УДК 519.713.8

Using an extension of CTL^* for specification and verification of sequential reactive systems

Gnatenko A.R., Zakharov V.A.

(National Research University Higher School of Economics)

Sequential reactive systems such as controllers, device drivers, computer interpreters operate with two data streams and transform input streams of data (control signals, instructions) into output streams of control signals (instructions, data). Finite state transducers are widely used as an adequate formal model for information processing systems of this kind. Since runs of transducers develop over time, temporal logics, obviously, could be used as both simple and expressive formalism for specifying the behavior of sequential reactive systems. However, the conventional applied temporal logics (HML, LTL, CTL, μ-calculus) do not suit this purpose well, since their formulae are interpreted over ω-languages, whereas the behavior of transducers are represented by binary relations on infinite sequences, i.e. by ω-transductions. To provide temporal logics with the ability to specify the property of transductions that characterize the behavior of reactive systems, we introduced new extensions of these logics. Two principal features distinguish these extension: 1) temporal operators are parameterized by sets of streams (languages) admissible for input, and 2) sets (languages) of expected output streams are used as basic predicates. In our previous papers [7, 8, 13] we studied the expressive power and the model checking problem for *Reg*-LTL and *Reg*-CTL which are the extensions of LTL and CTL where the languages mentioned above are regular ones. We discovered that parametrization of this kind increases expressive power of temporal logics though retains the decidability of the model checking problem. Our next step in the systematic exploration of new extensions of temporal logics intended for specification and verification of sequential reactive systems is the study of the model checking problem for finite state transducers against *Reg*-CTL^* formulae. In this paper we develop a model checking algorithm for *Reg*-CTL^* and show that this problem is in ExpSpace.

Keywords: *reactive system, model checking, finite state transducer, temporal logic, regular language, specification, verification*

1. Introduction

Finite state machines are widely used in computer science as models of sequential computing systems. In particular, finite state transducers serve as a suitable formal model for various software and hardware systems such as controllers, device drivers, network switches, computer interpreters, etc. which operate with two data streams. These devices and programs receive streams of data (control signals, instructions) at their inputs and transform them into output streams of control signals (instructions, data). Hardware devices of this type include adapters, network switches, controllers. In software engineering transducers are used as formal models of various programs and protocols that manipulate with strings of symbols, flows commands, data streams, etc. (see [2, 11, 19]). Such programs, systems and devices can be grouped under the general name *sequential reactive systems*. A sequential reactive system operates in discrete time. At each step of computation it receives a control signal (or a piece of input data) from the environment and generates (performs, outputs) a sequence of actions (or a piece of output data) in response. Those output actions and the order of their performance depend on the received input signal, but also on all the previous control signals.

We focus on so called *online systems* which are supplied with only finite memory. In [7] we presented and discussed a number of examples to show that finite state transducers is a simple albeit rather adequate formalism for modeling the behavior of reactive sequential systems in many applications. The behavior of such systems is characterized not by a set of sequences of events, but by a relationship between two sequences of events. A typical property of such a behavior that needs verification is that for for each input word of a given pattern the transducer always outputs a word of another given pattern. The requirements of this kind can be formally specified by means of temporal logics adapted for reasoning about pairs of sequences of events.

Such temporal logics were introduced and studied in [6, 7, 13]. In [13] a new extension of Linear Temporal Logics (LTL) was introduced as a formal language for specification of the behavior of sequential reactive systems. In this logic $\mathcal{LP}$-LTL the temporal operators are parameterized by sets of words (languages) that represent distinguished flows of control signals that impact on a reactive system. Basic predicates in $\mathcal{LP}$-LTL are also languages in the alphabet of basic actions of a transducer; they represent the expected response of a transducer to the specified environmental influences. In [13] the authors studied the model checking problem for regular fragment *Reg*-LTL of this logic when only regular languages are used as basic predicates and parameters of temporal operators. It was shown that the model checking problem for finite state transducers against the formulae of *Reg*-LTL is decidable in double exponential time. In [7] we estimate the expressive power $\mathcal{LP}$-LTL by comparing it with some well known logics widely used in computer science for specification of reactive systems behavior. In [6] a model checking algorithm was proposed for *Reg-CTL* which is a regular extension of Computational Tree Logic *CTL*.

In this paper we consider a more general specification language *Reg-CTL** which is a regular extension of Generalized Computation Tree Logic *CTL**, develop a model checking algorithm for this logic and estimate its complexity. This is the main contribution of the paper. The results obtained are based on the methods developed in [6, 8, 13]; they are to continue the line of research initiated in these papers. In Section 2 we introduce the formal definition of a finite state transducers as well as the syntax and the semantics *Reg-CTL**. In Section 3 we give a short survey of some previously known extensions of conventional temporal logics and compare them with our family of logics. In Section 4 we propose a model checking procedure for *Reg-CTL** and estimate its complexity. Section 5 is left for the comparative analysis of the results obtained and similar decisions of model checking problem for other extensions of temporal logics, and also for a discussion on the further lines of research.

This research is supported by RFBR project No. 18-01-00854.

2. Reactive systems models and their specifications

Sequential reactive systems such as adapters, controllers, device drivers, computer interpreters operate with two data streams and transform input streams of data (control signals, instructions, etc.) into output streams of data (control signals, instructions, etc.). Such mappings are called *transduction relations*, and finite state transducers are widely used as an adequate formal model for information processing systems of this kind.

Let $\mathcal{C}$ and $\mathcal{A}$ be finite alphabets. The elements of $\mathcal{C}$ are called *input signals* and the elements of $\mathcal{A}$ are *basic actions*. Denote by $\mathcal{A}^*$ the set of all finite words over $\mathcal{A}$, which are called *compound actions*. Given two compound actions u and v, we write uv for their concatenation, and ε for the empty word.

A *finite state transducer* over the $\mathcal{C}$ and $\mathcal{A}$ is a quintuple $\pi = (Q, \mathcal{C}, \mathcal{A}, q_{init}, T)$, where Q is a finite set of *control states*, $q_{init} \in Q$ is the *initial state*, and $T \subseteq Q \times \mathcal{C} \times Q \times \mathcal{A}^*$ is a total *transition relation*. The *size* $|\pi|$ of a transducer π is the size of a table description of its transition relation T. A *transition* $\tau = (q', c, q'', h) \in T$ means that a transducer is capable to output a compound action h and pass control to a state q'' at receiving an input signal c in a state q'. A *trajectory* of π from a state q_0 is any infinite sequence of transitions $tr = \{\tau_i\}_{i \geqslant 1}$ such that $\tau_i = (q_{i-1}, c_i, q_i, h_i) \in T$ for all i, $1 \leqslant i \leqslant n$. A sequence of pairs $(c_1, h_1), (c_2, h_2), \ldots$

of a trajectory tr displays a behavior of a transducer as seen by an outside observer. We denote by $Tr_\pi(q)$ the set of all trajectories of π from a state q, and write simply Tr_π in the case when $q = q_{init}$. We denote by $\mathrm{Fin}(Tr_\pi(q))$ the set of all finite prefixes of trajectories in $Tr_\pi(q)$; such prefixes will be called *finite trajectories* from a state q.

To be able to analyze the behavior of a transducer π, it is advisable to use a structure that represents all runs of π; it is obtained as an unfolding of the transition relation of π. A *computation graph* of a finite state transducer $\pi = (Q, \mathcal{C}, \mathcal{A}, q_{init}, T)$ is a labeled digraph $\Gamma_\pi = (V, E, v_{init})$, which has the set of nodes $V = Q \times \mathcal{A}^*$ and the set of labeled arcs $E \subseteq V \times \mathcal{C} \times V$, such that for every pair of nodes $u = (q', s')$ and $v = (q'', s'')$ the following relationship holds

$$(u, c, v) \in E \quad \Longleftrightarrow \quad \exists h \in \mathcal{A}^* \text{ such that } (q', c, q'', h) \in T \text{ and } s'' = s'h.$$

The node $v_{init} = (q_{init}, \varepsilon)$ is called the *initial node* of Γ_π. As it can be seen from the relationship above, the correspondence between trajectories of π and paths in Γ_π is as follows: for every state q_0 and a compound action s_0 a trajectory $tr = \{(q_{i-1}, c_i, q_i, d_i)\}_{i \geqslant 1}$ from q_0 corresponds to such a path $\rho = \{(v_{i-1}, c_i, v_i)\}_{i \geqslant 1}$ in Γ_π that $v_0 = (q_0, s_0)$ and $v_i = (q_i, s_{i-1}h_i)$ for all $i \geqslant 1$. If $tr = \{(q_{i-1}, c_i, q_i, d_i)\}_{i=1}^k$ is an initial finite trajectory then we denote by $v_{init}[tr]$ such a node (q_k, h) of Γ_π that $h = h_1 h_2 \dots h_k$. If $\rho = \{(v_{i-1}, c_i, v_i)\}_{i \geqslant 1}$ is a path in Γ_π then we denote for every $k \geq 0$ by $\rho|^k$ its prefix $\{(v_{i-1}, c_i, v_i)\}_{i=1}^k$.

The verification of information processing systems is a checking that the actual behavior of a system satisfies the expected properties. By choosing finite state transducers for a formal model of sequential reactive system we thus formalize the notion of "behavior" of such systems: a behavior of a transducer π manifests itself in the set of trajectories $Tr_\pi(q_{init})$ of all initial runs of π; this set is represented by the computation tree Γ_π. Any set of trajectories can be regarded as a property of transducers behavior. It is well known that the properties of behaviors represented by infinite sequences of events, as well as discrete structures that combine these sequences, can be conveniently specified by means of temporal logics (LTL, CTL, CTL^*, etc.). However, when specification of transducers behavior is concerned, one should keep in mind that an adequate specification language must admit interpretation over dual sequences of input and output events. The authors of [13] drew attention to this particular feature of formal languages for specifying the behavior of transducers, and they proposed a novel logic $\mathcal{LP}$-LTL intended for reasoning about the behavior of transducers. This logic is an extension of LTL, where temporal operators are parametrized with languages over $\mathcal{C}$ and $\mathcal{A}$. In [13] it was shown that in the case when only regular languages are used as parameters of temporal operators the model checking of finite state transducers against *Reg-LTL* specifications is decidable in double exponential time. Next, in [6] *Reg-CTL* — a regular extension of *CTL* adapted for reasoning about dual sequences of events — was introduced, and it was proved that model checking problem for finite state transducers against *Reg-LTL* specifications is PSPACE-complete. The expressive power of these and some other extensions of temporal logics was studied in [7]. It is quite natural that the next stage of research is the study of the verification problem for the extension of even more general logic CTL^*. In this section we present a temporal logic $\mathcal{LP}$-CTL^* and its regular fragment *Reg*-CTL^*, which is of particular interest for model checking of finite state transducers.

As it was noticed above, the correct behavior of a sequential reactive system depends on how it responds to certain environmental requests. The type of environmental impact on the system can be represented as a language L over the set of input signals, and the type of the response to such a stimulus — as a language P over the set of actions. A language L over $\mathcal{C}$ is called an *environment behavior pattern*, and a language P over $\mathcal{A}$ is called a *basic predicate*.

Suppose that we are given a family $\mathcal{L}$ of environment behavior patterns and a family $\mathcal{P}$ of basic predicates $\mathcal{P}$. The set of $\mathcal{LP}$-CTL^* formulae consists of the subset of *state formulae* and the subset of *path formulae* which are defined as follows:

1) every basic predicate $P \in \mathcal{P}$ is a state formula;

2) if φ_1, φ_2 are state formulae then $\neg\varphi_1$ and $\varphi_1 \wedge \varphi_2$ are state formulae;

3) if ψ is a path formula then $\mathbf{A}\psi$ and $\mathbf{E}\psi$ are state formulae;

4) if φ is a state formula then φ is a path formula;

5) if ψ_1, ψ_2 are path formulae then $\neg\psi_1$ and $\psi_1 \wedge \psi_2$ are path formulae;

6) if $\varphi, \varphi_1, \varphi_2$ are path formulae, $c \in \mathcal{C}$, and $L \in \mathcal{L}$ then $\mathbf{X}_c\varphi$ and $\varphi_1\,\mathbf{U}_L\varphi_2$ are path formulae.

An intuitive meaning of $\mathcal{LP}$-CTL^* formulae is as follows. A basic predicate P holds whenever an output result computed so far by a reactive system is a compound action from the set P. A formula $\mathbf{A}\psi$ (or $\mathbf{E}\psi$) means that every (some) computation of a reactive system satisfies the requirement ψ. A formula $\mathbf{X}_c\varphi$ claims that a reactive system is able to receive the input signal c and its subsequent behavior satisfies the requirement φ. A formula $\varphi_1\,\mathbf{U}_L\varphi_2$ asserts that every time after receiving a sequence w of input signals such that $w \in L$, the behavior of a system satisfies the requirement φ_1 until, upon receipt of an input sequence from L, the behavior of the system meets the requirement φ_2.

Formally, $\mathcal{LP}$-CTL^* formulae are interpreted over computation graphs Γ_π of finite state transducers $\pi = (Q, \mathcal{C}, \mathcal{A}, q_{init}, T)$. By writing $\Gamma_\pi, v \models \varphi$ we indicate that a state formula φ is satisfied at the node v of Γ_π, and by writing $\Gamma_\pi, \rho \models \psi$ we indicate that a path formula ψ is satisfied on a path ρ of Γ_π. The satisfiability relation $\models$ is defined by structural induction on the formulae for every node $v = (q, s)$ in the computation graph Γ_π, and a path ρ in Γ_π assuming that $\rho = (v_0, c_1, v_1), (v_1, c_2, v_2), \dots$ as follows:

1. $\Gamma_\pi, v \models P \iff s \in P$ for every basic predicate $P \in \mathcal{P}$;
2. $\Gamma_\pi, v \models \neg\varphi \iff$ it is not true that $\Gamma_\pi, v \models \varphi$;
3. $\Gamma_\pi, v \models \varphi_1 \wedge \varphi_2 \iff \Gamma_\pi, v \models \varphi_1$ and $\Gamma_\pi, v \models \varphi_2$;
4. $\Gamma_\pi, v \models \mathbf{E}\,\varphi \iff$ there exists such a path ρ from the node v in Γ_π that $\Gamma_\pi, \rho \models \varphi$;
5. if φ is a state formula then $\Gamma_\pi, \rho \models \varphi \iff \Gamma_\pi, v_0 \models \varphi$;
6. $\Gamma_\pi, \rho \models \neg\psi \iff$ it is not true that $\Gamma_\pi, \rho \models \psi$;
7. $\Gamma_\pi, \rho \models \psi_1 \wedge \psi_2 \iff \Gamma_\pi, \rho \models \psi_1$ and $\Gamma_\pi, \rho \models \psi_2$;
8. $\Gamma_\pi, \rho \models \mathbf{X}_c\varphi \iff c = c_1$ and $\Gamma_\pi, \rho|^1 \models \varphi$;
9. $\Gamma_\pi, \rho \models \varphi\,\mathbf{U}_L\psi \iff \exists i \geqslant 0$: $c_1c_2\dots c_i \in L$ such that $\Gamma_\pi, \rho|^i \models \psi$ and $\forall j,\ 0 \leqslant j < i$, if $c_1c_2\dots c_i \in L$ then $\Gamma_\pi, \rho|^j \models \varphi$.

In the case when $v = v_{init}$ we will write $\pi \models \varphi$ instead of $\Gamma_\pi, v \models \varphi$. In the definition above environment behavior patterns and basic predicates may be arbitrary languages over the alphabet of signals and the alphabet of actions, respectively. As one might expect, this freedom of choice leads to undecidability. For example, let $\mathcal{C}$ be an alphabet of two or more letters, and $\mathcal{A} = \mathcal{C}$. Consider such a transducer π that $T = \{(q_{init}, c, c, q_{init}) : c \in \mathcal{C}\}$, i.e. π just retransmits the received signals. Then, for any pair of context-free languages U and V over $\mathcal{C}$ it is true that $\pi \models \mathbf{EF}_U V$ iff $U \cap V \neq \varnothing$. Since the emptiness of intersection problem for context-free languages is undecidable (see [10]), the problem of checking whether a transducer π satisfies a $\mathcal{LP}$-CTL^* formula φ (the *model checking problem* $\pi \models \varphi$) is also undecidable when $\mathcal{L}$ and $\mathcal{P}$ are classes of context-free languages.

In order to find an effective solution to the model checking problem for $\mathcal{LP}$-CTL^* thus introduced, we restrict ourselves to more simple classes of environment patterns and basic predicates. In Section 4 we consider a fragment of $\mathcal{LP}$-CTL^*, namely, *Reg*-CTL^*, where all environment behavior patterns and all basic predicates are *regular languages.* But first, it is worthwhile to briefly compare the logic $\mathcal{LP}$-CTL^* introduced here with other previously known extensions of conventional temporal logics.

3. Other extensions of temporal logics

In [5] the first comprehensive analysis of LTL as a formal language for describing the behavior of computing systems was carried out, and several attempts were made shortly thereafter to improve the expressive power of this temporal logic. The modifications were made mainly in two directions: 1) adding new expressive means (quantifiers, modalities, etc.), and 2) changing the semantics of temporal operators existing in LTL.

For example, the authors of [15] supplied the syntax of LTL with quantification for basic predicates and discovered that the expressive power of the quantified extension thus introduced significantly exceeds the descriptive capabilities of the plain LTL. On the other hand, in [20] a method for introducing new temporal operators using right-linear grammars was proposed. 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. Similarly, in [14, 18] it was shown that the patterns for describing the semantics of new temporal operators can be defined by means of finite automata. The idea of supplying temporal operators with some parameters is not new: almost the same parameterization of temporal operators as in this paper was introduced in [9] for dynamically extend LTL. Since then several attempts of this kind were made to merge regular languages and temporal operators (e.g. see [16] among the latest). In almost all these cases a remarkable effect was found: an expressive power of such extensions increases and becomes equivalent to that of $S1S$ logic, while satisfiability checking problem for these logics remains PSPACE-complete.

When introducing $\mathcal{LP}$-CTL^*, we did not seek only to improve the expressive power of CTL^* as such. Our goal was to offer an adequate language for the specification of the behavior of reactive systems modeled by transducers. In the computation of a transducer, a coordinated formation of two sequences is carried out — a sequence of input signals and a sequence of output actions. Therefore, a distinctive feature of $\mathcal{LP}$-CTL^* semantics is a certain synchronization of the parameters of temporal operators (their interpretation is determined by the sequence of input signals) and the truth values of basic predicates (they depend on the sequence of output actions). It could be said that the semantics of our logic is defined on traces in two-dimensional space, while in all previously known parameterized extensions of temporal logics only one-dimensional traces were used. This feature significantly affects algorithms for checking the satisfiability of formulas.

4. Model checking of finite state transducers against *Reg*-CTL^* specifications

The study of model checking problem for finite state transducers was first initiated in [13]. The authors of this paper considered the case when specifications are given by *Reg-LTL* formulae. This logic is an extension of the well-known temporal logic LTL and it is also *linear* fragment of *Reg*-CTL^* which contains only the formulae of the type $\mathbf{A}\varphi$, where φ is a quantifier-free path formula. The model checking procedure developed in [13] is based on a translation of a pair (π, φ) to a Büchi automaton $B(\pi, \varphi)$ such that $\pi \models \varphi$ iff $B(\pi, \varphi) \neq \varnothing$.

In [6] the study of the model checking problem for finite state transducers was continued. The authors of this paper considered the temporal logic *Reg*-CTL. This logic is an extension of another well-known temporal logic CTL; and it is also another fragment of *Reg*-CTL^* which consists of those formulae where each temporal operator $\mathbf{F}, \mathbf{G}$ or $\mathbf{U}$ is preceded by a path quantifier $\mathbf{E}$ or $\mathbf{A}$.

Model checking algorithms for *Reg-LTL* and *Reg*-CTL developed in [6, 13] follow respectively the automata-theoretic and tableau-based approaches used for the solution of model checking problem for conventional temporal logics LTL and CTL. In [4] it was shown (see also [3]) that model checking problem for Extended Computational Tree Logic CTL^* can be solved with essentially the same complexity as LTL, using a combination of the algorithms for LTL and CTL. However, when model checking of transducers against *Reg*-CTL^* formulae is concerned some specific features of transducers behavior make it impossible a straightforward application of this combination techniques; model checking of *Reg*-CTL^* specifications of finite state transducers needs a far elaborate study. In this section we present a model checking algorithm for *Reg*-CTL^* which is based on an iterated translation of a *Reg*-CTL^* formula into Büchi automata.

4.1. Finite automata and Büchi automata

A deterministic *finite state automaton* (DFA) over an alphabet Σ is a quintuple $A = (Q, \Sigma, q_{init}, \delta, F)$, where Q is a finite set of states, $q_{init} \in Q$ is an initial state, $F \subseteq Q$ is a set of *accepting states* and $\delta\colon Q \times \Sigma \mapsto Q$ is a transition function. This function can be extended to the set of words Σ^* as follows: $\delta(q, \varepsilon) = q$, and $\delta(q, \sigma x) = \delta(\delta(q, \sigma), x)$ for all $q \in Q, \sigma \in \Sigma$ and $x \in \Sigma^*$. A DFA A *accepts* a word x (we will write $x \in A$ to denote this fact) iff $\delta(q_{init}, x) \in F$. A DFA A *recognizes* a language $L(A) = \{x : x \in A\}$ of all words it accepts.

A DFA $A[x] = (Q, \Sigma, \delta(q_{init}, x), \delta, F)$ is called a *shift* of a DFA A *by* a word x. The *size* $|A|$ of an automaton A is the size of a table description of its transition function δ. By the size $|L|$ of a regular language L we mean the size $|A|$ of the *minimal* DFA A which recognizes L.

In addition to deterministic automata, we will also use *nondeterministic* finite state automata (NFA) for manipulations with regular languages. NFA have transition functions of the type $Q \times \Sigma \mapsto 2^Q$, and its extension to Σ^* is defined by the equalities $\delta(q, \varepsilon) = \{q\}$, and $\delta(q, \sigma x) = \bigcup_{q' \in \delta(q,\sigma)} \delta(q', x)$. A NFA A accepts a word x iff $\delta(q^A_{init}, x) \cap F \neq \varnothing$ holds.

Finite state automata can be further extended to allow recognition of sets of *infinite* words over Σ. A nondeterministic generalized *Büchi automaton* over an alphabet Σ is a quintuple $B = (Q, \Sigma, q_{init}, \Delta, \mathcal{F})$, where Q is a finite set of states, q_{init} is an initial state, $\Delta \subseteq Q \times \Sigma \times Q$ is a *transition relation*, and $\mathcal{F} = \{F_1, \dots, F_m\}$ is an *accepting rule*, where $F_i \subseteq Q, 1 \leqslant i \leqslant m$. For every infinite word $x = \sigma_0\sigma_1\sigma_2 \dots \in \Sigma^\omega$, a *run* of B on x is an infinite sequence $q_0, q_1, q_2, \dots$ of states of B, where $q_0 = q_{init}$ and for all $i, i \geqslant 0$, $(q_i, \sigma_i, q_{i+1}) \in \Delta$. A run $q_0, q_1, q_2, \dots$ is *accepting* if for all $i, 1 \leqslant i \leqslant m$, there exist infinitely many j, such that $q_j \in F_i$. A word $x \in \Sigma^\omega$ is *accepted* by B (we write $x \in B$ to denote this fact) iff there exists an accepting run of B on x. In [3] it was shown that there exists a linear-time algorithm for checking, given a Büchi automaton B, iff $B = \varnothing$.

4.2. Translation of *Reg-CTL** formulae to automata

Consider a finite state transducer $\pi = (Q, \mathcal{C}, \mathcal{A}, q_{init}, T)$ and a *Reg-CTL** formula φ. A *model checking automaton* is such a DFA $A(\pi, \varphi) = (Q_A, T, q^A_{init}, \delta_A, F_A)$ over the finite alphabet T of transitions of π which satisfies for any finite trajectory $tr \in \mathrm{Pref}(Tr_\pi(q_{init}))$ the following relationship: $\Gamma_\pi, v_{init}[tr] \models \varphi \iff tr \in A(\pi, \varphi)$.

Clearly, having constructed such a DFA $A(\pi, \varphi)$, we obtain a solution to the model checking problem, since $\pi \models \mathbf{E}\varphi$ iff $\varepsilon \in A(\pi, \varphi)$. The main result of the paper is

Theorem 1. For every finite state transducer π and a state *Reg-CTL**-formula φ the model-checking automaton $A(\pi, \varphi)$ can be effectively constructed within a memory space $|\pi| \cdot 2^{\mathrm{poly}(|\varphi|)}$.

Proof. From the definition of satisfiability relation for *Reg-CTL** formulae it follows that $\models P \equiv \mathbf{E}P$ holds for every basic predicate P. Therefore, it may be assumed without loss of generality that every occurrence of a basic predicate in φ follows a path quantifier $\mathbf{E}$.

Next, it should be noted that every state *Reg-*CTL^*-formula φ can be attributed to one of three types of formulas:

1. $\varphi = \mathbf{E}P$ for some basic predicate P;
2. $\varphi = \Phi(\mathbf{E}\varphi_1, \dots, \mathbf{E}\varphi_m)$ for some Boolean formula $\Phi(p_1, \dots, p_m)$;
3. $\varphi = \mathbf{E}\psi(\mathbf{E}\varphi_1, \dots, \mathbf{E}\varphi_m)$ for some quantifier-free path *Reg-*CTL^*-formula $\psi(P_1, \dots, P_m)$.

Therefore, the construction of the model checking automaton $A(\pi, \varphi)$ is carried out recursively.

1. For every regular basic predicate P the model checking automaton $A(\pi, \mathbf{E}P)$ can be easily built of a transducer π and a DFA A_P which recognizes P.
2. For every Boolean formula $\Phi(p_1, \dots, p_m)$ a model checking automaton $A(\pi, \varphi)$ for a state *Reg-*CTL^*-formula $\varphi = \Phi(\mathbf{E}\varphi_1, \dots, \mathbf{E}\varphi_m)$ can be built as a Boolean combination of model checking automata $A(\pi, \mathbf{E}\varphi_i)$ corresponding to subformulae $\mathbf{E}\varphi_i, 1 \leq i \leq m$.
3. Suppose that $\varphi = \mathbf{E}\psi(\mathbf{E}\varphi_1, \dots, \mathbf{E}\varphi_m)$, where $\psi(P_1, \dots, P_m)$ is some quantifier-free path *Reg-*CTL^*-formula, and model checking automata $A(\pi, \mathbf{E}\varphi_i), 1 \leq i \leq m$, are available. Then regular languages $L(A(\pi, \mathbf{E}\varphi_i))$ recognized by these automata are regarded as basic predicates $P_1, \dots, P_m$, and model-checking automaton $A(\pi, \varphi)$ is built as follows:
   - by applying a model checking algorithm proposed in [13] build a Büchi automaton $B_{\psi(P_1,\dots,P_m)}$ which accepts a trace $tr \in Tr_\pi$ iff tr corresponds to such a path ρ that $\Gamma_\pi, \rho \models \psi(P_1, \dots, P_m)$;
   - next, by combining a Büchi automaton $B_{\psi(P_1,\dots,P_m)}$ and a transducer π build such a Büchi automaton $B(\pi, \varphi)$ that for any finite trajectory tr the following relationship holds: $\Gamma_\pi, v_{init}[tr] \models \mathbf{E}\psi(P_1, \dots, P_m)$ iff there exists an accepting run of $B(\pi, \varphi)$ on some trace tr' which is an extension of tr;
   - finally, by using a simple reachability checking techniques build a model checking automaton $A(\pi, \varphi)$ from $B(\pi, \varphi)$.

Corollary. The model checking problem for finite state transducers against *Reg-*CTL^* specifications is in EXPSPACE.

Theorem 2. The model checking problem for finite state transducers against *Reg-*CTL^* specifications is PSPACE-hard.

Proof (sketch). In [12] it was proved that it is PSPACE-hard to check whether the intersection of an arbitrary number of deterministic finite state automata is empty. This problem can be reduced to our model checking problem in polynomial time. □

5. Conclusion

In this paper we introduced an extension *Reg-CTL** of Generalized Computational Tree Logic *CTL** as a formal language for specification the behavior of sequential reactive systems, define its semantics on the models of finite state transducers, and give a solution to the model checking problem for finite state transducers against formulae from *Reg-CTL**. The solution is obtained by means of automata-theoretic techniques following the ideas of our model checking algorithms for and *Reg-LTL* and *Reg-CTL* presented in the early papaers [8**?**]. It may be noticed that this model checking algorithm is exponentially more time consuming than the similar algorithm for plain *CTL** (see [**?**]). However, there remains an exponential gap between the lower and the upper bounds on the complexity of the model checking problem for *Reg-CTL**. We assume that this gap can be filled by improving the proposed algorithm using the promising techniques suggested in [9], where a logic similar to *Reg-LTL* was studied.

The authors of the article are grateful to the anonymous reviewers for useful comments that helped to improve the article.

Список литературы

1. Гнатенко А. Р., Захаров В. А. О сложности верификации автоматов-преобразователей над коммутативными полугруппами // Материалы XVIII Международной конференции "Проблемы теоретической кибернетики". 2017. С. 68–71.
2. Alur R., Cerny P. Streaming transducers for algorithmic verification of single-pass list-processing programs // Proceedings of the 38-th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages. 2011. P. 599–610.
3. Clarke E.M., Gramberg O., Kroening D., Peled D.A., Veith H. Model Checking. MIT Press. 2018. 424 p.
4. Emerson E.A., Lei C.-L. Modalities for model checking: branching time logic strikes back // Science of Computer Programming. 1987. Vol. 8, № 3. P. 275–306.
5. Gabbay D., Pnueli A., Shelach S., Stavi J. The temporal analysis of fairness // Proceedings of the 7-th ACM Symposium on Principles of Programming Languages. 1980. P. 163–173.
6. Gnatenko A.R., Zakharov V.A. On the model checking of finite state transducers over semigroups // Proceedings of ISP RAS. 2018. Vol. 30, № 3, P. 303–324
7. Gnatenko A.R., Zakharov V.A. On the expressive power of some extensions of Linear Temporal Logic // Automatic Control and Computer Sciences. 2019. Vol. 53, № 7, P. 506–524.
8. Gnatenko A.R. On the complexity of model checking problem for finite state transducers over free semigroups. // Proceedings of the Student Session of European Summer School on Logic, Language and Information, Riga, 2019.

9. Henriksen J.J., Thiagarajan P.S., Dynamic linear time temporal logic // Annals of Pure and Applied Logic. 1999. Vol. 96, № 1–3. P. 187–207.
10. Hopcroft J.E., Ullman J.D. Introduction to Automata Theory, Languages, and Computation (1st ed.). Addison-Wesley. 1979.
11. Hu Q., D'Antoni L. Automatic Program Inversion using Symbolic Transducers // Proceedings of the 38-th ACM SIGPLAN Conference on Programming Language Design and Implementation. 2017. P. 376–389.
12. Kozen D. Lower bounds for natural proof systems. // Proceedings of the 18-th Symposium on the Foundations of Computer Science. 1977. P. 254–266.
13. Kozlova D.G., 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, P. 233–244.
14. Kupferman O., Piterman N., Vardi M.Y. Extended temporal logic revisited // Proceedings of the 12-th International Conference on Concurrency Theory. 2001. P. 519–535.
15. Manna Z., Wolper P. Synthesis of communicating processes from temporal logic specifications // ACM Transactions on Programming Languages and Systems. 1984. Vol. 6, № 1. P. 68–93.
16. Mateescu R., Monteiro P.T., Dumas E., De Jong H. CTRL: Extension of CTL with regular expressions and fairness operators to verify genetic regulatory networks // Theoretical Computer Science. 2011. Vol. 412, № 26. P. 2854–2883.
17. Savitch W.J. Relationships between nondeterministic and deterministic tape complexities. // Journal of Computer and System Sciences, 1970. Vol. 4, № 2.
18. Vardi M.Y., Wolper P. Yet another process logic // Proceedings of the Carnegie Mellon Workshop on Logic of Programs. 1983. P. 501–512.
19. Veanes M., Hooimeijer P., Livshits B., et al. Symbolic finite state transducers: algorithms and applications // Proceedings of the 39-th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, ACM SIGPLAN Notices. 2012. Vol. 147. P. 137–150.
20. Wolper P. Temporal logic can be more expressive // Information and Control. 1983. Vol. 56, №№ 1–2, P. 72–99.