq c \lambda \), then the formula \( \Gamma^{\text{ret}}(\beta) \) will be false, since it is either \( \psi_t(\hat{\eta}(\beta) \rightarrow \hat{h}_k) \) for \( \beta \in \{R, L\} \), or \( \hat{h}_k \approx c \beta \) for \( \beta \in A \). Hence the whole formula \( \langle \varphi(k) \rangle \) becomes true in this case. When \( \hat{h}_k = c \lambda \), the terminal \( \pi \)-formula in the \( \langle \varphi(k) \rangle \) becomes true, since \( \hat{d}_{t+1} = c \lambda \).

If the "incorrect" formula \( \Psi L(t+1) \) has a mistake in the record of timer or clause \( \psi_{t+1}(\hat{\eta}(\beta) \rightarrow \lambda) \), we will define \( \hat{x}_{t+1} = \hat{\eta}(\beta) \) and \( \hat{f}_{t+1} = c \lambda \) (we recall that \( \lambda = \beta \) for \( \beta \in A \)). But when these fragments are that as they should be, however, there is another "incorrect" clause \( \psi_{t+1}(\hat{\mu} \rightarrow \rho) \), where \( \rho \) is different from "real" \( \delta \), we will assign \( \hat{x}_{t+1} = \hat{\mu} \) and \( \hat{f}_{t+1} = \hat{g}_k = c \delta \). We obtain again that all the quasi-equations of the color \( t+1 \) in the formulae \( \langle \Delta^{\text{cop}}(\hat{u}_k(\beta)) \rangle \) and \( \Delta^{\text{ur}}(\beta) \) are true in both of these cases on the grounds of Lemma 5(i) or because their premises and conclusions are true. Therefore the whole formula \( \langle \varphi(k) \rangle \) is true. All the clauses contained in \( \Psi K(t+1) \) 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 \( \Omega^{(0)}(X, P)(\hat{g}_t, \hat{g}_{t+1}) \) are true, and

\[ \text{We note that this is the only case when we need to set the values of the variables } \hat{g}_k. \]
the configurations $K(t+1)$ and $L(t+1)$ are different; the “incorrect” formula $\Upsilon_{t+1}$ is false.

In view of the fact that the configuration $L(t+1)$ may differ from the real $K(t+1)$ in any place, Item (i) is established too. \hfill \Box

5.3 The sufficiency of modeling

We will now prove a converse to Proposition 2(ii).

Proposition 3. Let $K(t+1)$ be a configuration that has arisen from a configuration $K(t)$ as a result of an action of the machine $P$ at the step $t+1$. Then the formula $\Omega^{(0)}(X,P)(\hat{y}_t, \hat{y}_{t+1})$ is identically true on algebra $B$.

Proof. Let $M(k) = q_i \alpha \rightarrow q_j \beta$ be the instruction that transforms the configuration $K(t)$ into the $K(t+1)$; and $\varphi(k)(\hat{y}_t, \hat{y}_{t+1})$ be a formula, which is written for this instruction. This formula is the consequence of the $\Phi^{(0)}(P)(\hat{y}_t, \hat{y}_{t+1})$.

Let us replace the $\varphi(k)$ by a conjunction of formulae $\varphi(k)(\mu)$, they are each obtained as the result of the substitution the various values of the universal variables $\hat{k}$ for the variables themselves. Every formula $\varphi(k)(\mu)$ contains the premise $\hat{d}_t \approx c \alpha \land \hat{q}_t \approx (i)_2 \land \hat{z}_t \approx \hat{\mu}$, one of them coincides with the only timer $\pi_t(\alpha, \hat{\delta}, \hat{\eta})$ included in the $\Psi K(t)$ for $\hat{u} = \hat{\mu} = \hat{\eta}$ and $i = (\hat{\delta})$, as the instruction $M(k)$ is applicable to the configuration $K(t)$. Therefore the formula $\Psi K(t) \land \Delta^{cop}(\hat{\eta}(\beta)) \land \forall \hat{h}_k \{ \Gamma^{rel}(\beta)(\hat{\eta}) \rightarrow [\Delta^{w^r}(\beta)(\hat{\eta}) \land \pi_{t+1}(\hat{h}_k, (j)_2, \hat{z}(\beta))] \}$ follows from the $\Psi K(t)$ and $\varphi(k)(\hat{\eta})$.

The formula $\Delta^{cop}(\hat{\eta}(\beta))$ begins with the quantifiers $\forall \hat{w}_k$. Let us replace this formula by a conjunction that is equivalent to it, we substitute all possible values for the variables $\hat{w}_k$ to this effect. For every value of $\hat{w}_k$, there is a unique value of the tuple $\hat{g}_k$ such that the clause $\psi_{\hat{h}_k}(\hat{w}_k, \hat{g}_k)$ enters into the formula $\Psi K(t)$. When these values of $\hat{g}_k$ are substituted in their places, we will obtain all the quasi-equations from the $\Psi K(t+1)$, except one.

For the appropriate value of $\hat{h}_k$, either the formula $\Gamma^{rel}(\beta)(\hat{\eta})$ coincides with some clause existing in the $\Psi K(t)$, or it becomes true: $\hat{h}_k \approx c \beta$, owing to the applicability of the instruction $M(k)$ to the configuration $K(t)$. In any case, the formula $\Delta^{w^r}(\beta)(\hat{\eta})$ in an explicit form contains the quasi-equation $\psi_{t+1}(\hat{\eta}(\beta) \rightarrow \ldots)$, which is missing in the $\Psi K(t+1)$ so far; and the tuple $\hat{h}_k$ obtains the concrete value. If we substitute this value in the concluding $\pi$-formula of the $\varphi(k)$, then we will obtain the necessary timer $\pi_{t+1}(\hat{h}_k, (j)_2, \hat{z}(\beta))$ from the $\Psi K(t+1)$. \hfill \Box
6 The construction of the formula $\Omega(X, P)$

6.1 The simulation of the exponential computations

Let us define the formulae $\Omega^{(s)}(X, P)(\tilde{y}_t, \tilde{y}_{t+e(s)})$ that model $e(s) \equiv \exp(s)$ running steps of a machine $P$, when it applies to a configuration $K(t)$:

$$
\Omega^{(s)}(X, P)(\tilde{y}_t, \tilde{y}_{t+e(s)}) \equiv [\Psi K(t)(\tilde{y}_t) \& \Phi^{(s)}(P)(\tilde{y}_t, \tilde{y}_{t+e(s)})] \rightarrow \\
\rightarrow \Psi K(t+e(s))(\tilde{y}_{t+e(s)}).
$$

Proposition 4. Let $t, s \geq 0$ be the integers such that $t + e(s) \leq T$.

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

(ii) The formula $\Omega^{(s)}(X, P)(\tilde{y}_t, \tilde{y}_{t+e(s)})$ is identically true over the Boolean algebra $\mathcal{B}$ if and only if the machine $P$ converts the configuration $K(t)$ into $K(t + e(s))$ within $e(s)$ steps.

Proof. Induction on the parameter $s$. For $s = 0$, Item (i) is a consequence of 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 $\Phi^{(s+1)}(P)(\tilde{y}_t, \tilde{y}_{t+e(s+1)})$ in the equivalent, but longer form:

$$
\exists \widehat{v} \{ \forall \widehat{a} \forall \widehat{b}[(\tilde{y}_t \approx \widehat{a} \land \tilde{v} \approx \widehat{b}) \rightarrow \Phi^{(s)}(P)(\widehat{a}, \widehat{b})] \& \forall \widehat{a} \forall \widehat{b}[(\tilde{v} \approx \widehat{a} \land \tilde{b} \approx \tilde{y}_{t+e(s+1)}) \rightarrow \\
\rightarrow \Phi^{(s)}(P)(\widehat{a}, \widehat{b}) \}. 
$$

The following formula results from this immediately:

$$
\Xi_{s+1} \equiv \exists \widehat{v} \{ \Phi^{(s)}(P)(\tilde{y}_t, \widehat{v}) \& \Phi^{(s)}(P)(\widehat{v}, \tilde{y}_{t+e(s+1)}) \}. 
$$

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

Let the machine $\bar{P}$ transforms the configuration $K(t)$ into the $K(t + e(s))$ within $e(s)$ steps, and it converts the latter into the $K(t + e(s))$ within the same time.

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

$$\Psi K(t)(\hat{y}_t) \land \Phi^{(s)}(P)(\hat{y}_t, \hat{y}_{t+e(s)}) \land \Psi K(t + e(s))(\hat{y}_{t+e(s)})$$

and

$$\{\Psi K(t + e(s))(\hat{y}_{t+e(s)}), \Phi^{(s)}(P)(\hat{y}_{t+e(s)}, \hat{y}_{t+e(s+1)})\} \rightarrow \Psi K(t + e(s+1))(\hat{y}_{t+e(s+1)})$$

are identically true. Therefore, when we change the variables under the sign of the quantifier, we obtain from this that the following formula

$$\forall \hat{v}\{[\Psi K(t)(\hat{y}_t) \land \Phi^{(s)}(P)(\hat{y}_t, \hat{v}) \land \Phi^{(s)}(P)(\hat{v}, \hat{y}_{t+e(s+1)})] \rightarrow \Psi K(t + e(s + 1))(\hat{y}_{t+e(s+1)})\}$$

is identically true as well. This formula is equivalent to $[(\Psi K(t) \land \Xi_{s+1}) \rightarrow \Psi K(t + e(s+1))](\hat{y}_t, \hat{y}_{t+e(s+1)})$, 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 $\Omega^{(s+1)}(X, P)(\hat{y}_t, \hat{y}_{t+e(s+1)})$ is equivalent to $\Psi K(t) \land \Xi_{s+1}$ in accordance with the foregoing argument, the inductive step of Item (ii) is proven in one direction.

Now let the configurations $L(t + e(s + 1))$ and $K(t + e(s + 1))$ be different. For some $\hat{v}_1, \hat{y}_{t+e(s+1)}$, the formula

$$\{[\Psi K(t + e(s)) \land \Phi^{(s)}(P)] \rightarrow \Psi K(t + e(s + 1))\}(\hat{v}_1, \hat{y}_{t+e(s+1)})$$

is true, but

$$\{[\Psi K(t + e(s)) \land \Phi^{(s)}(P)] \rightarrow \Psi L(t + e(s + 1))\}(\hat{v}_1, \hat{y}_{t+e(s+1)})$$

is false by the inductive assumption of Item (ii). Therefore, the conclusion of the second formula is false, and its premise is true, i.e., the $\Psi L(t + e(s + 1))(\hat{y}_{t+e(s+1)})$ is false, and the $\Phi^{(s)}(P)(\hat{v}_1, \hat{y}_{t+e(s+1)})$ and the $\Psi K(t + e(s))(\hat{v}_1)$ are true. Since the last formula and the $\Psi K(t)(\hat{v}_0)$ are true for some special $\hat{v}_0$, which exists due to induction proposition of Item (i), the formula $\Phi^{(s)}(P)(\hat{v}_0, \hat{v}_1)$ is true. Thus, the implication $\{[\Psi K(t) \land \Phi^{(s+1)}(P)] \rightarrow \Psi L(t + e(s + 1))\}(\hat{v}_0, \hat{y}_{t+e(s+1)})$ has a true premise, and a false conclusion, therefore it is not identically true. Item (ii) is proven.

Inasmuch as the configuration $L(t + e(s + 1))$ may differ from the current one at any position, to finish the proof of Item (i) we set the values of the variables $\hat{v}_1$ in a special manner, using the inductive hypothesis. □
6.2 The short recording of the initial configuration and the condition of the successful termination of the machine run

Since we have the instructions for the machine run in the idle mode (see the beginning of Section 4), the statement that the machine \( P \) accepts an input string \( X \) within \( T = \exp(n) \) steps can be written rather briefly — by means of one quantifier-free formula of the color \( T \): \( \chi(\omega) \equiv \hat{q}_T \approx (1) \). This formula has a length of \( 4r+3 \) symbols nonmetering the indices. The writing of the first index \( T \) occupies \( \lfloor \lg T \rfloor + 1 \) digits (indices are written in decimal notation, not in binary), where \( \lg m = \log_{10} m \), \( [y] \) is the integer part of a number \( y \). The maximum length of the second indices is \( \lfloor \lg r \rfloor + 1 \), and so we have \( |\chi(\omega)| < (4r+3) \cdot (\lfloor \lg T \rfloor + \lfloor \lg r \rfloor + 2) \).

The formula \( \Psi K(t) \) was introduced in Subsection 4.3.2 to describe a configuration arising after the step \( t \). It is very long — much longer than \( n \cdot \exp(n) \). However, the initial configuration consists of the input string \( X \), which occupies the \(|X|\) squares to the right of the edge of a tape; the head points to this extreme left cell; and the remaining part of the tape is empty, starting with the cell, whose number is \(|X|+1=(\hat{\gamma})+1\). Therefore one can describe the initial tape configuration \( K(0) \) by a brief universal formula:

\[
\chi(0) \equiv \pi_0(\triangleright, \hat{0}, \hat{0}) \land \bigwedge_{0 \leq \langle \hat{\eta} \rangle \leq |X|} \psi_0(\hat{\eta} \to \alpha(\eta)) \land \forall \hat{u}_0[\hat{u}_0 > \hat{\gamma} \to \psi_0(\hat{u}_0 \to \Lambda)],
\]

where \( \Lambda \) denotes blank symbol; \( \triangleright \) is a sign of the left end of the tape; and \( \alpha(\eta) \) is a symbol, which is located in the number \( (\hat{\eta}) \) cell; and the \( \pi \)-formula of the color 0 signifies that a mechanism is ready for the execution of the instruction \( q_0 \triangleright \to \ldots \) at the zeroth instant, and the machine head is positioned on the extreme left square of tape and scans \( \triangleright \) symbol.

**Lemma 6.** (i) The formulae \( \chi(0) \) and \( \Psi K(0) \) are equivalent to each other.

(ii) \(|\chi(0)(\hat{\eta}_0)| \leq D_2 \cdot |X| \cdot |\psi_0(\hat{u}_0 \to \Lambda)| \) for a proper constant \( D_2 \).

**Proof.** (i) The quantifier-free part of the formula \( \chi(0) \) simply coincides with the initial fragment of the formula \( \Psi K(0) \). If we replace the second part of formula \( \chi(0) \), which begins with the quantifiers \( \forall \hat{u}_0 \), with its equally matched conjunction, the rest of the clauses from \( \Psi K(0) \) will appear.

(ii) According to Lemmata 2(i) and 3, the system of inequalities \( \hat{u}_0 > \hat{\gamma} \) has a length of the same order as \(|\psi_0(\hat{u}_0 \to \Lambda)|\), a quantifier prefix is a bit
shorter. Hence \(|\forall \tilde{u}_0[\tilde{u}_0 > \tilde{y} \rightarrow \psi_0(\tilde{u}_0 \rightarrow \Lambda)]| = O(|\psi_0(\tilde{u}_0 \rightarrow \Lambda)|). Since the expression \(\chi(\tilde{y}_0)\) includes \(|X|+1\) quasi-equations of a form \(\psi_0(\tilde{y} \rightarrow \alpha(\eta))\) and the timer, which have a length of the same order as \(|\psi_0(\tilde{u}_0 \rightarrow \Lambda)|\) by Lemma \(\text{XX}\), the whole formula \(\chi(\tilde{y}_0)\) is not more than \(D_2 \cdot |X| \cdot |\psi_0(\tilde{u}_0 \rightarrow \Lambda)|\) in length for some constant \(D_2\). \(\square\)

### 6.3 Simulating formula \(\Omega(X, P)\)

Let us define

\[
\Omega(X, P) \doteq \forall \tilde{y}_0, \tilde{y}_T \left\{ \begin{array}{l}
\chi(\tilde{y}_0) & \& \exists \tilde{v}_n \forall \tilde{a}_n \forall \tilde{b}_n \ldots \exists \tilde{v}_1 \forall \tilde{a}_1 \forall \tilde{b}_1 \\
\left\{ \bigwedge_{1 \leq s \leq n} [ (\tilde{a}_{s+1} \approx \tilde{a}_s \land \tilde{v}_s \approx \tilde{b}_s) \lor (\tilde{v}_s \approx \tilde{a}_s \land \tilde{b}_s \approx \tilde{b}_{s+1})] \right\} \rightarrow \Phi^{(0)}(P)(\tilde{a}_1, \tilde{b}_1) \right\} \rightarrow \chi(\omega)(\tilde{y}_T)\}
\]

(1)

here we designate \(\tilde{a}_{n+1} = \tilde{y}_0, \tilde{b}_{n+1} = \tilde{y}_T\) in the record of the ”big” conjunction for the sake of brevity.

**Proposition 5.** The formula \(\Omega(X, P)\) has the property (ii) from the statement of Theorem \(\text{[XX]}\). In other words, this sentence is true on the Boolean algebra \(\mathcal{B}\) if and only if the machine \(P\) accepts the input \(X\) within \(T\) steps.

**Proof.** Let \(\Theta_s = \Theta_s(\tilde{a}_s, \tilde{a}_{s+1}, \tilde{b}_s, \tilde{b}_{s+1})\) be a denotation for a disjunction of equalities \((\tilde{a}_{s+1} \approx \tilde{a}_s \land \tilde{v}_s \approx \tilde{b}_s) \lor (\tilde{v}_s \approx \tilde{a}_s \land \tilde{b}_s \approx \tilde{b}_{s+1}). If we carry the quantifiers through the subformulae, which do not contain the corresponding variables (according to the agreement of Subsection \(\text{[XX]}\) that a conjunction connects more intimately than an implication), then we will obtain that the part of the formula \(\Omega(X, P)\), which is located in the big square brackets in (1), is equivalent to each of the three following formulae:

1) \(\chi(\tilde{y}_0) \& \exists \tilde{v}_n \forall \tilde{a}_n \forall \tilde{b}_n \ldots \exists \tilde{v}_1 \forall \tilde{a}_1 \forall \tilde{b}_1 \{ \bigwedge_{1 \leq s \leq n} \Theta_s \rightarrow \Phi^{(0)}(P)(\tilde{a}_1, \tilde{b}_1)\} ;\)

2) \(\Psi K(0)(\tilde{y}_0) \& \exists \tilde{v}_n \forall \tilde{a}_n \forall \tilde{b}_n \ldots \exists \tilde{v}_1 \forall \tilde{a}_1 \forall \tilde{b}_1 \{ \Theta_n \rightarrow [\Theta_{n-1} \rightarrow (\ldots \rightarrow \\
\rightarrow \{ \Theta_1 \rightarrow \Phi^{(0)}(P)(\tilde{a}_1, \tilde{b}_1)\} \ldots )] ;\}

3) \(\Psi K(0)(\tilde{y}_0) \& \exists \tilde{v}_n \forall \tilde{a}_n \forall \tilde{b}_n \{ \Theta_n \rightarrow \exists \tilde{v}_{n-1} \forall \tilde{a}_{n-1} \forall \tilde{b}_{n-1} [\Theta_{n-1} \rightarrow (\ldots \rightarrow \\
\rightarrow \exists \tilde{v}_1 \forall \tilde{a}_1 \forall \tilde{b}_1 \{ \Theta_1 \rightarrow \Phi^{(0)}(P)(\tilde{a}_1, \tilde{b}_1)\} ] \ldots \} .\)
According to the definition, the formula $\exists \hat{v}_s \forall \hat{a}_s \forall \hat{b}_s (\Theta_s \rightarrow \Phi^{(s-1)}(P)(\hat{a}_s, \hat{b}_s))$ contracts into the $\Phi^{(s)}(P)(\hat{a}_{s+1}, \hat{b}_{s+1})$. Therefore the whole $\Omega(X, P)$ is equivalent to the $\forall \hat{y}_0, \hat{y}_T \left[ (\Psi K(0) \& \Phi^{(n)}(P)) \rightarrow \chi(\omega) \right]$. Consequently, based on Proposition 4(ii) and Lemma 6(i), one could say that the formula (1) is the modeling formula.

### 6.4 The time of writing of $\Omega(X, P)$

The simulating formula $\Omega(X, P)$ is described by the definition (1) in an explicit form, this notation allows us to design an algorithm for its construction. It remains only to prove the properties (i) and (iii) of the statement of Theorem 1. Before we substantiate the polynomiality of the algorithm, we will make sure that the formula $\Omega(X, P)$ of a form (1) has a polynomial length. We recall that the length of a formula is calculated in the natural language — see Subsections 3.1 and 3.2.

**Lemma 7.** There exists a constant $D > 0$ such that it does not depend on the $P$ and $n$ and the inequalities $|X| \leq |\Omega(X, P)| \leq D \cdot |P| \cdot |X|^{2+\varepsilon}$ hold for all the long enough $X$ and any preassigned $\varepsilon > 0$.

**Proof.** Many components of the modeling formula were estimated already during their description, but their lengths were estimated on the assumption that their subformulae are written with basic variables $\langle \hat{x}_t, \hat{q}_t, \hat{z}_t, \hat{a}_t, \hat{f}_t \rangle$, which were denoted in Subsection 4.3 as $\hat{y}_t$. However, they are not included in the composition of the subformulae of $\Phi^{(0)}(P)(\hat{a}_1, \hat{b}_1)$ — we have written the variables from the tuples $\hat{a}_1$ and $\hat{b}_1$ instead theirs. Namely, the first $n+1$ variables in the tuple $\hat{a}_1$ serve as $\hat{x}_t$, and they serve as $\hat{x}_{t+1}$ in the tuple $\hat{b}_1$; the second $r+1$ variables in $\hat{a}_1$ are put instead of $\hat{q}_t$, and they are put in place of $\hat{q}_{t+1}$ in $\hat{b}_1$ and so on. Certainly, this replacement has no an influence on a length of those formulae, where the variables are located, if one disregards a length of indices.

However, the length of the indices has changed markedly. Just because of this reason, they were earlier taken into account only implicitly for the estimation of the lengths of the formulae, e.g., see Lemmata 2 and 4, or they were not counted at all (see Lemma 3).

The first indices of variables of the form $\hat{a}_1$ and $\hat{b}_1$ are $|\lg 1| + 1 = 1$ in length. The second subscripts of these variables have their lengths restricted from above by $E := |\lg(2n+3r+5)| + 1$. The second indices of variables...
\( \hat{y}_0 \) are shorter — they are bounded by \( E_0 := \lfloor \max\{\log n, \log(r)\} \rfloor + 1 \); besides, the subscripts are not included in the record of the tuples of constants. The number \( n = |X| \) will grow bigger than \( r \), and so the inequality \( E, E_0 \leq \lfloor \log n \rfloor + 2 \) holds for the long enough \( X \). Therefore by Lemmata 3 and 4, the quasi-equations and timers of the subformulae \( \chi(0)(\hat{y}_0) \) and \( \Phi^{(0)}(P)(\hat{a}_1, \hat{b}_1) \) from (1), in which the tuples \( \hat{u}(\beta) \) are not included for \( \beta = R, L \), are not greater than \( D_3 \cdot n \cdot \lfloor \log n \rfloor + 2 \) in length; the clauses and timers comprising the \( \hat{u}(\beta) \) have a length not more than \( D_4 \cdot \lfloor n \cdot (\log n + 2) \rfloor \) for the suitable constants \( D_3 \) and \( D_4 \). By Lemma 4 we have \( |\varphi(k)| \leq D_5 \cdot (n \cdot (\log n + 2))^2 \), but with another constant \( D_5 \) and for the long enough \( X \).

The system of equalities, which are under the "big" conjunction in (1), is \( \mathcal{O}(n \cdot (n \cdot (\log n + 2))) \) in length; and the quantifier prefix, which is situated before this conjunction, has approximately the same length. It is easy to notice that an inequality \( (\log n + 2)^2 \leq n^\varepsilon \) holds for all \( \varepsilon > 0 \) and the big enough \( n \). It follows from this and Lemma 5(ii) that \( |\Omega(X, P)| \leq D \cdot |P| \cdot |X|^2 + \varepsilon \) for some constant \( D \).

**Corollary 4.** There is a constant \( D_\varepsilon \) such that the inequality \( |\Omega(X, P)| \leq D_\varepsilon \cdot |P| \cdot |X|^2 + \varepsilon \) holds for all \( X \) and \( P \).

**Proof.** For each given \( \varepsilon \), there exists only a finite number of the strings \( X \), for which the inequality from the statement of the lemma can be violated. Therefore the ratio \( \lceil |\Omega(X, P)|/(|P| \cdot |X|^{2+\varepsilon}) \rceil \) attains its maximum for these \( X \). Clearly, it is fit for our \( D_\varepsilon \). \( \square \)

**Corollary 5.** There exists a polynomial \( g \) such that for all \( X \) and \( P \) the construction time of the sentence \( \Omega(X, P) \) is not greater than \( g(|X| + |P|) \).

**Proof.** We will at first estimate the time needed for a multi-tape Turing machine \( P_1 \) to write down the formula \( \Omega(X, P) \). The running alphabet of this machine contains all symbols of natural language (see Subsection 3.1).

Let the input tape of the machine comprises a string \( X \) and a program \( P \), and \( |A| \) be a quantity of different symbols in the record of the \( X \) and the \( P \). The machine \( P_1 \) can determine a length \( n \) of the input \( X \), a maximal number \( U \) of internal states in \( P \), and a size of the \( |A| \) during one passage along its input tape. The calculation of the values of \( r \leq \log_2(U + 1 + |A|) \) and the decimal notation of it and \( n \) takes a time bounded by a polynomial of the \( |P| \) and \( |X| \). Further, the \( P_1 \) moves again along the record of the \( X \) and \( P \) and writes the formula \( \chi(0)(\hat{y}_0) \) at first, after that it writes \( \Phi^{(0)}(P)(\hat{a}_1, \hat{b}_1) \),

28
and finally, it designs $\Omega(X, P)$. It is clear that this process takes the time, which is no greater than the value of $p(|X| + |P|)$ for some polynomial $p(y)$.

The single-tape variant $P_2$ of the machine $P_1$ will do the same actions in the time equal to $g(|X| + |P|)$, which is of the form of $O((p(|X| + |P|))^2)$. \[1\] \[2\]

### 7 The complexity of the theory of a single equivalence relation

Let $\mathfrak{A}$ be a class of the algebraic systems, whose signature (or underlying language) $\sigma$ 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. We denote these relations by the same symbol.

**Definition 3.** Let us assume that there exists a $\sim$-nontrivial system $E$ in a class $\mathfrak{A}$, namely, such structure that contains at least two $\sim$-nonequivalent elements. Then the class $\mathfrak{A}$ is also termed $\sim$-nontrivial. A theory $T$ is named $\sim$-nontrivial if it has a $\sim$-nontrivial model. When $\sim$ is either the equality relation or there is such formula $N(x, y)$ of the signature $\sigma$ that the sentence $\exists x, yN(x, y)$ is consistent with the theory $Th(\mathfrak{A})$ (or $T$, or belongs to $Th(E)$), and the sense of this formula is that the elements $x$ and $y$ are not equal, then we will replace the term ”$\sim$-nontrivial” with ”equational-nontrivial”.

**Theorem 2.** Let $E$, $\mathfrak{A}$, and $T$ accordingly be a $\sim$-nontrivial system, class, and theory of the signature $\sigma$, in particular, they may be equational-nontrivial. Then there is an algorithm such that for every program $P$ and any input string $X$, it builds the sentence $\Omega(T)(X, P)$ of the signature $\sigma$, where $T \in \{Th(E), Th(\mathfrak{A}), T\}$; this formula possesses the properties (i) and (ii) of the word $S(P, X)$ from the statement of Proposition \[4\] where $L = T$; $F(|X|) = \exp(|X|)$. Moreover, for each $\varepsilon > 0$, there exists a constant $E_{T, \sigma}$ such that the inequality $|\Omega(T)(X, P)| \leq E_{T, \sigma} \cdot |P| \cdot |X|^{2+\varepsilon}$ holds true for any long enough $X$.

**Proof.** At first, for given $X$ and $P$, we write a simulating sentence $\Omega(X, P)$ of the theory of the Boolean algebra $B$ in the signature $\langle \cap, \cup, C, 0, 1 \rangle$ with the equality symbol $\approx$, applying Theorem \[4\]. Then, we will transform it in the required formulae $\Omega(T)(X, P)$ within polynomial time.
For the sake of simplicity of denotations, we assume that the \( \lnot \) nontrivial structure \( \mathcal{E} \) is a model for the theory \( \mathcal{T} \), belongs to the class \( \mathfrak{R} \), and has the signature \( \sigma \).

Let \( \varphi \) be a sentence of Boolean signature. We will construct the closed formulae \( \varphi^{(2,j)} \) so that \( \mathcal{B} \models \varphi \Leftrightarrow \mathcal{E} \models \varphi^{(2,j)} \), where \( j \) can be 0, 1, or 2 depending on the signature \( \sigma \).

We consider the case, when the \( \sigma \) contains the equivalence symbol \( \lnot \) and the two constant symbols \( c_0 \) and \( c_1 \) such that \( \mathcal{E} \models \lnot c_0 \lnot c_1 \). In the first stage, we accordingly replace each occurrence of the subformulae of the kind \( \lnot \exists y \psi; \forall x \psi; t \approx s \) with the formulae \( \exists y((y \lnot c_0 \lor y \lnot c_1) \land \psi); \forall x((x \lnot c_0 \lor x \lnot c_1) \to \psi); t \approx s \), where \( t \) and \( s \) are the terms.

We carry out the second stage's transformations during several passages until the formula ceases to change. In this stage, a) we replace the subformulae of the kind \( C(t) \lnot s \) and \( t \lnot C(s) \) with the formula \( \lnot t \lnot s \); if a term \( u \) is not the constant 0 or 1, then we replace b) the subformulae of the kind \( t_1 \cup t_2 \cup \ldots \cup t_s \lnot u \) and \( u \lnot t_1 \cup t_2 \cup \ldots \cup t_s \) with the formula \( [(t_1 \lnot c_1 \lor t_2 \lnot c_1 \lor \ldots \lor t_s \lnot c_1) \to u \lnot c_1] \land [(t_1 \lnot c_0 \land t_2 \lnot c_0 \land \ldots \land t_s \lnot c_0) \to u \lnot c_0] \); and c) the subformulae \( t_1 \cap t_2 \cap \ldots \cap t_s \lnot u \) and \( u \lnot t_1 \cap t_2 \cap \ldots \cap t_s \) with the formula \( [(t_1 \lnot c_0 \lor t_2 \lnot c_0 \lor \ldots \lor t_s \lnot c_0) \to u \lnot c_0] \land [(t_1 \lnot c_1 \lor t_2 \lnot c_1 \lor \ldots \lor t_s \lnot c_1) \to u \lnot c_1] \). We complete the second stage by replacing the constants 0 and 1 with the constants \( c_0 \) and \( c_1 \), respectively.

After the second stage, the resulting record can contain symbols \( \cup, \cap \) of the signature of Boolean algebras. In the third stage, we replace accordingly each occurrence of the subformulae of the kind \( t_1 \cup t_2 \cup \ldots \cup t_s \lnot c_1 \), \( t_1 \cap t_2 \cap \ldots \cap t_s \lnot c_1 \), \( t_1 \cup t_2 \cup \ldots \cup t_s \lnot c_0 \), \( t_1 \cap t_2 \cap \ldots \cap t_s \lnot c_0 \) with the formulae \( t_1 \lnot c_1 \lor t_2 \lnot c_1 \lor \ldots \lor t_s \lnot c_1 \), \( t_1 \lnot c_1 \land t_2 \lnot c_1 \land \ldots \land t_s \lnot c_1 \), \( t_1 \lnot c_0 \land t_2 \lnot c_0 \land \ldots \land t_s \lnot c_0 \).

We execute these transformations as long as the record contains at least one symbol of the signature of Boolean algebras. The number of such symbols is decreased at least by one on every passage for the second and third stage, and the first stage can be realized on the only passage. So we need at most \( n \) passages, where \( n \) is a length of the sentence \( \varphi \). The length of the whole record grows linearly on each pass, since the transformation of the kind b) or c) of the second stage is longest, but it increases the length no more than in five times (for \( s = 2 \)).

Nevertheless, the length of the resulting record \( \varphi^{(2,0)} \models \varphi^{(2,0)}_{c_0,c_1} \) can increase non-linearly in common case. For instance, if \( \varphi \) contains an atomic
formula of the kind
\[ \bigcup_{i,j} \bigcap_{k} \{ \bigcup(\ldots) \} \approx u, \]

where the number of the alternations of the "big" conjunctions and disjunctions depends on the \( n \).

However, there are not such subformulae in the sentence \( \Omega(X, P) \) simulating for the theory of algebra \( B \). Indeed, in accordance with its definition, the conversion of the subformulae of the kind \( \hat{x}_i \approx \hat{u}(\beta) \) (this is the system of equalities) and \( \neg \hat{\omega} \approx \hat{u}(\beta) \) (this is the disjunction of inequalities) make the most increase if \( \beta \in \{R, L\} \), because they comprise the atomic formulae of the form \( x_{t,j} \approx u_j \oplus u_{j+1} \oplus \ldots \oplus u_n \) and \( \neg w_j \approx u_j \oplus u_{j+1} \oplus \ldots \oplus u_n \), where \( u_k \) is either \( u_k \) for \( \beta = R \) or \( Cu_k \) for \( \beta = L \) — see Subsection 3.3 We recall that these subformulae are \( x_{t,j} \approx [u_j \cap C(u_{j+1} \cap \ldots \cap u_n)] \cup [Cu_j \cap u_{j+1} \cap \ldots \cap u_n] \) and \( \neg w_j \approx [u_j \cap C(u_{j+1} \cap \ldots \cap u_n)] \cup [Cu_j \cap u_{j+1} \cap \ldots \cap u_n] \) by our denotation. So, we need to perform only the three transformations of the kind \( b) \) and \( c) \) in order to convert the sentence \( \Omega(X, P) \) into the \( \tau(X, P)^{(2,0)} \). Therefore the estimation \( |\Omega(X, P)^{(2,0)}| \leq D_0|\Omega(X, P)| \) is valid for appropriate constant \( D_0 \). Since one can execute every passage of any stage within \( O(|\Omega(X, P)|^2) \) steps, the entire transformation takes the polynomial time.

Let us suppose that the signature of the structure \( E \) has no the constant symbols. Then we replace the constants \( c_0 \) and \( c_1 \) in the formula \( \varphi^{(2,0)} \models \varphi_{\alpha_0, \alpha_1} \) with the new variables \( a \) and \( b \), respectively. We obtain the formula \( \varphi^{(2,1)} \models \exists a, b[\neg a \sim b \& \varphi_{\alpha, \beta}] \). It is clear that \( |\varphi^{(2,1)}| \leq 2|\varphi^{(2,0)}| \) for \( |\varphi^{(2,0)}| \geq 11 \), and so \( |\Omega(X, P)^{(2,1)}| \leq D_1|\Omega(X, P)| \) for appropriate constant \( D_1 \).

Finally, when the signature \( \sigma \) does not contain the equivalence symbol, but there exists a formula \( N(x, y) \), which asserts that the elements \( x \) and \( y \) is not equal, then we replace every occurrence of the atomic subformula of the kind \( t \sim s \) in the \( \varphi_{\alpha, \beta} \) with the formula \( \neg N(t, s) \) and add the prefix: \( \varphi^{(2,2)} \models \exists a, b[N(a, b) \& \varphi_{\alpha, \beta}] \). It is obvious that \( |\varphi^{(2,2)}| \leq |N(x, y)| \cdot |\varphi^{2,1}| \), hence \( |\Omega(X, P)^{(2,2)}| \leq D_2|\Omega(X, P)| \) for some constant \( D_2 \).

One can easily prove by induction on the complexity of the formulae that the condition \( B \models \varphi \) is equally matched to one of the following conditions (depending on the signature \( \sigma \)): either \( \mathcal{E} \models \varphi^{(2,0)} \), or \( \mathcal{E} \models \varphi^{(2,1)} \), or \( \mathcal{E} \models \varphi^{(2,2)} \). It is clear that if \( \mathcal{R} \) and \( \mathcal{T} \) are the equational-nontrivial class and theory respectively, then the condition \( B \models \varphi \) is also equally matched to the conditions \( Th(\mathcal{R}) \vdash \forall a, b(N(a, b) \rightarrow \varphi_{\alpha, \beta}^{N}) \) and \( \mathcal{T} \vdash \forall a, b(N(a, b) \rightarrow \varphi_{\alpha, \beta}^{N}) \). \( \square \)
Corollary 6. The recognition complexity of each \(\sim\)-nontrivial decidable theory \(\mathcal{T}\), in particular, equational-nontrivial, has the non-polynomial lower bound, more precisely

\[
\mathcal{T} \notin DTIME(\exp(D_{T,\sigma} \cdot n^\delta)), \quad \text{where} \quad \delta = (2 + \varepsilon)^{-1}, \quad D_{T,\sigma} = (E_{T,\sigma})^{-\delta}.
\]

Proof. It immediately follows from the theorem and Corollary 1. \(\square\)

8 Results and Discussion

Let us notice that nearly all of the decidable theories mentioned in the surveys [6, 15] are nontrivial regarding equality or equivalence. So, if we regard ”the polynomial algorithm” as a synonym for ”the fast-acting algorithm”, then the quickly decidable theories are almost completely absent. Furthermore, the examples given in the introduction and [4, 7, 8, 12, 13, 14, 15, 16, 17, 19] show that the complexity of the recognition procedures can be perfectly enormous for many natural, and seemingly, relatively simple theories.

It seems plausible that the estimation obtained in Corollary 2 is precise enough. One can substantiate this assertion, if firstly, to find the upper bound on the recognition complexity of theory \(Th(B)\) by the multitape Turing machines; secondly, to obtain the lower bound for this complexity for the same machines. The author suspects that the inequalities from Item (iii) of the main theorem are valid as well for the \(k\)-tape machines, but the constant \(D\) must be about in \(k\) times bigger at that.

Let us point out that the number of the alternation of quantifiers depends on the input length in the modeling formula \(\Omega(X, P)\). Therefore this sentence does not belong to any class of polynomial hierarchy. However, if one can build the \(\Omega(X, P)\) belonging to \(\Delta^p_k\) for some \(k > 1\), then the class \(P\) will be different from this \(\Delta^p_k\), hence the \(P\) will be not equal to the \(NP\) [14].

8.1 The totality and locality of the simulating methods

The method of Cook’s formulae has arisen for the modeling of the nondeterministic Turing machine actions within polynomial time, and the construction of Stockmeyer and Meyer is also applicable for the same simulation in polynomial space, provided that the running time of the machine is exponential. This is a significant advantage of these techniques.
However, our method of modeling by means of formulae is ineligible for nondeterministic machines. More precisely, such modeling formula must be exponential in length, when the machine runs in exponential space within the exponential time. Unfortunately, the corresponding example is too cumbersome for this paper. This example rests on that simple fact that if we set the values of the basic color \( t \) variables, then we can ”see” only at most two the tape squares (the \( \widehat{x}_t \)th and maybe \( \widehat{z}_t \)th) when we are situated within the framework of our approach — see the proof of Proposition 2. So our simulating method is strictly local, pointwise. At the same time, the techniques of Cook and Stockmeyer and Meyer are total, since they allow to ”see” all of the tape cells simultaneously at any instant. These methods are also complete in the sense that we will point in the next subsection.

The author is sure that the technique of the direct encoding of machines continues to be a potent tool for investigating the computational complexity of theories, despite the emergence of other powerful approaches for obtaining the lower bounds on this complexity such as the Compton and Henson method [4] or the method of the bounded concatenations of Fleischmann, Mahr, and Siefkes [8].

Nevertheless, the author agrees with the opinion that the coding of the machine computations into the models of the theory being studied is the very difficult task in many cases. Such coding is partly like to the modeling of the machine actions with the aid the defining relations, when one wants to prove the insolubility of some algorithmically problem for the finitely presented algebraical structures of given variety (see, for instance, [3, 10]).

In both cases, we have the strong restrictions, which are dictated by the necessity to be within the framework of the given signature or variety. But the case of the algorithmic problem for the finitely presented structures is, perhaps, somewhat easier than the simulation of computations by means of the formulae of a certain theory. In the first case, we can apply the suitable words consisting of the generators of the algebraical system for the description of tape configurations or their parts. The value of these words can change depending on the defining relations and the identity of variety. However, these changes have the local character relatively of the entire structure; whereas the variables can take on any values inside the system when we make a simulation in the second case.

The task becomes slightly easier if there are some constants in the theory signature. Just for this reason, we work with the Boolean algebra having two elements, but not with the language \( TQBF \) consisting of the true quantified
Boolean formulae.

Note also, that the actions simulation of the computational mechanisms, which is realized in [3] and [10] (these devices are the Minsky machines in the former, and they are the Minsky operators algorithms in the latter), is total. On the other hand, this modeling is somewhat like the Compton and Henson method too. Indeed, in all of these cases, the coding of computations is done once and for all. In [3], this is made for Turing machines in proving the inseparability results; then, the authors transfer the obtained lower bounds from one theory to another, using interpretations. Both in [3] and in [10], such simulation is made in proving the insolubility of the words problem for the appropriate module over a certain integral domain; afterward, this module is embedded (isomorphically in the first article and homomorphically in the second) in the solvable group under construction.

8.2 Complete simultaneous and conventionally consistent modeling

Let us investigate the modernization of the Stockmeyer and Meyer method that arose in this paper.

8.2.1 A modernization to some extent is present

When we were saying that the formula \( A_{0,m} \) (from the proof of Theorem 4.3 in [18]) is the analog of the \( \Omega^{(0)}(X,P) \) in Remark 4, we had in mind that the \( A_{0,m}(\hat{U}, \hat{V}) \) consists of the conjunction of the subformulae \( u_1 \ldots u_m, v_1 \ldots v_m \), which describe the adjacent configurations, and also of the subformula that describes the transformation from the former to the latter. One can consider that this transfer formula has the kind \( A_t \land B_t \land C_t \land D_t \land A_{t+1} \land B_{t+1} \land C_{t+1} \land D_{t+1} \land E_{t,t+1} \), where the \( A_t, A_{t+1}, B_t, B_{t+1}, C_t, C_{t+1}, D_t, D_{t+1}, \) and \( E_{t,t+1} \) are the subformulae of the formulae \( A, B, C, D, \) and \( E \) respectively from the proof of Theorem 10.3 in [1] and are obtained from them by means the restriction of the last formulae on the fixed value of the parameter \( t \). One can say that the complete simultaneous modeling of actions has been applied there. So one can regard the \( A_{0,m} \) in the kind

\[
configuration(t) \land configuration(t+1) \land step(t+1).
\]

But the formula \( \Omega^{(0)}(X,P)(\hat{g}_t, \hat{g}_{t+1}) \) is constructed in another way. It asserts that if the descriptions of the \( t \)th step’s configuration (the formula
\(\Psi K(t)(\hat{y}_t)\) and of the step \(t + 1\) (the \(\Phi^{(0)}(P)(\hat{y}_t, \hat{y}_{t+1})\) are correct, then the configuration, which appeared after this step, will be adequately described as well (by the \(\Psi K(t+1)(\hat{y}_{t+1})\)). We call this approach as a conventionally consistent modeling of actions, i.e., the \(\Omega^{(0)}(X, P)\) has such structure:

\[
\text{configuration}(t) \& \text{step}(t + 1) \rightarrow \text{configuration}(t + 1).
\]

Thus, the designs of the formulae \(\Omega^{(0)}(X, P)(\hat{y}_t, \hat{y}_{t+1})\) and \(A_{0,m}(\hat{U}, \hat{V})\) are essentially different, if even one does not take into consideration the presence of the inner quantifiers in the former. Furthermore, their free variables "demand" the quantifiers of the various kind in order the formulae become true. Seemingly, just this difference in the external quantifiers dictates the diversity, above mentioned, in the internal structure of these formulae.

The additional argument for this conclusion is that the conventionally consistent modeling is also used in [3] and [10]. Recall in this connection that the investigation of the finitely presented algebraic system, which is given with the aid of the generators \(g_1, \ldots, g_k\) and the defining relations \(R_1(g_1, \ldots, g_k), \ldots, R_m(g_1, \ldots, g_k)\), is equivalent (in many respects) to the study of the formulae of the kind

\[
\forall g_1 \ldots \forall g_k[(R_1(g_1, \ldots, g_k) \& \ldots \& R_m(g_1, \ldots, g_k)) \rightarrow S(g_1, \ldots, g_k)].
\]

Furthermore, we can see in Proposition 2(i) that the \(\Omega^{(0)}(X, P)\) can completely model too, but existential quantifiers are applied at that.

Let us notice that the author applies an internal compression of simulating formulae when defining the formulae \(\Phi^{(s+1)}(P)(\hat{y}_t, \hat{y}_{t+e(s+1)})\) in Subsection 4.3.2. The compression is internal, since figuratively speaking, the "compressing spring", which is the subformula \(\exists \hat{v} \forall \hat{a} \forall \hat{b}[(\hat{y}_t \approx \hat{a} \land \hat{v} \approx \hat{b}) \lor (\hat{v} \approx \hat{a} \land \hat{b} \approx \hat{y}_{t+e(s+1)})]\), is inserted inside this \(\Phi^{(s+1)}(P)\), i.e., this "spring" is located deep inside the modeling formula \(\Omega^{(s+1)}(X, P)\). From this standpoint, Stockmeyer and Meyer use the method of an "external compression", as their simulating formula \(A_{s+1,m}\) contains similar "compressing spring" at the its "surface", if it is considered as the analog of the \(\Omega^{(0)}(X, P)\).

### 8.2.2 Seemingly, there is almost no modernization

However, the author is inclined to believe nevertheless that the analog of the formula \(A_{0,m}(\hat{U}, \hat{V})\) is the \(\Phi^{(0)}(P)(\hat{y}_t, \hat{y}_{t+1})\). In other words, the former corresponds to the \(A_t \& B_t \& C_t \& D_t \& A_{t+1} \& B_{t+1} \& C_{t+1} \& D_{t+1} \& E_{t,t+1}\) (see the first
paragraph of the previous subsubsection), i.e., this formula simply describes
the transformation of one configuration to another, but it does not contain
the descriptions of these configurations (the formulae $u_1 \ldots u_m$ and $v_1 \ldots v_m$) in explicit form.

This is supported by the fact that the descriptions of intermediate con-
figurations do not enter into the formula from the proof of Cook’s theorem
(Theorem 10.3) in [1]. Their inclusion will lead to an increase in the length
of the entire modeling formula almost in $2p(n)$ times in the case of Cook’s
theorem, and this increase will be linear in the proof of Theorem 4.3 in [18].
Certainly, that is not essential for the proof of these theorems, but this inclu-
sion makes the simulating formulae be excessively cumbersome. And what
is more, there is no need for this.

Indeed, one can easily prove by induction in this case that if

$$B_{s,m}(\tilde{U}, \tilde{V}) \equiv u_1 \ldots u_m \& A_{s,m} \& v_1 \ldots v_m,$$

then $\exists \tilde{U} \exists \tilde{V} B_{s,m}(\tilde{U}, \tilde{V})$ is true if and only if the configuration encoded by $v_1 \ldots v_m$ follows from the configuration that corresponds to $u_1 \ldots u_m$ in at
most $\exp(s)$ steps of the $P$. So this $B_{s,m}(\tilde{U}, \tilde{V})$ is the simulating formula in
[18], and it is analog of our $\Omega(s)(X, P)(\tilde{y}_t, \tilde{y}_{t+e(s)})$.

Thus, the Stockmeyer and Meyer method uses, seemingly, the internal
compression of modeling formulae as our one. Nevertheless, this simulating
method does not cease to be complete and simultaneous.

8.3 Open problems

It is well known that the theory of two equivalence relations is not decidable,
but the theory of one such relation $\sim$ is decidable [6]. Now it turns out
according to Corollary [8] that although it is decidable, but for a very long
time.

What will happen if we add some unary predicates or functions to the
signature with the only equivalence symbol $\sim$ so that the resulting theory
remains decidable? Will it be possible to find such functions and/or pre-
dicates in order that the recognition complexity ”smoothly” increases? We can
formulate this in a more precise way.

Problem 1. Let $\sigma_0, \sigma_1, \ldots$ be a sequence of signatures such that $\sigma_0 \supseteq \{\sim\}$
or $\sigma_0 \supseteq \{\approx\}$ and $\sigma_i \subset \sigma_{i+1}$ for each natural $i$. Does there exist a sequence
of the algebraical structures $\mathfrak{M}_0, \mathfrak{M}_1, \ldots$ such that their signature accordingly are $\sigma_0, \sigma_1, \ldots$ and

$$Th(\mathfrak{M}_j) \in DTIME(\exp_{j+2}(n)) \setminus DTIME(\exp_{j+1}(n))?$$

Recall that $Th(\mathfrak{M})$ denotes the first-order theory of the system $\mathfrak{M}$. It is possible that there already is a candidate for the like sequence of the higher-order theories with the ”smoothly” increasing recognition complexity.

**Problem 2.** Let $\Omega^{(k)}$ be a fragment of the type theory $\Omega$ from [19], which is obtained with the aid the restriction of the types of variables by level $k$. Can one point out for each natural $k$ such a number $s$ that $\Omega^{(k)} \in DTIME(\exp_{k+s+1}(n)) \setminus DTIME(\exp_{k+s}(n))$?

**Problem 3.** It seems quite plausible that the theory of finite Boolean algebras has a double exponential as the lower bound on the complexity of recognition.

**Problem 4.** Let $F(n)$ be a limit upper bound for all polynomials (see Definition 2). What algebraic and/or model-theoretic properties must be possessed an algebraical structure $\mathfrak{A}$ in order that $Th(\mathfrak{A}) \in DTIME(F(n))$ or $Th(\mathfrak{A}) \notin DTIME(F(n))$ holds true?

**References**

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

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

[3] *Baumslag G., Gildenhuys D., Strebel R.* Algorithmically insoluble problems about finitely presented solvable groups, Lie and associative algebras I // J. Pure and Applied Algebra. — 1986. — Vol. 39, No. 1–2. — P. 53–94.

[4] *Compton K.J. and Henson C.W.* A uniform method for proving lower bounds on the computational complexity of logical theories // Ann. Pure Appl. Logic. — 1990. — Vol. 48 — P. 1–79.

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

*We quote from this book and [6] and from review [6] by their versions in Russian.*
[6] Ershov Y.L., Lavrov I.A., Taimanov A.D., and Taitslin M.A. Elementary theories // Russian Math, Surveys. — 1965. — Vol. 20. — P. 35–100 (English translation).

[7] Fisher M.J. and Rabin M.O. Super exponential complexity of Presburger’s arithmetic // SIAM, AMS Proceedings. — 1974. Vol. 7. — P. 27–41.

[8] Fleischmann K., Mahr B., and Siefkes D. Bounded concatenation theory as a uniform method for proving lower complexity bounds // in: R. Gandy, M. Hyland, eds. — North-Holland, Amsterdam: Logic Colloquium. — 1977. Vol. 76. — P. 471–490.

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

[10] Latkin I.V. The Occurrence Problem for Subvarieties of the Variety $N_2A$ Revisited // Algebra and Logic. — 2001. — Vol. 40, No 5 — P. 327–333.

[11] Malcev A.I. Algorithms and recursive functions — Groningen: Wolters-Noordhoff Pub. Co. — 1970.

[12] Meyer A.R. Weak monadic second order theory of successor is not elementary-recursive // Technical Report (Memo 38) Massachusetts Institute of Technology Cambridge, MA, USA, 1973.

[13] Meyer A.R. The inherent complexity of the theories of ordered sets // in: International Congress of Mathematicians. Vancouver, 1974. — also in Canadian Math. Congress, 1975. — P. 477–482.

[14] Meyer A.R. and Stockmeyer L.J. The equivalence problem for regular expressions with squaring requires exponential space // in: Proc. Thirteenth Annual IEEE Symposium on Switching and Automata theory, 1972. — P. 125–129.

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

[16] Robertson E.L. Structure of complexity in the weakmonadic second-order theories of the natural numbers // Research Report CS-73-31, Dept. of Applied Analysis and Computer Science, kiv. of Waterloo, Dec., 1973; also Proc. 6th ACM Symp. on Theory of Computing, 1974. — P. 161–171.

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

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

[19] Vorobyov S. The most nonelementary theory // Information and Computation, 2004. — Vol. 190. — P. 196–219.