ISSN 0146-4116, Automatic Control and Computer Sciences, 2019, Vol. 53, No. 7, pp. 663–675. © Allerton Press, Inc., 2019.
Russian Text © The Author(s), 2018, published in Modelirovanie i Analiz Informatsionnykh Sistem, 2018, No. 5, pp. 506–524.

On the Expressive Power of Some Extensions of Linear Temporal Logic

A. R. Gnatenko[a], * and V. A. Zakharov[b], **

[a] *Lomonosov Moscow State University, Moscow, 119991 Russia*
[b] *National Research University, Higher School of Economics (HSE), Moscow, 101000 Russia*
**e-mail: gnatenko.cmc@gmail.com*
***e-mail: zakh@cs.msu.ru*

Received September 10, 2018

Abstract—One of the most simple models of computation suitable for representation of reactive systems behavior is a finite state transducer that operates on an input alphabet of control signals and an output alphabet of basic actions. The behavior of such a reactive system displays itself in the correspondence between the flow of control signals and sequence of basic actions performed by the system. We believe that the behavior of this kind requires more complex and expressive means of specifications than the conventional linear temporal logic (LTL). In this paper, we define a new language of formal specifications $\mathcal{LP}$-LTL, which is an extension of LTL, specifically intended for describing the properties of transducer computations. In this extension, the temporal operators are parameterized by sets of words (languages) which describe flows of control signals that impact on a reactive system. Basic predicates in the $\mathcal{LP}$-LTL formulas are also defined by the languages in the alphabet of basic actions; these languages describe the expected response of the system to the environmental influences. In our earlier papers, we considered a verification problem for finite state transducers with regard to specifications represented by the $\mathcal{LP}$-LTL and $\mathcal{LP}$-CTL formulas. It was shown that the problem is algorithmically solvable for both logics. The aim of this paper is to estimate the expressive power of $\mathcal{LP}$-LTL by comparing it with some well-known logics widely used in computer science for specification of reactive systems. Two fragments were isolated in the $\mathcal{LP}$-LTL: $\mathcal{LP}$-1-LTL and $\mathcal{LP}$-n-LTL. We discovered that the specification language of $\mathcal{LP}$-1-LTL is more expressive than LTL, while the $\mathcal{LP}$-n-LTL fragment has the same expressive power as the monadic second order logic S1S.

Keywords: temporal logics, expressive power, specification, verification, Büchi automata, infinite words

DOI: 10.3103/S014641161907006X

1. INTRODUCTION

Transducers are among the earliest models of computation; they are used in computer science, software engineering, microelectronics, and linguistics. In this paper, we will use them as a formal model of reactive information systems. At each step of the computation, a transducer receives one of the letters of the input alphabet and yields a sequence of letters (word) of the output alphabet. The letters of the input alphabet correspond to control signals from the external environment, and the letters of the output alphabet correspond to basic actions performed by the reactive system. On the set of basic actions performed by the transducer, it is possible to introduce a composition operation, thus forming a semigroup of automaton actions. The behavior of the system in this case is characterized by the relationship of converting the flow of control signals into actions performed by the system. A model of sequential reactive systems of this type was proposed and investigated in [6, 21–23].

However, verifying the correctness of the behavior of reactive systems that are modeled by finite-state transducers over semigroups requires a language of formal specifications that is adequate to this model. Our proposed approach to constructing such a language is based on the following considerations. A typical requirement for the correct behavior of a reactive system is that for each input word (flow of control signals) that falls under some given pattern, a response is generated that also falls under a given pattern. To express such requirements, not only temporal operators will be needed, with the help of which one can describe the sequence of events (control signals and actions), but also means for describing and account-

ing for the specified patterns. The patterns of the flow of control signals can be defined by any means for describing formal languages: automata, formal grammars, regular expressions, etc. To describe the response of the transducer operating on some given semigroup of elementary actions, one can use any predicates defined in this semigroup. In the case of a free semigroup, such predicates are, in essence, the sets of words (languages) over the alphabet of elementary actions. They can also be specified using any means for defining formal languages. In this case, the behavior properties of reactive systems can be described, for example, using temporal formulas of the form $\mathbf{G}_L P$, which mean that for each input word w that belongs to the language L, the output word h, which is generated by a transducer as a response, belongs to the language P.

This approach has two advantages. Firstly, it allows one to explicitly express the correspondences between the input and output words of transducers and thus describe the typical requirements for the correct behavior of reactive systems. In addition to that, it allows adapting model checking techniques suitable for traditional temporal logics (see [2]). In [11], this idea was implemented for LTL-based specification language $\mathcal{LP}$-LTL, and in [7] it was extended to the specification language $\mathcal{LP}$-CTL, which extends the computation tree logic (CTL).

The purpose of this study is to evaluate the expressive power of the new temporal logic $\mathcal{LP}$-LTL by comparing it with other applied logics used for verification of programs. Such a comparison is complicated by the fact that the semantics of $\mathcal{LP}$-LTL and the semantics of the most well-known temporal logics are defined on different structures. To overcome this difficulty, in the logic $\mathcal{LP}$-LTL we have isolated two fragments, $\mathcal{LP}$-1-LTL and $\mathcal{LP}$-n-LTL, the semantics of which can be adapted to infinite words (omega-words); this makes it possible to compare these fragments with the linear temporal logic LTL and monadic second-order logic S1S. As a result, it has been found that $\mathcal{LP}$-1-LTL is strictly more expressive than LTL, while $\mathcal{LP}$-n-LTL and S1S have the same expressive power.

This paper is organized as follows. Section 2 defines the model of a finite-state sequential transducer. The next section describes the syntax and semantics of the temporal logic $\mathcal{LP}$-LTL. In Section 4, the fragments $\mathcal{LP}$-1-LTL and $\mathcal{LP}$-n-LTL of the logic $\mathcal{LP}$-LTL are distinguished, the semantics of which can be defined on omega-words. The next two sections present the main results of this paper. In conclusion, we briefly compare our extension of LTL with other extensions of the same logic and outline the directions for further studies of the properties of $\mathcal{LP}$-LTL.

2. MODELING OF REACTIVE SYSTEMS BY TRANSDUCERS

A reactive system performs the computation by receiving input signals from the environment and performing certain basic actions in response. Let us consider a finite set $\mathcal{C}$ of *input signals* and a finite set $\mathcal{A}$ of *basic actions*. Finite words in the alphabet $\mathcal{C}$ will be called *flows of signals*; finite words in the alphabet $\mathcal{A}$ will be called *composite actions* that can be considered as sequential compositions of basic actions. Let us denote the sets of all possible flows of signals and all possible composite actions by $\mathcal{C}^*$ and $\mathcal{A}^*$, respectively. We use the standard notation adopted in the theory of formal languages: uv indicates the concatenation of the words u and v; the symbol ε denotes an empty word.

Definition 1. A *finite-state transducer* over the alphabets $\mathcal{C}$ and $\mathcal{A}$ is a quintuple $\Pi = (Q, \mathcal{C}, \mathcal{A}, q_{\text{init}}, T)$, where (1) Q is a set of control states; (2) $q_{\text{init}} \in Q$ is a initial state; and (3) $T \subseteq Q \times \mathcal{C} \times Q \times \mathcal{A}^*$ is a transition relation.

We will assume that the transition relation T is total; i.e., for each state q and control signal c there are such a state q' and action h, for which $(q, c, q', h) \in T$ holds. A quadruple $(q', c, q'', h) \in T$ is called a *transition*: being in a state q' and receiving a signal c, the automaton goes into a state q'' and performs a composite action h. For clarity, we will denote the transition by $q' \xrightarrow{c,h} q''$.

To describe the behavior of a transducer, we need the concepts of a run and a trajectory. A *run* of a transducer Π is an infinite sequence of transitions: $\rho = q_1 \xrightarrow{c_1,h_1} q_2 \xrightarrow{c_2,h_2} \cdots \xrightarrow{c_n,h_n} q_{n+1}, \ldots$ A run that starts in the state q_{init} is called a basic run.

Definition 2. Let s_0 be some composite action, and ρ is the abovementioned run of a transducer Π. A *trajectory* of Π on the run ρ is a pair $tr = (s_0, \alpha)$, where the sequence $\alpha = (c_1, s_1), (c_2, s_2), \ldots, (c_i, s_i), \ldots$, satisfies the condition $s_i = s_{i-1} h_i$ for each natural i.

A trajectory is a possible reaction of the system that previously performed the action s_0 in response to a flow of input signals $w = c_1c_2 \ldots c_i \ldots$. In the case if ρ is a basic run of the transducer, and $s_0 = \varepsilon$, the corresponding trajectory is also called the basic trajectory. The set of all basic trajectories of a transducer Π will be denoted by $Tr(\Pi)$. The notation $tr\big|^i$ is understood as a trajectory, which represents the "tail" of a trajectory tr, i.e., the trajectory $(s_i, \alpha\big|^i)$, where $\alpha\big|^i (c_{i+1}, s_{i+1}), (c_{i+2}, s_{i+2}), \ldots$

3. $\mathcal{LP}$-LTL SPECIFICATION LANGUAGE

To verify the correctness of the behavior of reactive systems, it is necessary to have a formal specification language in which the correctness requirements are written. The behavior of a reactive system is a process that proceeds in time and manifests itself in the correspondence between the flow of control signals arriving at the input of the system and the actions performed by the system. The behavior of this kind is usually specified by means of a certain type of temporal logic. When choosing the appropriate temporal logic for a formal description of the transducer's behavior, one should take into account the following two distinctive features of the proposed model of reactive systems:

1. The result of the transducer operation is a sequence of actions from the set $\mathcal{A}$ performed by it; therefore, atomic formulas (basic predicates) must be interpreted on the set $\mathcal{A}^*$.

2. The behavior of the transducer depends not only on time, but also on the flow of input control signals; therefore, the temporal operators should be *parameterized*, i.e., supplied with a description of those flows of control signals on which the behavior of the transducer is checked.

In order to provide these properties to the language of reactive system behavior specifications, the authors of [11] proposed a new extension of linear temporal logic (LTL). This extension is based on the following idea. Verification of the correctness of transducer's reactions should be carried out on control signal flows. For an outside observer, the behavior of a transducer on a given flow of control signals is completely determined by the trajectories of the transducer's runs on this flow. The events available to the observation are the actions performed by the automaton during the runs. The correctness of the transducer's behavior is manifested in the fact that these events occur in a certain desired order. To describe the order of implementation of the events, one can use LTL formulas in which certain types of events act as elementary statements (basic predicates). At the same time, the sets of control signal flows on which a correctness requirement of the automaton's behavior is checked are indicated as parameters of temporal operators. Thus, a parameterized extension of LTL is formed, in which the satisfiability relation is determined on the trajectories of transducers. A more detailed description of this extension is as follows.

Any set of control signal flows will be considered as a language in the alphabet $\mathcal{C}$. Let us distinguish a certain family of languages $\mathcal{L}$ in the alphabet $\mathcal{C}$ (for example, regular languages) and call any language from this family an *environment behavior pattern*. Behavior patterns are intended to describe the allowable effects of external environment on the reactive system.

The composite actions performed by the system in response to control signals are also words in the alphabet $\mathcal{A}$. Let us select a class of languages $\mathcal{P}$ in the alphabet $\mathcal{A}$ and call any language from this class a *basic predicate*. Each basic predicate can be considered as some type of observed response events of the reactive system in reply to the influence of the external environment. When using environment behavior patterns and basic predicates in logical formulas, we assume that the respective languages are defined in a certain constructive way (for example, using automata, grammars, Turing machines, etc.).

Definition 3. Let there be specified some family $\mathcal{L}$ of environment behavior patterns and a class $\mathcal{P}$ of basic predicates that allow constructive description. The set of $\mathcal{LP}$-LTL formulas is the smallest set of expressions that can be constructed according to the following rules:

1. each basic predicate $P \in \mathcal{P}$ is a formula;

2. if φ_1, φ_2 are formulas, then expressions $\neg\varphi_1$ and $\varphi_1 \wedge \varphi_2$ are formulas;

3. if φ_1 and φ_2 are formulas, $c \in \mathcal{C}$, and $L \in \mathcal{L}$, then the expressions $\mathbf{X}_c\varphi_1, \mathbf{Y}_c\varphi_1, \mathbf{F}_L\varphi_1, \mathbf{G}_L\varphi_1$, and $\varphi_1\mathbf{U}_L\varphi_2$ are formulas.

The $\mathcal{LP}$-LTL semantics is defined by means of satisfiability relation of formulas on interpretations, which are the trajectories of transducers. Let us assume there is a transducer Π and its trajectory $tr = (s_0, \alpha)$, where $\alpha = (c_1, s_1), (c_2, s_2), \ldots, (c_i, s_i), \ldots$ For any formula ψ, the notation $tr \models \psi$ will then mean that the formula ψ is satisfied on the trajectory tr of the automaton Π.

Definition 4. The satisfiability relation $\models$ in $\mathcal{LP}$-LTL will be defined by induction on the depth of the formula:

1. if P is a basic predicate, then $tr \models P \Leftrightarrow s_0 \in P$;

2. $tr \models \neg\varphi \Leftrightarrow$ it is false that $tr \models \varphi$;

3. $tr \models \varphi_1 \wedge \varphi_2 \Leftrightarrow tr \models \varphi_1$ and $tr \models \varphi_2$;

4. $tr \models \mathbf{X}_c\varphi \Leftrightarrow c = c_1$ and $tr^1 \models \varphi$;

5. $tr \models \mathbf{Y}_c\varphi \Leftrightarrow c \neq c_1$ or $tr^1 \models \varphi$;

6. $tr \models \mathbf{F}_L\varphi \Leftrightarrow \exists i \geqslant 0$: $c_1c_2 \ldots c_i \in L$ and $tr^i \models \varphi$;

7. $tr \models \mathbf{G}_L\varphi \Leftrightarrow \forall i \geqslant 0$: if $c_1c_2 \ldots c_i \in L$, then $tr^i \models \varphi$;

8. $tr \models \varphi\mathbf{U}_L\psi \Leftrightarrow \exists i \geqslant 0$: $c_1c_2 \ldots c_i \in L$, such that $tr^i \models \psi$ and $\forall j$, $0 \leqslant j < i$, if $c_1c_2 \ldots c_i \in L$, then $tr^j \models \varphi$.

For the first time, $\mathcal{LP}$-LTL was introduced in [11]. Unlike LTL, in this logic, two different operators, $\mathbf{X}_c$ and $\mathbf{Y}_c$, are necessary to refer to the next time point (NextTime) depending on how mandatory is the appearance of the control signal c (which is used as a parameter) in the trajectory at the current time. The operators $\mathbf{X}_c$ and $\mathbf{Y}_c$ are dual to each other. The operators $\mathbf{F}_L$ and $\mathbf{G}_L$ are dual, as well. Using negation $\neg$ and conjunction $\wedge$, other Boolean connectives can be expressed, such as $\vee, \rightarrow, \equiv$. Just as in the above definition for the operator $\mathbf{U}$ (Until), in $\mathcal{LP}$-LTL one can introduce parameterized analogues of other temporal operators of LTL, e.g. $\mathbf{R}$ (Release) or $\mathbf{W}$ (Weak Until). It is shown in [11] that many useful equivalence relations between LTL formulas, like fixed-point identities, remain valid for parameterized logic.

In the framework of $\mathcal{LP}$-LTL, the verification problem for transducers is set up as follows: for an arbitrary given transducer Π and a formula φ, check whether the relation $tr \models \varphi$ holds for each basic trajectory tr from the set $Tr(\Pi)$. In [11], it is shown that this problem is decidable in double exponential time in the case when the families $\mathcal{L}$ and $\mathcal{P}$ are classes of regular languages, and at the same time, the environment behavior patterns and basic predicates in $\mathcal{LP}$-LTL formulas are described using deterministic finite-state automata.

In [7], the idea of parameterization of temporal operators by environment behavior patterns was extended to the Computation Tree Logic (CTL). In that paper, the syntax and semantics of $\mathcal{LP}$-CTL were defined, and it was shown that the verification problem for transducers with respect to the specifications of their behavior represented by $\mathcal{LP}$-CTL formulas over regular language families $\mathcal{L}$ and $\mathcal{P}$ is decidable in exponential time.

It is possible to propose a more sophisticated interpretation of the actions performed by transducers, rather than the simple construction of output words. For example, these actions can be viewed as elements of a certain semigroup or group with decidable problem of identities (so that the results of actions can be effectively compared). In this case, basic predicates can be specified by special algebraic means that are typical for a particular type of semigroups. The verification problem for transducers can then become significantly more complicated: in [6], it was shown, for example, that the verification problem for automata operating on a free commutative semigroup is undecidable.

4. TWO FRAGMENTS OF $\mathcal{LP}$-LTL

There are numerous logics that are used to describe the behavior of computing systems. In these logics, each formula is a description of the set of all runs of the system on which it is satisfied. Thus, it becomes possible to compare the expressive power of different logics depending on which sets of system runs can be specified by the formulas of these logics. More strictly, the comparison relation to estimate the expressive power of logics is defined as follows.

Definition 5. Let there be two logics $\mathcal{L}_1$ and $\mathcal{L}_2$, in which the satisfiability relation is determined in the same class of interpretations. We say that a formula ψ_1 of one of these logics is *equivalent* to a formula ψ_2 of any of these logics if the relation $I \models \psi_1 \Leftrightarrow I \models \psi_2$ holds for every interpretation I. The logic $\mathcal{L}_1$ is said to be *no less expressive* than the logic $\mathcal{L}_2$ (this is denoted by $\mathcal{L}_2 \preccurlyeq \mathcal{L}_1$) if for any formula of the logic

$\mathcal{L}_2$ there is an equivalent formula of the logic $\mathcal{L}_1$. The logics $\mathcal{L}_1$ and $\mathcal{L}_2$ have *equal expressive power* (denoted by $\mathcal{L}_2 \equiv \mathcal{L}_1$) if the relations $\mathcal{L}_2 \preccurlyeq \mathcal{L}_1$ and $\mathcal{L}_1 \preccurlyeq \mathcal{L}_2$ take place. The logic $\mathcal{L}_1$ is *more expressive* than logic $\mathcal{L}_2$ (denoted by $\mathcal{L}_2 \prec \mathcal{L}_1$), if $\mathcal{L}_2 \preccurlyeq \mathcal{L}_1$, but, at the same time, $\mathcal{L}_2 \not\equiv \mathcal{L}_1$.

Labeled transition systems are widely used as models of reactive systems. The observed behavior of such systems is represented by an infinite sequence of observed events. On the set of such sequences, one can define the semantics of three logics: linear order theory $(\mathbb{N}, <)$, linear temporal logic LTL, and monadic second-order logic with a single successor function S1S. In [10], it was found that the logics $(\mathbb{N}, <)$ and LTL have equal expressive power, and in [17, 19], it was shown that the S1S logic is more expressive than LTL.

Transducers are also models of reactive systems, and their behavior can be described by means of a new extension of the linear temporal logic, $\mathcal{LP}$-LTL. The purpose of this paper is to compare the expressive power of the three logics used to describe the behavior of reactive systems: LTL, S1S, and $\mathcal{LP}$-LTL. However, the satisfiability relation for $\mathcal{LP}$-LTL formulas is defined on interpretations that differ from those used in the definitions of the semantics of LTL and S1S logics. To compare $\mathcal{LP}$-LTL with LTL and S1S, we will select two fragments of $\mathcal{LP}$-LTL whose semantics can be equivalently defined on infinite sequences of events, and compare the expressive power of these fragments with the expressive capabilities of LTL and S1S.

Formulas of the first of these fragments, $\mathcal{LP}$-1-LTL, are built over a single-letter alphabet $\mathcal{C} = \{c\}$ of input signals and arbitrary finite alphabet of basic actions $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$. As a family $\mathcal{L}$ of environment behavior patterns in the formulas of this fragment, it is allowed to use any regular languages over the alphabet $\{c\}$, and as a family $\mathcal{P}$ of basic predicates we will use a set of n regular languages of the form $P_i = \mathcal{A}^* a_i$. It will be said that a omega-word $w = a_{i_1} a_{i_2} \ldots a_{i_m} \ldots$ satisfies the formula φ' from the fragment $\mathcal{LP}$-1-LTL if and only if for the trajectory $tr' = (\varepsilon, \alpha'_w)$, where $\alpha'_w = (c, a_{i_1}), (c, a_{i_1} a_{i_2}), \ldots, (c, a_{i_1} a_{i_2} \ldots a_{i_m}), \ldots$, it is true that $tr' \models \varphi'$.

The fragment $\mathcal{LP}$-n-LTL is defined similarly, with the only difference being that the alphabets $\mathcal{C}$ and $\mathcal{A}$ coincide; i.e., $\mathcal{C} = \mathcal{A} = \{a_1, a_2, \ldots, a_n\}$. A omega-word $w = a_{i_1} a_{i_2} \ldots a_{i_m} \ldots$ satisfies the formula φ'' from the fragment $\mathcal{LP}$-n-LTL if and only if for the trajectory $tr'' = (\varepsilon, \alpha''_w)$, where $\alpha''_w = (a_{i_1}, a_{i_1}), (a_{i_2}, a_{i_1} a_{i_2}), \ldots, (a_{i_m}, a_{i_1} a_{i_2} \ldots a_{i_m}), \ldots$, it is true that $tr'' \models \varphi''$.

To compare the expressive power of LTL, S1S and two selected fragments of $\mathcal{LP}$-LTL logic, we give an alternative definition to the semantics of formulas from the considered fragments on omega-words. Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$ be some final alphabet. A omega-*word* is any mapping w: $\mathbb{N} \to \Sigma$, where $\mathbb{N}$ denotes a set of nonnegative integers. The set of all omega-words will be denoted by Σ^ω. The notation $w|^k$ will indicate the suffix of the word w starting with the letter with a number k, i.e., an infinite word v, for which $v(i) = w(i + k)$ at any $i, i \geqslant 0$. The notation $w[0 \ldots k]$ will indicate the prefix of w, i.e., a finite word $w(0)w(1) \ldots w(k)$. In the case when L is a regular language over a single-letter alphabet $\{c\}$, we will write $i \in L$ instead of $c^i \in L$ for brevity.

Let us modify the definition of the semantics of formulas of the fragment $\mathcal{LP}$-1-LTL as a set of formulas built from letters of the alphabet Σ (used instead of basic predicates) by means of Boolean connectives $\neg, \wedge$ and temporal operators $\mathbf{X}$, $\mathbf{F}_L$, $\mathbf{G}_L$, and $\mathbf{U}_L$ that are parameterized by regular languages L over a single-letter alphabet $\mathcal{C} = \{c\}$.

Definition 6. The satisfiability relation $\models$ of the $\mathcal{LP}$-1-LTL formulas on a set of omega-words is defined by the following rules:

1. for atomic formula $a \in \Sigma$: $w \models a \Leftrightarrow w(0) = a$;
2. $w \models \neg\varphi \Leftrightarrow$ it is false that $w \models \varphi$;
3. $w \models \varphi \wedge \psi \Leftrightarrow w \models \varphi$ and $w \models \psi$;
4. $w \models \mathbf{X}\varphi \Leftrightarrow w|^1 \models \varphi$;
5. $w \models \mathbf{F}_L\varphi \Leftrightarrow \exists i \geqslant 0$, such that $i \in L$ and $w|^i \models \varphi$;

6. $w \models \mathbf{G}_L\varphi \Leftrightarrow \forall i \geqslant 0$, if $i \in L$, then $w|^i \models \varphi$;

7. $w \models \varphi\mathbf{U}_L\psi \Leftrightarrow \exists i \geqslant 0$, such that $i \in L$ and $w|^i \models \psi$, and $\forall j, 0 \leqslant j < i$, if $j \in L$, then $w|^j \models \varphi$.

An example of a formula of $\mathcal{LP}$-1-LTL fragment is the expression $\mathbf{G}_{(cc)^*}a$. As can be understood from the definition, a omega-word w satisfies this formula if and only if for any $i, i \geqslant 0$ $w(2i) = a$ is true.

The syntax of $\mathcal{LP}$-n-LTL differs from $\mathcal{LP}$-1-LTL in three aspects:

1. in addition to the operator $\mathbf{X}$ ("NeXttime", at the next time point), its dual operator $\mathbf{Y}$ appears;

2. the operators $\mathbf{X}$ and $\mathbf{Y}$ are parameterized by letters of the alphabet Σ : $\mathbf{X}_a, \mathbf{Y}_b$;

3. the operators $\mathbf{F}_L$, $\mathbf{G}_L$, and $\mathbf{U}_L$ are parameterized by regular languages over the alphabet Σ.

Definition 7. The satisfiability relation $\models$ of the $\mathcal{LP}$-n-LTL formulas on omega-words is defined as follows:

1. satisfiability of atomic formulas, negation $\neg$ and conjunction $\wedge$ are defined exactly the same as in the case of $\mathcal{LP}$-1-LTL;

2. $w \models \mathbf{X}_c\varphi \Leftrightarrow w(0) = c$ and $w|^1 \models \varphi$;

3. $w \models \mathbf{Y}_c\varphi \Leftrightarrow w(0) \neq c$ or $w|^1 \models \varphi$;

4. $w \models \mathbf{F}_L\varphi \Leftrightarrow \exists i \geqslant 0$, such that $w[0 \ldots i] \in L$ and $w|^i \models \varphi$;

5. $w \models \mathbf{G}_L\varphi \Leftrightarrow \forall i \geqslant 0$, if $w[0 \ldots i] \in L$, then $w|^i \models \varphi$;

6. $w \models \varphi\mathbf{U}_L\psi \Leftrightarrow \exists i \geqslant 0$, such that $w[0 \ldots i] \in L$ and $w|^i \models \psi$, and $\forall j, 0 \leqslant j < i$, if $w[0 \ldots j] \in L$, then $w|^j \models \varphi$.

5. $\mathcal{LP}$-1-LTL VS LTL

It can be easily seen that the temporal logic LTL is a subset of $\mathcal{LP}$-1-LTL, in which the regular language $\mathcal{C}^*$ is used as a parameter of all temporal operators. It follows that $LTL \preccurlyeq \mathcal{LP}$-1-$LTL$. Moreover, the following stronger statement holds true.

Theorem 1. $LTL \prec \mathcal{LP}$-1-LTL

The study [19] showed that LTL formulas are unable to describe a set of infinite words that have a given letter in all positions, the number of which is a multiple of a certain number $k, k \geq 2$. Here, we present another proof of this result using the *Ehrenfeucht–Fraïssé game* (see [3]).

Everywhere further in LTL formulas instead of the operator $\mathbf{G}$ we will use the equivalent combination $\neg\mathbf{F}\neg$. The technique of *Ehrenfeucht–Fraïssé games* for temporal logics was introduced by the authors of [3] to build a hierarchy of LTL formulas. To describe this technique, we need the concept of the *nesting depth of the operators* of the formula.

Definition 8. Let us define the *nesting depth of operators depth*(φ) of a formula $\varphi \in LTL$ by induction on its structure:

1. for an atomic formula $a \in \Sigma$, it is assumed that $depth(a) = 0$;

2. $depth(\neg\varphi) = depth(\varphi)$;

3. $depth(\varphi \wedge \psi) = \max(depth(\varphi), depth(\psi))$;

4. $depth(\mathbf{X}\varphi) = depth(\varphi) + 1$;

5. $depth(\mathbf{F}\varphi) = depth(\varphi) + 1$;

6. $depth(\varphi\mathbf{U}\psi) = \max(depth(\varphi), depth(\psi)) + 1$.

The Ehrenfeucht–Fraïssé game has two players: Player 1 and Player 2; one of them ultimately wins, and the other one loses. The game is played on a pair of omega-words (w_0, w_1), which is called the *configuration* of the game. Player 1 seeks to show that the omega-words w_0 and w_1 are *distinguishable* by a certain LTL formula, while Player 2 attempts to prove the opposite.

Definition 9. The Ehrenfeucht–Fraïssé game consisting of k *rounds* (k-game) with an initial configuration (w_0, w_1) is defined by induction on k:

• Player 1 wins a game of 0 rounds if $w_0(0) \neq w_1(0)$. Otherwise, Player 2 wins.

• Player 2 wins a game of $(k+1)$ rounds if $w_0(0) \neq w_1(0)$. Otherwise, a *round* is played, which leads either to the victory of Player 1 or to a new configuration (w_0', w_1') on which the game of rounds is played.

In each *round*, a change of configurations takes place. Player 1 chooses one of three possible *moves*:

• **X-move.** Sets the pair $\left(w_0|^1, w_1|^1\right)$ as the new configuration;

• **F-move.**

1. Player 1 chooses a word w_s, $s \in \{0,1\}$ and position $i_s \geqslant 0$;

2. Player 2 responds by choosing a position $i_{1-s} \geqslant 0$ in a word w_{1-s};

3. The pair $\left(w_s|^{i_s}, w_{1-s}|^{i_{1-s}}\right)$ is set as the new configuration.

• **U-move.**

1. Player 1 chooses the word w_s, $s \in \{0,1\}$ and position $i \geqslant 0$;

2. Player 2 responds, choosing such position $i_{1-s} \geqslant 0$ in the word w_{1-s} that if $i_s = 0$, then $i_{1-s} = 0$;

3. Player 1 chooses one of two options:

(a) Set the pair $(w_s|^{i_s}, w_{1-s}|^{i_{1-s}})$ as the new configuration;

(b) Player 1 chooses the position i'_{1-s} in the word w_{1-s}, where $0 \leqslant i'_{1-s} < i_{1-s}$, and Player 2 responds, choosing the position $i'_s \in w_s$, $0 \leqslant i'_s < i_s$. The pair $\left(w_s|^{i'_s}, w_{1-s}|^{i'_{1-s}}\right)$ is set as the new configuration.

Player 2 is said to have a *winning strategy* in a game consisting of k rounds with initial configuration (w_0, w_1) if he can win this game regardless of Player 1's moves. This fact is indicated by the notation $w_0 \sim_k w_1$. Otherwise, it is said that Player 1 has a winning strategy.

In [3], an important property of the Ehrenfeucht–Fraïssé game was proved:

Proposition 1. For any integer nonnegative k and any two omega-words w_0 and w_1, the relation $w_0 \sim_k w_1$ is true if and only if for any formula $\varphi \in LTL$ with $depth(\varphi) \leqslant k$ it is true that $w_0 \models \varphi \Leftrightarrow w_1 \models \varphi$.

Now we can proceed to the proof of Theorem 1. Let us show that the property $W_a^{2n} = \left\{w \in \Sigma^\omega \middle| w(2i) = a,\ i \in \mathbb{N}_0\right\}$ of the omega-words in the alphabet $\Sigma = \{a, b\}$, which is described by the $\mathcal{LP}$-1-LTL formula $\mathbf{G}_{(cc)^*}a$, cannot be expressed by any formula of the LTL.

Proof. Let us consider a pair of words $u_n = a^{2n+1}ba^\omega \in W_a^{2n}$ and $v_n = a^{2n}ba^\omega \notin W_a^{2n}$. We will show, using the technique of the Ehrenfeucht–Fraïssé games, that at $2n > k$, LTL formulas do not distinguish u_n and v_n. For convenience, the prefixes of the words u_n and v_n up to the letter b will be called heads, and after the letter b, *tails*. The proposed rigorous proof is based on the idea that Player 1 fails to distinguish blocks a^{2n+1} and a^{2n} in k rounds and, therefore, will not be able to find the difference between u_n and v_n.

For a rigorous justification, let us describe Player 2's winning strategy:

1. If Player 1 makes an **X**-move, Player 2 must play in accordance with the rules (the configuration is "shifted" by one letter, while the number of rounds is reduced by one);

2. If Player 1 makes an **F**-move

(a) to the *tail*, Player 2 must also make a move to the *tail* to the same position (counting from the first letter of the *tail*);

(b) to the head, Player 2 must make a move to the head, so that the distance from the new position to the last letter of the head is the same as in the word of Player 1.

(c) to the *position of the letter b*, Player 2 must also move to the *position of the letter b*;

3. If Player 1 makes a **U**-move, Player 2 should play similar to the case of the **F**-move, except for the situation when Player 1 makes a move to the first letter of the head (i.e., remains in the previous position); in this case, Player 2 must play according to the rules and also make a move to the first letter of the head.

Similarly, if Player 1 then makes a "move back" (applying the third item of the description of a **U**-move), Player 2 should respond to his action as described above, except when Player 1 makes a move to the first letter of the head (then Player 2 should also make a move to the first letter of the head).

It can be easily seen that the following is true.

Proposition 2. When Player 2 uses the strategy described above, the configuration (u, v) satisfies the following properties throughout the game:

1. $u[0] = v[0]$;

2. if $u \neq v$, then the lengths of the shortest letter blocks a in the words u and v are no smaller than the remaining number of rounds;

3. between the second and third parts of the **U**-move for each position j in Player 2's word w_2 that is smaller than the current position, there is a corresponding position i in Player 1's word w_1, such that the configuration $\left(w_1|^i, w_2|^j\right)$ satisfies properties 1 and 2.

Thus, Theorem 1 is proved.

6. $\mathcal{LP}$-n-LTL VS S1S

Let us now proceed to the study of the expressive power of $\mathcal{LP}$-n-LTL fragment. Obviously, $\mathcal{LP}$-n-LTL fragment is no less expressive than $\mathcal{LP}$-1-LTL fragment. Further we will show that it is as expressive as the monadic second-order logic S1S. To prove this fact, we will not consider the syntax and semantics of S1S logic, but use instead the interconnection of this logic with finite automata designed to recognize omega-words. It is well known (see, e.g., [15]) that the following three statements regarding omega-languages are equivalent:

- the set L consists of all omega-words on which a certain formula Φ of the S1S logic is satisfied, i.e., $L = \{w \mid w \models \Phi\}$,
- the set of omega-words L is an ω-*regular language*
- the set of omega-words LL is recognized by some *Büchi (Rabin, Muller, Street) automaton*.

Therefore, we will compare the expressive power of the $\mathcal{LP}$-n-LTL fragment with computational capabilities of finite ω-automata.

Definition 10. *A finite automaton over superwords* (ω-*automaton*) is a quintuplet $A = (\Sigma, Q, Q_0, \Delta, \mathcal{F})$, where

- Σ is the final alphabet,
- Q is a finite set of *states*,
- $\Delta \subseteq Q \times \Sigma \times Q$ is the *transition relation*,
- $Q_0 \in Q$ is the set of *initial states*,
- $\mathcal{F}$ is the *accepting rule*.

An ω-automaton is called *deterministic*, if $Q_0 = \{q_0\}$ and the transition relation is a function; i.e., for any state q and any letter a there is only one state $q' \in Q$, for which $(q, a, q') \in \Delta$. A *run* of an ω-automaton A on a omega-word w is any mapping $\rho: \mathbb{N}_0 \to Q$, for which $\rho(0) \in Q_0$ and for any i the relation $(\rho(i), w(i), \rho(i+1)) \in \Delta$ is satisfied.

The notation $\inf(\rho)$ indicates the set of states that occur in the run ρ infinitely often. Various types of ω-automata differ from each other by accepting rules. Here, we consider the Büchi automata and Muller automata. The accepting rule of a generalized Büchi automaton B is a family $\mathcal{F} = \{F_1, F_2, \ldots, F_m\}$ of subsets F_i, $F_i \subseteq Q$, $1 \leqslant i \leqslant m$, of the sets of states of the automaton B.

Definition 11. A Büchi automaton B *accepts* a omega-word w if and only if there is a run ρ of this automaton on the omega-word w such that $\inf(\rho) \cap F_i \neq \varnothing$ for each i, $1 \leqslant i \leqslant m$.

The accepting rule of the Muller automaton M is also specified with the family $\mathcal{F} = \{E_1, E_2, \ldots, E_m\}$ of subsets E_i, $E_i \subseteq Q$, $1 \leqslant i \leqslant m$, of the set of states of the ω-automaton M.

Definition 12. A Muller automaton M *accepts* a omega-word w if and only if there is a run ρ of this automaton on the omega-word w such that $\inf(\rho) \in \mathcal{F}$.

A set of omega-words accepted by automaton A will be denoted by $L(A)$. It is said that the automaton A *recognizes* the language $L(A)$.

We show that the class of languages recognized by ω-automata coincides with the class of languages that can be described by $\mathcal{LP}$-n-LTL formulas. To do this, firstly, let us set each formula $\psi \in \mathcal{LP}$-n-LTL into correspondence with a nondeterministic Büchi automaton B_ψ such that $L(B_\psi) = \{w | w \models \psi\}$. Secondly, for each deterministic Muller automaton M we will construct such a formula $\psi_M \in \mathcal{LP}$-n-LTL that $L(M) = \{w | w \models \psi_M\}$.

In the first case, we will need to slightly modify the well-known scheme of constructing a Büchi automaton according to an LTL formula (see, for example, [1]). Let $\varphi \in \mathcal{LP}$-n-LTL. We rewrite the formula φ using the identities $\mathbf{G}_L\psi = \neg\mathbf{F}_L\neg\psi$ and $\mathbf{F}_L\psi = \mathbf{true}\mathbf{U}_L\psi$ so as to exclude the operators $\mathbf{F}$ and $\mathbf{G}$ leaving only $\mathbf{U}$. Further, without loss of generality, we assume that φ is constructed with the help of $\neg, \wedge, \mathbf{X}_c, \mathbf{Y}_c, \mathbf{U}_L$.

Definition 13. A *closure* $cl(\varphi)$ of the formula φ is the set of all subformulas of this formula, i.e., the smallest set such that

- $\varphi \in cl(\varphi)$;
- if $\neg\psi \in cl(\varphi)$, then $\psi \in cl(\varphi)$;
- if $\psi \wedge \chi \in cl(\varphi)$, then $\psi, \chi \in cl(\varphi)$;
- if $\mathbf{X}\psi \in cl(\varphi)$, then $\psi \in cl(\varphi)$;
- if $\psi\mathbf{U}_L\chi \in cl(\varphi)$, then $\psi, \chi \in cl(\varphi)$.

We set a omega-word $w \in \Sigma^\omega$ into correspondence to such a omega-word ρ_w in the alphabet $2^{cl(\varphi)}$ that for each i the inclusion $\psi \in \rho_w(i)$ takes place if and only if $w^i \models \psi$. Next, we will construct a Büchi automaton $\mathcal{B}_\varphi$ whose set of states is the family of all subsets of the closure $cl(\varphi)$ of the formula φ, and the sequence ρ_w is a run of B on the word w. We will describe the semantics of the formula φ using states, transitions, and an accepting rule; the semantics of the parameters of operators will be described using finite automata.

Definition 14. An (uninitialized) *finite automaton* is a triple $A = (\Sigma, Q, \delta)$, where Σ is the alphabet, Q is the set of states, and δ: $Q \times \Sigma \to Q$ is the transition function.

Let us extend the transition function in a standard way on the set Σ^* of all finite words in the alphabet Σ: $\delta(q, \varepsilon) = q$ for an empty word ε, and $\delta(q, \alpha a) = \delta(\delta(q, \alpha), a)$ for any $a \in \Sigma$ and $\alpha \in \Sigma^*$. An automaton A is called *initialized* if an *initial state* q_0 and a set of accepting states F are specified. An automaton initialized in such a way is denoted by $A(q_0, F)$; it *accepts* the finite word α if and only if $\delta(q_0, \alpha) \in F$. The notation $L(A(q_0, F))$ indicates the set of all words accepted by the automaton $A(q_0, F)$ i.e. the *language* of this automaton. Further, mentioning a regular language L we assume that it is recognized by the automaton $A^L(q_0^L, F^L)$, where $A^L = (\Sigma, Q^L, \delta^L)$.

Proposition 3. For each formula φ of the $\mathcal{LP}$-n-LTL logic, there is a generalized Büchi automaton $\mathcal{B}_\varphi$ with the following property: for any omega-word w the relation $w \models \varphi$ is satisfied if and only if w is accepted by the Büchi automaton $\mathcal{B}_\varphi$.

Proof. Let φ be an arbitrary formula of $\mathcal{LP}$-n-LTL. We describe the structure of the corresponding generalized Büchi automaton $\mathcal{B}_\varphi$. To do this, we construct a set $\widehat{Q} = 2^{cl(\varphi)} \times Q^{L_1} \times Q^{L_2} \times \ldots \times Q^{L_m}$, where $L_1, L_2, \ldots, L_m$ are the parameters of all temporal operators used in φ. It is important to note that if some operators are given the same parameter L, this parameter is still included in the list $L_1, L_2, \ldots, L_m$ for each operator separately.

We will call the element $q = (S, q^{L_1}, \ldots, q^{L_m}) \in \widehat{Q}$ *normal* if

- the set of formulas S is consistent with respect to classical propositional logic, i.e.

—if $\psi \wedge \chi \in S$, then $\psi \in S$ and $\chi \in S$;

—if $\psi \in S$, then $\neg\psi \notin S$;

—if $\mathbf{true} \in cl(\varphi)$, then $\mathbf{true} \in S$;

—if $a \in S \cap \Sigma$ and $b \in S \cap \Sigma$, then $a = b$;

• q is locally consistent with the operator $\mathbf{U}$, i.e. for any formula of the form $\psi \mathbf{U}_{\mathbf{L}} \chi \in cl(\varphi)$

—if $\chi \in S$ and $q^L \in F^L$, then $\psi \mathbf{U}_{\mathbf{L}} \chi \in S$;

—if $\psi \mathbf{U}_{\mathbf{L}} \chi \in S$, and $\chi \notin S$, and $q^L \in F^L$, then $\psi \in S$;

• the set of formulas S is maximum, i.e.,

—$S \cap \Sigma \neq \varnothing$;

—for any formula $\psi \in cl(\varphi)$ if $\psi \notin S$, then $\neg\psi \in S$.

The notation $Norm(\varphi)$ indicates the set of all normal elements $\hat{Q}$.

We now proceed directly to the construction of the Büchi automaton $\mathcal{B}_\varphi$, which recognizes a set of omega-words on which a given formula φ of $\mathcal{LP}$-n-LTL is satisfied. In contrast to the similar method described in the monograph [1] for constructing a Büchi automaton corresponding to an LTL formula, we have to take into account the finite automata that are used to parameterize temporal operators.

Let us consider a Büchi automaton $\mathcal{B}_\varphi = (\Sigma, Q, Q_0, \Delta, \mathcal{F})$, where

• $Q = Norm(\varphi)$;

• $Q_0 = \left\{ q \in Q \middle| q = (S, q_0^{L_1}, \ldots, q_0^{L_m}), \varphi \in S \right\}$;

• $\mathcal{F} = \left\{ F_{\psi \mathbf{U}_L \chi} \middle| \psi \mathbf{U}_L \chi \in cl(\varphi) \right\}$, where

$F_{\psi \mathbf{U}_L \chi} = \left\{ q \in Q \middle| \psi \mathbf{U}_L \chi \notin S \text{ or simultaneously } \chi \in S \text{ and } q^L \in F^L \right\}$

• The transition relation Δ: $Q \times \Sigma \to 2^Q$ for the state $q = (S, q^{L_1}, \ldots, q^{L_m}) \in Q$ and letter $a \in \Sigma$ is defined as follows:

—if $a \notin S \cap \Sigma$ or there is such a subformula $\mathbf{X}_c \psi \in S$ that $a \neq c$, then $\Delta(q, a) = \varnothing$;

—if $S \cap \Sigma = \{a\}$ and for each subformula $\mathbf{X}_c \psi \in S$ it is true that $a = c$, then $\Delta(q, a)$ is equal to the set of all states of such $\hat{q} = (\hat{S}, \hat{q}^{L_1}, \ldots, \hat{q}^{L_m}) \in Q$ for which the following conditions are met:

1. for each subformula $\mathbf{X}_c \psi \in cl(\varphi)$: $\mathbf{X}_c \psi \in S \Leftrightarrow \psi \in \hat{S}$;

2. for each subformula $\mathbf{Y}_c \psi \in cl(\varphi)$: if $a = c$, then $\mathbf{Y}_c \psi \in S \Leftrightarrow \psi \in \hat{S}$;

3. for each subformula $\psi \mathbf{U}_L \chi \in cl(\varphi)$: $\psi \mathbf{U}_L \chi \in S \Leftrightarrow (q^L \in F^L$ and $\chi \in S)$ or simultaneously $\psi \mathbf{U}_L \chi \in \hat{S}$ and, if $q^L \in F^L$, then $\psi \in S)$.

4. for each parameter L_i, if the subformula $\psi \mathbf{U}_{L_i} \chi$ marked by it belongs to S, then $\hat{q}^{L_i} = \delta^{L_i}(q^{L_i}, a)$, else $\hat{q}^{L_i} = q^{L_i}$.

In essence, the restrictions imposed on the transition relation Δ fully describe the semantics of the temporal operators $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{U}$. The last rule in the description of the transition relation Δ, in particular, guarantees that each automaton responsible for the parameter of the temporal operator starts its work as soon as the processing of the subformula entitled with this operator begins.

In order for the computations of the Büchi automaton $\mathcal{B}_\varphi$ to match correctly the semantics of the operators Until $\mathbf{U}_L$ occurring in the formula φ, for each subformula $\zeta = \psi \mathbf{U}_L \chi$ of the formula φ there is a set of states F_ζ in the accepting rule for this ω-automaton. The presence of such a set in the accepting rule ensures that in each run ρ_w for which $\zeta \in \rho_w(0)$, the inclusion $\chi \in S_j$ will be true for some $j \geqslant 0$, $\rho_w(j) = (S_j, \ldots, q_j^L, \ldots)$, $q_j^L \in F^L$ (i.e., $w[0 \ldots j] \in L$); and $\psi \in S_i$ for all $i < j$ such that $q_i^L \in F^L$ (i.e., $w[0 \ldots i] \in L$). The construction of F_ζ also takes into account the influence of the parameter of the operator *Until* on the satisfiability of the latter: the omega-word w satisfies the formula $\psi \mathbf{U}_L \chi$ only when χ ultimately becomes true in the position that is "consistent" with the parameter L.

Further, it can be easily shown, using the same line of reasoning that is given in the monograph [1] to substantiate a similar discussion, that the Büchi automaton $\mathcal{B}_\varphi$ constructed in such a way has the required

property: for any omega-word w the relation $w \models \varphi$ is satisfied if and only if w is accepted by the Büchi automaton $\mathcal{B}_\varphi$.

If we take into account the note made earlier regarding the relationship between Büchi automata and S1S logic, then, based on Statement 3, it can be concluded that $\mathcal{LP}\text{-}n\text{-}LTL \preccurlyeq$ S1S.

We now demonstrate how, using the Muller automaton, to construct the $\mathcal{LP}\text{-}n\text{-}LTL$ formula equivalent to it.

Proposition 4. For each deterministic Muller automaton there is a formula φ of the $\mathcal{LP}\text{-}n\text{-}LTL$ that satisfies the following requirement: an automaton M accepts a omega-w if and only if $w \models \varphi_M$.

Proof. Let there be an arbitrary deterministic Muller automaton $M = (\Sigma, Q, q_0, \Delta, \mathcal{F})$ with a single initial state q_0 and accepting rule $\mathcal{F} = \{E_1, E_2, \ldots, E_m\}$. Recall that the *accepting run* of an automaton M is its run ρ such that $\inf(\rho) = E_i$ for some $E_i \in \mathcal{F}$. It should be noted that the only run ρ_w of a deterministic Muller automaton on the word w is uniquely defined by this word, and the state in which the automaton is at some step of the computation depends only on the prefix of the word w read by this step. We will describe this dependence by $\mathcal{LP}\text{-}n\text{-}LTL$ formulas.

Let us consider an uninitialized finite automaton $A = (\Sigma, Q, \Delta)$, as the basis of a Muller automaton M, and its initialization $M(q, Q') = A(q, Q')$ for all possible conditions $q \in Q$ and subsets $Q' \subseteq Q$. For simplicity, we will use the notation $M(q, Q')$ instead of $L(M(q, Q'))$.

For each set of states $E_i \in \mathcal{F}$ we construct the following three formulas:

1. $\psi_1^i = \bigwedge_{q_e \in E_i} \mathbf{F}_{M(q_0, q_e)}\mathbf{true}$. Formula ψ_1^i is satisfied on a omega-word w if and only if the run of the Muller automaton M on w goes via all states of the set E_i.

2. $\psi_2^i = \bigwedge_{q \in Q} \mathbf{G}_{M(q_0, q)} \left(\bigwedge_{q_e \in E_i} \mathbf{F}_{M(q, q_e) \setminus \{\varepsilon\}}\mathbf{true} \right)$. This formula is satisfied on a omega-word $a \in \Sigma$ if and only if the run of the Muller automaton M on w goes via each state from the set E_i infinitely frequently.

3. $\psi_3^i = \bigwedge_{q \in Q} \mathbf{F}_{M(q_0, q)} \left(\mathbf{G}_{M(q, Q \setminus E_i)}\mathbf{false} \right)$. This formula is satisfied on a omega-word w if and only if the run of the Muller automaton M on w goes infinitely frequently only via states from the set E_i.

From the above explanation of the conceptual meaning of the formulas ψ_1^i, ψ_2^i, and ψ_3^i, it follows that $\varphi_M = \bigvee_{i=1}^m (\psi_1^i \wedge \psi_2^i \wedge \psi_3^i)$ is the desired formula.

Thus, we arrive at

Theorem 2. $\mathcal{LP}\text{-}n\text{-}LTL \equiv$ S1S.

7. CONCLUSIONS

After the publication of the paper [5], in which a detailed study of the LTL was carried out as a specification means for describing the behavior of computing systems, several attempts have been made to improve the expressive power of this logic. In [14], the LTL syntax was supplied with quantification for basic predicates. It turned out that the expressive power of the quantified extension $QLTL$ significantly exceeds the descriptive capabilities of the LTL. For example, with the use of quantifiers, the property $W_1^{2n} = \left\{ w \in \Sigma^\omega \middle| w(2i) = 1,\ i \in \mathbb{N}_0 \right\}$ of omega-words in the alphabet $\Sigma = \{0, 1\}$, which cannot be expressed in the LTL, is described by the following formula of its extention $QLTL$:

$$\exists q(q \wedge \mathbf{G}(q \to \mathbf{X}\neg q) \wedge \mathbf{G}(\neg q \to q) \wedge \mathbf{G}(q \to p)).$$

The author of [19] proposed a method for introducing new temporal operators using right-linear grammars. The words generated by these grammars define the patterns on which the satisfiability of the formulas in the scope of a temporal operator is checked. For example, the temporal operator $\mathbf{E}p$ that checks the abovementioned property W_1^{2n} is specified by the grammar of two rules, $V_0 \to pV_1$ and $V_1 \to \mathbf{true}\ V_0$. Similarly, the patterns for describing the semantics of new temporal operators can be described using finite automata, as shown in [12, 16]. The fixed-point operators have also received attention: the linear temporal logic $\mu\text{-}LTL$ enriched with these operators was introduced and studied in [18].

For all listed LTL extensions (except for $\mu\text{-}LTL$), it was shown that they have exactly the same expressive power as the S1S logic, but at the same time the satisfiability problem for all these logics remains

PSPACE-complete, same as for LTL. Introducing $\mathcal{LP}$-LTL, we did not seek only to improve the expressive power of LTL as such. Our goal was to offer an adequate language for the specification of the behavior of reactive systems modeled by transducers. Nevertheless, Theorem 2 shows that the expressive power of the logical specification language that we proposed has made the same sudden jump as other LTL extensions studied in [12, 14, 16, 18, 19].

The idea of parameterization of temporal operators is not new: almost the same parameterization of temporal operators as in this paper was introduced to dynamically extend LTL in [9]. In fact, the fragment $\mathcal{LP}$-n-LTL is very similar to $DLTL$ logic introduced in that paper. However, our LTL extension differs mainly in the fact that the basic predicates in it are also parameterized. In the future, we intend to perform a comparison of the expressive power of $\mathcal{LP}$-n-LTL and dynamic logics, including $DLTL$ [9] and PDL [4].

Theorem 1 that we have proven opens up yet another line of research. Each of PDL extensions proposed in [9, 14, 16, 18, 19] turns out to be as expressive as one of the two logics LTL and S1S. However, as it is shown in Theorem 1, the fragment $\mathcal{LP}$-1-LTL considered in our paper turned out to be more expressive than LTL, and, as shown in Theorem 2, another fragment $\mathcal{LP}$-n-LTL turned out to be as expressive as S1S logic. We make an assumption that the two fragments of $\mathcal{LP}$-LTL logic considered in our paper have different expressive power. If this assumption proves to be true, the fragment $\mathcal{LP}$-1-LTL will be a natural new example of temporal logic, which by expressive power occupies an intermediate position between the well-known logics $(\mathbb{N}, <)$ and S1S. We believe that this hypothesis can be proved with the aid of the Ehrenfeucht–Fraïssé games that were used in this paper.

ACKNOWLEDGMENTS

This work was supported by the Russian Foundation for Basic Research, grant no. 18-01-00854.

REFERENCES

1. Baier, C. and Katoen, J., *Principles of Model Checking,* MIT Press, 2008.
2. Clarke, E.M., Gramberg, O., and Peled, D.A., *Model Checking,* MIT Press, 1999.
3. Etessami, K. and Wilke, T., An until hierarchy and other applications of an Ehrenfeucht-Fraisse game for temporal logic, *Inf. Comput.,* 2000, vol. 160, no. 1, pp. 88–108.
4. Fisher, J.M. and Ladner, R.E., Propositional dynamic logic of regular programs, *J. Comput. Syst. Sci.,* 1979, vol. 18, pp. 194–211.
5. Gabbay, D., Pnueli, A., Shelach, S., and Stavi, J., The temporal analysis of fairness, *Proceedings of 7-th ACM Symposium on Principles of Programming Languages,* 1980, pp. 163–173.
6. Gnatenko, A.R. and Zakharov, V.A., On the complexity of verification of finite automata-converters over commutative semigroups, *Trudy 18-1 Mezhdunarodnoi konferenii Problemy teoreticheskoi kibernetiki* (Proceedings of the 18th International Conference Problems of Theoretical Cybernetics), 2017, pp. 68–70.
7. Gnatenko, A.R. and Zakharov, V.A., On the model checking of finite state transducers over semigroups, *Tr. Inst. Sist. Program. Ross. Akad. Nauk,* 2018, vol. 30, no. 3, pp. 303–324.
8. Harel, D., Kozen, D., and Parikh, P., Process logic: Expressiveness, decidability, completeness, *J. Comput. Syst. Sci.,* 1982, vol. 25, no. 2, pp. 144–170.
9. Henriksen, J.J. and Thiagarajan, P.S., Dynamic linear time temporal logic, *Ann. Pure Appl. Logic,* 1999, vol. 96, nos. 1–3, pp. 187–207.
10. Kamp, J.A.W., Tense logic and the theory of linear order, *PhD Thesis,* Los Angeles: University of California, 1968.
11. Kozlova, D.G. and Zakharov, V.A., On the model checking of sequential reactive systems, *Proceedings of the 25th International Workshop on Concurrency, Specification and Programming (CS&P 2016), CEUR Workshop Proceedings,* 2016, vol. 1698, pp. 233–244.
12. Kupferman, O., Piterman, N., and Vardi, M.Y., Extended temporal logic revisited, *Proceedings of 12-th International Conference on Concurrency Theory,* 2001, pp. 519–535.
13. Leucker, M. and Sanchez, C., Regular linear temporal logic, *Proceedings of the 4-th International Colloquium on Theoretical Aspects of Computing,* 2007, pp. 291–305.
14. Manna, Z. and Wolper, P., Synthesis of communicating processes from temporal logic specifications, *Lect. Notes Comput. Sci.,* 1981, vol. 131, pp. 253–281.
15. Thomas, W., Automata on infinite objects, in *Handbook of Theoretical Computer Science,* vol. B: *Formal Models and Semantics,* 1990, pp. 133–192.
16. Vardi, M.Y. and Wolper, P., Yet another process logic, *Lect. Notes Comput. Sci.,* 1983, vol. 14, pp. 501–512.

17. Vardi, M.Y. and Wolper, P., An automata-theoretic approach to automatic program verification, *Proceedings of the First Symposium on Logic in Computer Science,* 1986, pp. 322–331.
18. Vardi, M.Y., A temporal fixpoint calculus, *Proceedings of the 15-th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages,* 1989, pp. 250–259.
19. Wolper, P., Temporal logic can be more expressive, *Inf. Control,* 1983, vol. 56, nos. 1–2, pp. 72–99.
20. Wolper, P. and Boigelot, B., Verifying systems with infinite but regular state spaces, *Lect. Notes Comput. Sci.,* 1998, vol. 1427, pp. 88–97.
21. Zakharov, V.A., Equivalence checking problem for finite state transducers over semigroups, *Lect. Notes Comput. Sci.,* 2015, vol. 9270, pp. 208–221.
22. Zakharov, V.A. and Temerbekova, G.G., On the minimization of finite state transducers over semigroups, *Autom. Control Comput. Sci.,* 2017, vol. 51, no. 7, pp. 523–530.
23. Zakharov, V.A. and Jaylauova S.R., On the minimization problem for sequential programs, *Autom. Control Comput. Sci.,* 2017, vol. 51, no. 7, pp. 689–700.

Translated by M. Chubarova

SPELL: 1. OK