On the model checking of finite state transducers over semigroups

[1] *A.R. Gnatenko<gnatenko.cmc@gmail.com>*
[2] *V.A. Zakharov<zakh@cs.msu.su>*
[1] *Lomonosov Moscow State University,*
GSP-1, Leninskie Gory, Moscow, 119991, Russia
[2] *National Research University High School of Economics,*
20, Myasnitskaya str., Moscow, 101000, Russia

Abstract. Sequential reactive systems represent programs that interact with the environment by receiving signals or requests and react to these requests by performing operations with data. Such systems simulate various software like computer drivers, real-time systems, control procedures, online protocols etc. In this paper, we study the verification problem for the programs of this kind. We use finite state transducers over semigroups as formal models of reactive systems. We introduce a new specification language LP-CTL* to describe the behavior of transducers. This language is based on the well-known temporal logic CTL* and has two distinguished features: 1) each temporal operator is parameterized with a regular expression over input alphabet of the transducer, and 2) each atomic proposition is specified by a regular expression over the output alphabet of the transducer. We develop a tabular algorithm for model checking of finite state transducers over semigroups against LP-CTL* formulae, prove its correctness, and estimate its complexity. We also consider a special fragment of LP-CTL* language, where temporal operators are parameterized with regular expressions over one-letter alphabet, and show that this fragment may be used to specify usual Kripke structures, while it is more expressive than usual CTL*.

Keywords: reactive program**;** transducer**;** verification**;** model checking**;** temporal logic**;** finite state automaton**;** regular language

DOI: 10.15514/ISPRAS-2018-30(3)-21

For citation: Gnatenko A.R., Zakharov V.A. On the model checking of finite state transducers over semigroups. Trudy ISP RAN/Proc. ISP RAS, vol. 30, issue 3, 2018, pp. 303-324. DOI: 10.15514/ISPRAS-2018-30(3)-21

1. Introduction

Finite state machines are widely used in the field of computer science and formal methods for various purposes. While finite automata represent regular sets, *transducers* stand for regular (or, rational) *relations* and, therefore, can serve as models of programs and algorithms that operate with input and output data. For

example, transducers are used as formal models in software engineering to represent numerous algorithms, protocols and drivers that manipulate with strings, dataflows, etc. [1, 15, 25].

By extending the concept of ordinary transducers, we build a new formal model for sequential reactive systems. These systems are software programs or hardware devices that receive requests (control signals, commands) from the environment and perform in response some manipulations (actions, transformations) with data, interactions with the environment, mechanical movements, etc. While the flow of requests can be represented by finite or infinite words in some fixed alphabet, the sequence of actions of the system needs a more sophisticated interpretation. The key point here is that different sequences of actions may bring a computing system to the same result. To capture this effect the collection of actions performed by a reactive system can be viewed as a generating set of some algebraic structure (e.g. semigroup, group, ring, etc.) and particular algebraic properties of basic actions should be taken into account when choosing adequate formal models for this class of information processing systems. Let us illustrate this consideration by several examples.

- A network switch with several input and output ports. A switch is a device, which receives data packets on its input port, modifies their heads and commutes them to one of the output ports. Once received a special control signal, this switch changes its packet-forwarding table and, thus, its behaviour. Since packets from different flows may be processed in any order, the switch can be modeled by a transducer, which operates over a free partially commutative semigroup, or a trace monoid. Trace monoids are commonly used as an algebraic foundation of concurrent computations and process calculi (see, e.g., [9]).
- A real-time device that control the operation of some industrial equipment (say, a boiling system). Such device receives data from temperature and pressure sensors and switches some processes on and off according to its instructions and the current state of the system. It seems reasonable that for some actions the order of their implementation is not important (routine actions), while others must follow in a strictly specified order (e.g. an execution of some complex operation). Moreover, there are also actions which bring system to certain predefined operation mode (set-up actions). These actions are implemented in the emergency situations. A partially commutative semigroup with right-zero elements $\mathbf{0}$ which satisfy the equalities $x\mathbf{0} = \mathbf{0}$ for every element x provides an adequate interpretation for such operations.
- A system supervisor that maintains a log-file. For each entry its date and time is recorded in the file and there is no way to delete entries from the log – only to append it. Thus, for any two basic actions (record operations to the log-file) it is crucial in which order they are performed and such a supervisor can be modeled by a transducer over a free semigroup. Verification techniques for such reactive

systems are considered in [17]; this is the main topic of this paper as well.

- A radio-controlled robot, that moves on a surface. It can make one step moves in any of direction. When it receives a control signal in a state q' it must choose and carry out a sequence of steps and enter to the next state q''. At some distinguished state q_f the robot reports its current location. Movements of the robot may be regarded as basic actions, and the simplest model of computation which is suitable for analyzing a behaviour of this robot is a nondeterministic finite state transducer operating on a free Abelian group of rank 2.

To construct a reliable system or network it is crucial for its components to have a correct behaviour. For example, a network switch must process received data packets within a specified number of execution steps and the boiling system should never be overheated, that is, will never remain for a long time in a particular condition without appropriate responses from the control device. By using finite state transducers as formal models of reactive systems, one can develop verification algorithms for these models to solve such problems as equivalence checking, deductive verification or model checking.

The study of the equivalence checking problem for classical transducers began in the early 60s. It was established that the equivalence checking problem for non-deterministic transducers is undecidable [13] even over 1-letter input alphabet [16]. However, the undecidability displays itself only in the case of unbounded transductions when an input word may have arbitrary many images. The equivalence checking problem was shown to be decidable for deterministic [4], functional (single-valued) [5, 19], and k-valued transducers [6, 26]. In a series of papers [20-22] techniques for checking bounded valuedness, k-valuedness and equivalence of finite state transducers over words were developed. Recently in [29] equivalence checking problem was shown to be decidable for finite state transducers that operate over finitely generated semigroups embeddable in decidable groups.

There are also papers where equivalence checking problem for transducers is studiedin the framework of program verification. The authors of [23] proposed models of communication protocols as finite state transducers operating on bit strings. They set up the verification problem as equivalence checking between the protocol transducer and the specification transducer. The authors of [25] extended finite state transducers with symbolic alphabets, which are represented as parametric theories. They showed that a number of classical problems for extended transducers, including equivalence checking problem, are decidable modulo underlying theories. In [1] a model of streaming transducers was proposed for programs that access and modify sequences of data items in a single pass. It was shown that a number of verification problems such as equivalence checking, assertion checking, and checking correctness with respect to pre/post conditions, are decidable for this program model.

Meanwhile, very few papers on the model checking problem for transducers are known. Transducers can be conveniently used as auxiliary means in regular model

checking of parameterized distributed systems where configurations are represented as words over a finite alphabet. In such models, a transition relation on these configurations may be regarded as a rational relation and, thus, it may be specified by finite state transducers (see [7, 28]). In these papers, finite state transducers just play the role of verification instrument, but not an object of verification. However, as far as we know, a deeper investigation of the model checking problem for the reactive systems represented by transducers has not yet been carried out. We think that this is due the following main reason. A transducer is a model of computation which, given an input word, computes an output word. The letters of input and output alphabets can be regarded as valuations (tuples of truth values) of some set of basic predicates. Therefore, a transducer can be viewed as some special representation of a labeled transition system (Kripke structure) (see [2]). From this viewpoint model checking problem for finite state transducers conforms well to standard model checking scheme for finite structures, and, hence, it is not worthy of any particular treatment.

However, our viewpoint is quite different. Transducer is a more complex model of computation than a finite state automaton (transition systems). Its behaviorism characterized by the correspondence between input and output words. A typical property of such behaviour to be checked is whether for every (some) input word from a given pattern a transducer outputs a word from another given pattern. Therefore, when formally expressing the requirements of this kind one needs not only temporal operators to specify an order in which events occur but also some means to refer to such patterns. Conventional Temporal Logics like $\boldsymbol{LTL}$ or $\boldsymbol{CTL}$ are not sufficient in this case; they should be modified in such a way as to acquire an ability to express correspondences between the sets (languages) of input words and the sets (languages) of output words. This could be achieved by supplying temporal operators with patterns as parameters. Every such pattern is a formal description of a language L over an input alphabet$\mathcal{C}$; automata, formal grammars, regular expressions, language equations are suitable for this purpose. The basic properties of output words can be also represented by languages over an output alphabet. Then, for instance, an expression $\boldsymbol{G}_L P$ can be understood as the requirement that for every input word w from the language L the output word h of a transducer belongs to the language P.

The advantages of this approach are twofold. On the one hand, such extensions of Temporal Logics make it possible to express explicitly relationships between input and output words and specify thus desirable behaviours of transducers. On the other hand, it can be hoped that such extensions could rather easily assimilate some well-known model checking techniques (see [3, 8]) developed for conventional Temporal Logics. The first attempt to implement this approach was made in [17]. The authors of this paper introduced an $\boldsymbol{\mathcal{LP}\text{-}LTL}$ specification language based on $\boldsymbol{LTL}$ temporal logic and developed a checking procedure for transducers over free monoids against specifications from $\boldsymbol{\mathcal{LP}\text{-}LTL}$. It was shown that this procedure has double exponential time complexity.

In this paper we continue this line of research and "raise" the specification language introduced in [17] to the level of $\mathcal{LP}$-$\mathbf{CTL}^*$. We will focus only on one task related to the use of new logic for the verification of reactive systems, namely, the development of a general model checking algorithm for finite state transducers against specifications in $\mathcal{LP}$-$\mathbf{CTL}^*$. Such issues as expressive power of $\mathcal{LP}$-$\mathbf{CTL}^*$, complexity of model checking and satisfiability checking problems, the influence of types of languages used as parameters and basic predicates in $\mathcal{LP}$-$\mathbf{CTL}^*$ on decidability and complexity of model checking problem remain topic of our further research and will be covered in our subsequent works. We also leave beyond this work a number of applied questions, which are worthy of consideration in a separate paper. For example, it is important to understand to what extent the already developed model checking tools can be adapted to the new temporal logic. And, of course, in the future we will have a well-chosen series of examples that illustrate the new possibilities of using $\mathcal{LP}$-$\mathbf{CTL}^*$to describe the behavior of reactive systems.

The paper is organized as follows. In Section 2, we define the concept of finite state transducer over semigroup as a formal model of sequential reactive systems (see [29]) and in Section 3, we describe the syntax and the semantics of $\mathcal{LP}$-$\mathbf{CTL}^*$ as a formal language for specifying behaviour of sequential reactive systems. In Section 3 we also set up formally model checking problem for finite state transducers against $\mathcal{LP}$-$\mathbf{CTL}^*$ formulae. In Section 4, we present an $\mathcal{LP}$-$\mathbf{CTL}^*$ model checking algorithm for the case when parameters of temporal operators and basic predicates are regular languages represented by finite state automata. The model checking algorithm we designed has time complexity which is linear of the size of a transducer but exponential of the size of $\mathcal{LP}$-$\mathbf{CTL}^*$ formula. This complexity estimate is in contrast to the case of conventional $\mathbf{CTL}$ model checking: its time complexity is linear of both the size of a model and the size of a $\mathbf{CTL}$ formula. To explain this effect in Section 5 we show how $\mathcal{LP}$-$\mathbf{CTL}^*$ formulae can be also checked on the conventional Kripke structures. Finally, we compare $\mathcal{LP}$-$\mathbf{CTL}^*$ with some other known extensions Temporal Logics and discuss some topics for further research.

2. Finite state transducers as models of reactive systems

In this section, we introduce a Finite State Transducer as a formal model of a reactive computing system which receives control signals from the environment and reacts to these signals by performing operations with data.

Let $\mathcal{C}$ be a finite set of *signals*. Finite words over $\mathcal{C}$ are called *signal flows*; the set of all signal flows is denoted by$\mathcal{C}^*$. Given a pair of signal flows u and v we write uv for their concatenation, and denote by ε the empty flow.

Let $\mathcal{A} = \{a_1, \dots, a_n\}$ be a finite set of elements called *basic actions*; these actions stand for the elementary operations performed by a reactive system. Finite words over $\mathcal{A}$ are called *compound actions*; they denote sequential compositions of basic actions. Since different sequences of basic actions could produce the same result,

one may interpret compound actions over a semigroup $(S,\ e,\ \circ)$ generated by a set of basic actions $\mathcal{A}$. The elements of S are called *data states*. Every compound action $h = a_{i_1}a_{i_2}\ldots a_{i_k}$ is evaluated by the data state $[h] = [a_{i_1}] \circ [a_{i_2}] \circ \ldots \circ [a_{i_k}]$. For example, if a reactive system just keeps a track of input requests by adding certain records to a log-file then a free semigroup will be suitable for interpretation of these operations. But when a robot moves on a 2-dimensional surface then the actions (movements) performed by this robot may be regarded as generating elements of Abelian group G of rank 2, and the positions on the surface occupied by this robot can be specified by the elements from G. In this paper we restrict ourselves to the consideration of free semigroups when $[h] = h$ holds for every compound action h, and $\circ$ is the word concatenation operation.

Let $\mathcal{C}$ be a set of signals and $\mathcal{A}$ be a set of basic actions that are interpreted over a semigroup $(S, e, \circ)$. Then a Finite State Transducer (in what follows, FST) 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 an *initial control state*;
- $T \subseteq Q \times \mathcal{C} \times Q \times \mathcal{A}^*$is a finite *transition relation*.

Each tuple (q', c, q'', h) in T is called a *transition*: when a transducer is in a control state q' and receives a signal c, it changes its state to q'' and performs a compound action h. We denote such transition by $q' \xrightarrow{c,h} q''$. A *run* of a FST Π is any finite sequence of transitions

$$q_1 \xrightarrow{c_1,h_1} q_2 \xrightarrow{c_2,h_2} q_3 \xrightarrow{c_3,h_3} \cdots \xrightarrow{c_n,h_n} q_{n+1};$$

this run transduces a signal flow $w = c_1c_2\ldots c_n$ into a data state $[h_1h_2\ldots h_n]$.

The behaviour of a FST $\Pi = (Q,\ \mathcal{C}, \mathcal{A}, q_{init}, T)$ over a semigroup (data space) $(S, e, \circ)$ is presented formally by a *transition system* $TS(\Pi, S) = (D, \mathcal{C}, d_{init}, \mathcal{T})$, where

- $D = Q \times S$ is (in general case, infinite) set of *states of computation*,
- $d_{init} = (q_{init}, e)$ is the *initial state*, and
- $\mathcal{T} \subseteq D \times \mathcal{C} \times D$ is a *transition relation* such that for every states of computation $d' = (q', s'), d'' = (q'', s'')$ and every signal $c \in \mathcal{C}$ the relationship

  $$(d', c, d'') \in \mathcal{T} \ \Rightarrow\ \exists h \in \mathcal{A}^*(q', c, q'', h) \in T \text{ and } s'' = s' \circ [h]$$

  holds.

As usual, a transition $(d', c, d'') \in \mathcal{T}$ is denoted by $d' \xrightarrow{c} d''$.

A *trajectory* in a transition system $TS(\Pi, S)$ is a pair $tr = (d_0, \alpha)$, where $d_0 \in D$ and $\alpha = (c_1, d_1), (c_2, d_2), \ldots, (c_i, d_i), \ldots$ is a sequence of pairs (c_i, d_i) such that $d_{i-1} \xrightarrow{c_i} d_i$ holds for every $i, i \geq 1$. A trajectory represents a possible scenario of a behaviour of a sequential reactive system: when receiving a signal flow

$c_1, c_2, \dots, c_i, \dots$ the reactive system performs a sequence of basic actions h and follows sequentially via the states of computation $d_1, d_2, \dots, d_i, \dots$. By $tr|^i$ we mean the trajectory$(d_i, \alpha|^i)$, where $\alpha|^i = (c_{i+1}, d_{i+1}), (c_{i+2}, d_{i+2}), \dots$ is a suffix of α, respectively.

3. $\mathcal{LP}$-CTL^* specification language

When designing sequential reactive systems one should be provided with a suitable formalism to specify the requirements for their desirable behaviour. For example, one may expect that

- a mobile robot, receiving an equal number of control signals "go_left" and "go_right", will always return to its original position,
- a network switch will never commute data packets from different packet flows into the same output buffer,
- it is not possible for the interrupt service routine to complete the processing of one interrupt before it receives a request to handle another.

These and many other requirements which refer to the correspondences between control flows and compound actions in the course of FST runs can be specified by means of Temporal Logics. When choosing a suitable temporal logic as a formal specification language of FST behaviours one should take into account two principal features of our model of sequential reactive systems:

- since a FST operates over a data space which is semigroup, the basic predicates must be interpreted over semigroups as well, and
- since a behaviour of a FST depends not on the time flow itself but on a signal flow which it receives as an input, temporal operators must be parameterized by certain descriptions of admissible signal flows.

To adapt traditional temporal logic formalism to these specific features of FST behaviours the authors of [17] introduced a new variant of Linear Temporal Logic (***LTL***). We assume that in general case one may be interested in checking the correctness of FST's responses to arbitrary set of signal flows. Every set of control flows may be regarded as a language over the alphabet $\mathcal{C}$ of signals. Therefore, it is reasonable to supply temporal operators ("globally" $\boldsymbol{G}$, "eventually" $\boldsymbol{F}$, etc.) with certain descriptions of such languages as parameters. In more specific cases we may confine ourselves with considering only a certain family of languages (finite, regular, context-free, etc.) $\mathcal{L}$ used as parameters of temporal operators. These languages will be called *environment behaviour patterns*.

A reactive system performs finite sequences of basic actions in response to control signals from the environment and thus follows in the course of its run via a sequence of data states, which are elements of a semigroup $(S, e, \circ)$, Therefore, basic predicates used in ***LTL*** formulae may be viewed as some sets of data states S', $S' \subseteq S$. These sets can be also specified in language-theoretic fashion.

Any language P over the alphabet of basic actions $\mathcal{A}$ corresponds to a predicate (set of data states) $S_P = \{[h] \mid h \in P\}$. As in the case of environment behaviour patterns we may distinguish a certain class $\mathcal{P}$ of languages and use them as specifications of basic predicates. When these languages are used as parameters in temporal formulae then it will be assumed that they are defined constructively by means of automata, grammars, Turing machines, etc.

Thus, we arrive at the concept of $\boldsymbol{\mathcal{LP}}$-variants of Temporal Logics. In [17] the syntax and semantics of $\boldsymbol{\mathcal{LP}}$-**LTL** was studied in some details in the case when both environment behaviour patterns and basic predicates are regular languages presented by finite automata. In this paper we make one step further and extend the concept of $\boldsymbol{\mathcal{LP}}$-variants of Temporal Logics to $\boldsymbol{CTL}^*$. Select an arbitrary family of environment behaviour patterns $\mathcal{L}$ and a family of basic predicates $\mathcal{P}$. The set of $\boldsymbol{\mathcal{LP}}$-$\boldsymbol{CTL}^*$ formulae consists of *state formulae* and *trajectory formulae*, which are defined as follows:

- each basic predicate $P \in \mathcal{P}$ is a state formula;
- if φ_1, φ_2 are state formulae then $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$ and $\varphi_1 \vee \varphi_2$ are state formulae;
- if ψ is a trajectory formula then $\boldsymbol{A}\psi$ and $\boldsymbol{E}\psi$ are state formulae;
- if φ is a state formula then φ is a trajectory formula;
- if ψ_1, ψ_2 are trajectory formulae then $\neg\psi_1$, $\psi_1 \wedge \psi_2$ and $\psi_1 \vee \psi_2$ are trajectory formulae;
- if φ, φ_1, φ_2 are state formulae, $c \in \mathcal{C}$, and $L \in \mathcal{L}$ then $\boldsymbol{X}_c\varphi$, $\boldsymbol{Y}_c\varphi$, $\boldsymbol{F}_L\varphi$, $\boldsymbol{G}_L\varphi$ and $\varphi_1 \boldsymbol{U} \varphi_2$ are trajectory formulae.

The specification language $\boldsymbol{\mathcal{LP}}$-$\boldsymbol{CTL}^*$ is the set of all state formulae constructed as defined above.

Now we introduce the semantics of $\boldsymbol{\mathcal{LP}}$-$\boldsymbol{CTL}^*$ formulae. These formulae are interpreted over transition systems. Let $M = TS(\Pi, S)$ be a transition system, d be a state of computation in this system, and tr be a trajectory in M. Then for every state formula φ we write $M, d \vDash \varphi$ to denote the fact that the assertion φ is true in the state d of M, and for every trajectory formula ψ we write $M, tr \vDash \psi$ to denote the fact that the assertion ψ holds for the trajectory tr in M.

In the definition below it is assumed that M is a transition system, $d = (q, s)$ is a state of computation in M, and $tr = (d_0, \alpha)$ is a trajectory in M such that $\alpha = (c_1, d_1), (c_2, d_2), \ldots, (c_i, d_i), \ldots$. We define the satisfiability relation $\vDash$ by induction on the height of formulae:

- $M, d \vDash P \Leftrightarrow s \in P$;
- $M, d \vDash \neg\varphi \Leftrightarrow$ it is not true that $M, d \vDash \varphi$;
- $M, d \vDash \varphi_1 \wedge \varphi_2 \Leftrightarrow M, d \vDash \varphi_1$ and $M, d \vDash \varphi_2$;
- $M, d \vDash \boldsymbol{E}\varphi \Leftrightarrow$ there exists a trajectory $tr' = (d, \alpha')$ in M such that $M, tr' \vDash \varphi$;

- $M, d \vDash \boldsymbol{A}\varphi \iff$ for any trajectory $tr' = (d, \alpha')$ in M it is true that $M, tr' \vDash \varphi$;
- if φ is a state formula then $M, tr \vDash \varphi \iff M, d_0 \vDash \varphi$;
- $M, tr \vDash \neg \psi \iff$ it is not true that $M, tr \vDash \psi$;
- $M, tr \vDash \psi_1 \wedge \psi_2 \iff M, tr \vDash \psi_1$ and $M, tr \backslash models\ \psi_2$;
- $M, tr \vDash \boldsymbol{X}_c\varphi \iff c = c_1$ and $M, d_1 \vDash \varphi$;
- $M, tr \vDash \boldsymbol{Y}_c\varphi \iff$ either $c \neq c_1$, or $M, d_1 \vDash \varphi$;
- $M, tr \vDash \boldsymbol{F}_L\varphi \iff \exists\, i \geq 0 : c_1c_2 \ldots c_i \in L$ and $M, tr|^i \vDash \varphi$;
- $M, tr \vDash \boldsymbol{G}_L\varphi \iff \forall\, i \geq 0 :$ if $c_1c_2 \ldots c_i \in L$ then $M, tr|^i \vDash \varphi$;
- $M, tr \vDash \varphi\, \boldsymbol{U}_L\psi \iff \exists\, i \geq 0 : c_1c_2 \ldots c_i \in L, M, tr|^i \vDash \psi$ and $\forall\, j, 0 \leq j < i,$ if $c_1c_2 \ldots c_i \in L$ then $M, tr|^j \vDash \varphi$.

Observe, that operators $\boldsymbol{X}_c$ and $\boldsymbol{Y}_c$, as well as $\boldsymbol{F}_L$ and $\boldsymbol{G}_L$, are dual to each other:

Proposition 1. *For any $\mathcal{LP}$-$\mathcal{CTL}^*$ formula φ, any $c \in \mathcal{C}$ and any $L \in \mathcal{L}$, and for an arbitrary trajectory tr in M*

- $tr \vDash \boldsymbol{X}_c\varphi \iff tr \vDash \neg \boldsymbol{Y}_c \neg \varphi$,
- $tr \vDash \boldsymbol{F}_L\varphi \iff tr \vDash \neg \boldsymbol{G}_L \neg \varphi$.

As usual, other Boolean connectives like $\vee$, $\rightarrow$, $\equiv$ may be defined by means of $\neg$ and $\wedge$. Some other $\boldsymbol{CTL}^*$ operators like, for example, **R** (release) or $\boldsymbol{W}$ (weak until) may be parametrized by environmental behaviour patterns in the same fashion.

The model checking problem we deal with is that of checking, given a finite state transducer Π operating over a semigroup $(S, e, \circ)$, and an $\mathcal{LP}$-$\mathcal{CTL}^*$ formula φ, whether $TS(\Pi, S), d_{init} \vDash \varphi$ holds. When a semigroup is fixed then we use a brief notation $\Pi \vDash \varphi$.

4. Model checking against $\mathcal{LP}$-$\mathcal{CTL}^*$ specifications

In this paper, we discuss only the most simple case of model checking problem for finite state transducers against $\mathcal{LP}$-$\mathcal{CTL}^*$ formulae when

- the semigroup $(S, \circ, e)$ the transducers operate over is a *free monoid*, which means that S is the set of all finite words in the alphabet $\mathcal{A}$, the binary operation $\circ$ is concatenation of words, and the neutral element e is the empty word ε;
- the family of environment behaviour patterns $\mathcal{L}$ is the family of regular languages in the alphabet $\mathcal{C}$;
- all basic predicates in $\mathcal{P}$ are specified by regular languages in the alphabet $\mathcal{A}$.

All regular languages used as environment behaviour patterns and basic predicate specifications are defined by means of deterministic finite state automata (DFAs). Therefore, the size of a $\mathcal{LP}$-$\mathcal{CTL}^*$ formula is the number of Boolean connectives and temporal operators occurred in φ plus the total size of automata used in φ to specify environment behaviour patterns and basic predicates.

Let us first describe a model checking algorithm for $\mathcal{LP}$-$\mathcal{CTL}$ fragment of $\mathcal{LP}$-$\mathcal{CTL}^*$, which consists of all $\mathcal{LP}$-$\mathcal{CTL}^*$ formulae such that every temporal operator $\mathbf{X}_c$, $\mathbf{Y}_c$, $\mathbf{F}_L$, $\mathbf{G}_L$ $\mathbf{U}_L$ is immediately preceded by a trajectory quantifier **E** or **A**. In our algorithm, we involve an explicit iterative model checking techniques for the ordinary $\mathcal{CTL}$ (see [8, 10]). Following this approach satisfiability checking of a formula φ in a state d of a model M is reduced to satisfiability checking of the largest subformulae of φ in the state d and in the neighboring states of M. In other words, a model checking procedure incrementally labels all states of a model by those subformulae of φ which are satisfied in these states.

Let $\Pi = (Q, \mathcal{C}, \mathcal{A}, q_{init}, T)$ be a finite state transducer over the free semigroup $(\mathcal{A}^*, \cdot, \varepsilon)$ and let φ be an $\mathcal{LP}$-$\mathcal{CTL}$ formula. There are five pairs of coupled $\mathcal{LP}$-$\mathcal{CTL}$ temporal operators: $\mathbf{AX}_c$ and $\mathbf{EX}_c$, $\mathbf{AY}_c$ and $\mathbf{EY}_c$, $\mathbf{AF}_L$ and $\mathbf{EF}_L$, $\mathbf{AG}_L$ and $\mathbf{EG}_L$, $\mathbf{AU}_L$ and $\mathbf{EU}_L$. As in the case of "ordinary" $\mathcal{CTL}$ (see), each of these couple can be expressed in terms of four main coupled operators $\mathbf{EX}_c$, $\mathbf{EY}_c$, $\mathbf{EG}_L$ and $\mathbf{EU}_L$:

Proposition 2. *For every formula φ the following equalities hold*

1. $\models \mathbf{AX}_c\varphi \equiv \neg\mathbf{EY}_c\neg\varphi$,
2. $\models \mathbf{AY}_c\varphi \equiv \neg\mathbf{EX}_c\neg\varphi$,
3. $\models \mathbf{AF}_L\varphi \equiv \neg\mathbf{EG}_L\neg\varphi$,
4. $\models \mathbf{EF}_L\varphi \equiv \mathbf{E}[true\ \mathbf{U}_L\varphi]$,
5. $\models \mathbf{AG}_L\varphi \equiv \neg\mathbf{EF}_L\neg\varphi$,
6. $\models \mathbf{A}[\varphi\ \mathbf{U}_L\psi] \equiv \neg\mathbf{E}[\neg\psi\ \mathbf{U}_L\ (\neg\varphi \wedge \neg\psi)] \wedge \neg\mathbf{EG}_L\neg\psi$.

Certainly, some other relationships like fixed-point identities are also valid in $\mathcal{LP}$-$\mathcal{CTL}^*$ (see [17]) but they will not be involved in this paper.

We can now bound our consideration with those $\mathcal{LP}$-$\mathcal{CTL}$ formulae which are constructed using only $\neg$, $\wedge$, $\mathbf{EX}_c$, $\mathbf{EY}_c$, $\mathbf{EG}_L$ and $\mathbf{EU}_L$. Let M be a transition system $TS(\Pi, \mathcal{A}^*) = (D, \mathcal{C}, d_{init}, \mathcal{T})$ of Π over $\mathcal{A}^*$. It should be noticed that M is, in general, infinite. Therefore, to obtain an effective model checking procedure we need a construction that will model the behaviour of M w.r.t. a target formula φ.

For every basic predicate $P \in \mathcal{P}$ let $A_P = (Q_P, \mathcal{A}, init_P, \delta_P, F_P)$ be a *minimal* DFA recognizing this language. Here Q_P is a finite set of states, $init_P$ is an initial state, F_P is a set of accepting states and $\delta_P : Q_P \times \mathcal{A} \to Q_P$ is a transition function. The latter can be extended to the set $\mathcal{A}^*$ in the usual fashion:

$$\delta_P(q_P, \varepsilon) = q_P \quad \textbf{and} \quad \delta_P(q_P, \gamma a) = \delta_P(\delta_P(q_P, \gamma), a).$$

Let $P_1, P_2, \ldots, P_k$ be all basic predicates occurred in the formula φ. Given a transducer $\Pi = (Q, \mathcal{C}, \mathcal{A}, q_{init}, T)$ and a formula φ, we build a *checking machine* — a transducer $\mathcal{M} = (\hat{Q}, \mathcal{C}, \mathcal{A}, \hat{q}_{init}, \hat{T})$, where

- $\hat{Q} = Q \times Q_{P_1} \times \ldots \times Q_{P_k}$ is a set of states (to avoid misunderstanding we will call them metastates);
- $\hat{q}_{init} = (q_{init}, init_{P_1}, \ldots, init_{P_k})$ is an initial metastate;
- $\hat{T} \subseteq \hat{Q} \times \mathcal{C} \times \hat{Q} \times \mathcal{A}^*$ is a transition relation, such that:

$$(\hat{\mathbf{q}}', \boldsymbol{c}, \hat{\boldsymbol{q}}'', \boldsymbol{h}) \in \hat{\boldsymbol{q}} \;\Leftrightarrow\; \begin{cases} (\boldsymbol{q}'_{\boldsymbol{\pi}}, \boldsymbol{c}, \boldsymbol{q}''_{\boldsymbol{\pi}}, \boldsymbol{h}) \in \boldsymbol{T} \textbf{ and} \\ \quad \boldsymbol{\delta}_{\boldsymbol{P_j}}(\boldsymbol{q}'_{\boldsymbol{P_j}}, \boldsymbol{h}) = \boldsymbol{q}''_{\boldsymbol{P_j}} \\ \qquad \textbf{for all } \boldsymbol{j}, \mathbf{1} \le \boldsymbol{j} \le \boldsymbol{k}. \end{cases}$$

Thus, every metastate is a tuple $\hat{q} = (q_0, q_1, \dots, q_k)$ such that $q_0 \in Q$ and $q_j \in Q_{P_j}$ for every j, $1 \le j \le k$, and the transition relation $\hat{T}$ synchronizes transitions of Π and the automata $A_{P_1}, \dots, A_{P_k}$ in response to every signal c. Recall that the elements of the free monoid are words s from $\mathcal{A}^*$. The checking machine $\mathcal{M}$ induces a binary relation $\sim$ on the set D: for an arbitrary pair $d' = (q', s')$ and $d'' = (q'', s'')$ of states of computation of Π over $\mathcal{A}^*$

$$\boldsymbol{d}' \sim \boldsymbol{d}'' \;\Leftrightarrow\; \begin{cases} \boldsymbol{q}' = \boldsymbol{q}'' \textbf{ and} \\ \boldsymbol{\delta}_{\boldsymbol{P_j}}(\boldsymbol{init}_{\boldsymbol{P_j}}, \boldsymbol{s}') = \boldsymbol{\delta}_{\boldsymbol{P_j}}(\boldsymbol{init}_{\boldsymbol{P_j}}, \boldsymbol{s}'') \textbf{ for all } \boldsymbol{j}. \end{cases}$$

The relation $\sim$ is clearly an equivalence relation of finite index, and every equivalence class of states of computation in $\boldsymbol{M}$ corresponds to a metastate of the checking machine $\boldsymbol{\mathcal{M}}$. As it can be seen from the definition of $\sim$, if two states of computation $\boldsymbol{d}'$ and $\boldsymbol{d}''$ are equivalent and there is a trajectory $\boldsymbol{tr}' = (\boldsymbol{d}', \boldsymbol{\alpha}')$ in $\boldsymbol{M}$, where $\boldsymbol{\alpha}' = (\boldsymbol{c_1}, \boldsymbol{d}'_{\boldsymbol{1}}), (\boldsymbol{c_2}, \boldsymbol{d}'_{\boldsymbol{2}}), \dots$, from one of these states, then there is also a corresponding trajectory $\boldsymbol{tr}'' = (\boldsymbol{d}'', \boldsymbol{\alpha}'')$, where $\boldsymbol{\alpha}'' = (\boldsymbol{c_1}, \boldsymbol{d}''_{\boldsymbol{1}}), (\boldsymbol{c_2}, \boldsymbol{d}''_{\boldsymbol{2}}), \dots$ from the other state, such that $\boldsymbol{d}'_{\boldsymbol{i}} \sim \boldsymbol{d}''_{\boldsymbol{i}}$ holds for every $\boldsymbol{i}, \boldsymbol{i} \ge \mathbf{1}$. Actually, this means that $\sim$ is a bisimulation relation on the state space of the transition system $\boldsymbol{M}$. It is well known (see [3, 8]) that bisimulation preserves the satisfiability of $\boldsymbol{CTL}$ formulae. The Proposition below shows that the same is true for $\boldsymbol{\mathcal{LP}}$-$\boldsymbol{CTL}$. This means that the checking machine provides a finite contraction of the infinite transition system $\boldsymbol{M} = \boldsymbol{TS}(\boldsymbol{\Pi}, \boldsymbol{\mathcal{A}}^*)$ w.r.t. satisfiability of $\boldsymbol{\mathcal{LP}}$-$\boldsymbol{CTL}$ formulae.

Proposition 3. *Suppose that* $\boldsymbol{d}'$ *and* $\boldsymbol{d}''$ *are two states of computation in* $\boldsymbol{M}$ *such that* $\boldsymbol{d}' \sim \boldsymbol{d}''$. *Then* $\boldsymbol{M}, \boldsymbol{d}' \vDash \boldsymbol{\varphi} \;\Leftrightarrow\; \boldsymbol{M}, \boldsymbol{d}'' \vDash \boldsymbol{\varphi}$.

Proof. It is carried out by induction on the nesting depth of $\boldsymbol{\varphi}$. When $\boldsymbol{\varphi}$ is a basic predicate the assertion follows from the definition of $\sim$. The cases when $\boldsymbol{\varphi} = \neg\boldsymbol{\psi}$ and $\boldsymbol{\varphi} = \boldsymbol{\psi_1} \wedge \boldsymbol{\psi_2}$ are obvious. We focus only on the case of $\boldsymbol{\varphi} = \mathbf{E}[\boldsymbol{\psi}\, \mathbf{U}_{\boldsymbol{L}} \boldsymbol{\chi}]$; the other cases when $\boldsymbol{\varphi}$ is of the form $\mathbf{EX}_{\boldsymbol{c}}\boldsymbol{\psi}$, $\mathbf{EY}_{\boldsymbol{c}}\boldsymbol{\psi}$, or $\mathbf{EG}_{\boldsymbol{L}}\boldsymbol{\psi}$ can be treated similarly.
Suppose that $\boldsymbol{M}, \boldsymbol{d}' \vDash \mathbf{E}[\boldsymbol{\psi}\, \mathbf{U}_{\boldsymbol{L}} \boldsymbol{\chi}]$. Then, by the definition of $\boldsymbol{\mathcal{LP}}$-$\boldsymbol{CTL}$ semantics, there exists a trajectory $\boldsymbol{tr}' = (\boldsymbol{d}', \boldsymbol{\alpha}')$, such that $\boldsymbol{M}, \boldsymbol{tr}' \vDash \boldsymbol{\psi}\, \mathbf{U}_{\boldsymbol{L}} \boldsymbol{\chi}$ and $\boldsymbol{\alpha}' = (\boldsymbol{c_1}, \boldsymbol{d}'_{\boldsymbol{1}}), (\boldsymbol{c_2}, \boldsymbol{d}'_{\boldsymbol{2}}), \dots$. As it was noticed above, there is also a corresponding trajectory $\boldsymbol{tr}'' = (\boldsymbol{d}'', \boldsymbol{\alpha}'')$ in $\boldsymbol{M}$, where $\boldsymbol{\alpha}'' = (\boldsymbol{c_1}, \boldsymbol{d}''_{\boldsymbol{1}}), (\boldsymbol{c_2}, \boldsymbol{d}''_{\boldsymbol{2}}), \dots$, such that $\boldsymbol{d}'_{\boldsymbol{i}} \sim \boldsymbol{d}''_{\boldsymbol{i}}$ holds for every $\boldsymbol{i}, \boldsymbol{i} \ge \mathbf{1}$. Then, by induction hypotheses, $\boldsymbol{M}, \boldsymbol{d}'_{\boldsymbol{i}} \vDash \boldsymbol{\psi} \Leftrightarrow \boldsymbol{M}, \boldsymbol{d}''_{\boldsymbol{i}} \vDash \boldsymbol{\psi}$ and $\boldsymbol{M}, \boldsymbol{d}'_{\boldsymbol{i}} \vDash \boldsymbol{\chi} \Leftrightarrow \boldsymbol{M}, \boldsymbol{d}''_{\boldsymbol{i}} \vDash \boldsymbol{\chi}$ hold for every $\boldsymbol{i}, \boldsymbol{i} \ge \mathbf{1}$.
Since $\boldsymbol{M}, \boldsymbol{tr}' \vDash \boldsymbol{\psi}\, \mathbf{U}_{\boldsymbol{L}} \boldsymbol{\chi}$, there exists $\boldsymbol{i}$ such that

1. $c_1 c_2 \dots c_i \in L$ and $M, tr'|^i \vDash \chi$;
2. for all $j < i$ if $c_1 c_2 \dots c_j \in L$ then $M, tr'|^j \vDash \psi$.

However, taking into account the fact that ψ and χ are state formulas, we must recognize that $M, tr''|^i \vDash \chi$ and that $M, tr''|^j \vDash \psi$ every time when $M, tr'|^j \vDash \psi$. Thus, we arrive at the conclusion that $M, tr'' \vDash \psi \mathbf{U}_L \chi$ and, hence, $M, d'' \vDash \mathrm{E}[\psi \mathbf{U}_L \chi]$. ■

Each metastate $\hat{\boldsymbol{q}} = (\boldsymbol{q_0}, \boldsymbol{q_1}, \dots, \boldsymbol{q_k})$ of the checking machine $\boldsymbol{\mathcal{A}}$ represents an equivalence class $\boldsymbol{D_{\hat{q}}}$ which includes all states $\boldsymbol{d} = (\boldsymbol{q}, \boldsymbol{h}) \in \boldsymbol{D}$ such that $\boldsymbol{q} = \boldsymbol{q_0}$ and $\boldsymbol{\delta_{P_j}}(\boldsymbol{init_{P_j}}, \boldsymbol{h}) = \boldsymbol{q_j}$ for all $\boldsymbol{j}, \boldsymbol{1} \leq \boldsymbol{j} \leq \boldsymbol{k}$. By using Proposition 3, we can correctly introduce a new satisfiability relation $\vDash_{\boldsymbol{0}}$ on the metastates of the checking machine:

$$\hat{\boldsymbol{q}} \underset{\boldsymbol{0}}{\vDash} \boldsymbol{\varphi} \quad \Leftrightarrow \quad \textbf{for some } \boldsymbol{d} \in \boldsymbol{D_{\hat{q}}}\colon \ \boldsymbol{M}, \boldsymbol{d} \vDash \boldsymbol{\varphi}\ .$$

Not only the states of the transition system $\boldsymbol{M} = \boldsymbol{TS}(\boldsymbol{\Pi}, \boldsymbol{S})$ correspond to the metastates of the checking machine $\boldsymbol{\mathcal{M}}$, but also there is a relationship between the trajectories in $\boldsymbol{M}$ and the traces in $\boldsymbol{\mathcal{M}}$ (they can be quite naturally called metatrajectories). More formally, every trajectory $\boldsymbol{tr} = (\boldsymbol{d_0}, \boldsymbol{\alpha})$ in $\boldsymbol{M}$ with $\boldsymbol{\alpha} = (\boldsymbol{c_1}, \boldsymbol{d_1})(\boldsymbol{c_2}, \boldsymbol{d_2}) \dots$, corresponds to a *metatrajectory* $\widehat{\boldsymbol{tr}} = (\hat{\boldsymbol{q}}_{\boldsymbol{0}}, \hat{\boldsymbol{\alpha}})$, where $\hat{\boldsymbol{\alpha}} = (\boldsymbol{c_1}, \hat{\boldsymbol{q}}_{\boldsymbol{1}})(\boldsymbol{c_2}, \hat{\boldsymbol{q}}_{\boldsymbol{2}}) \dots$ is such that for all $\boldsymbol{i} \geq \boldsymbol{0}$: $\boldsymbol{d_i} \in \boldsymbol{D}_{\hat{\boldsymbol{q}}_i}$. It is easy to see that every metatrajectory $\widehat{\boldsymbol{tr}} = (\hat{\boldsymbol{q}}_{\boldsymbol{0}}, \hat{\boldsymbol{\alpha}})$ corresponds to the only trajectory $\boldsymbol{tr} = (\boldsymbol{d_0}, \boldsymbol{\alpha})$, which originates in a given state $\boldsymbol{d_0}$ from $\boldsymbol{D}_{\hat{\boldsymbol{q}}_0}$.

The well-known labeling algorithm for conventional $\boldsymbol{CTL}$ and ordinary Kripke structures can be now adapted in such a way as to cope with model checking problem for $\boldsymbol{\mathcal{LP}\text{-}CTL}$. The algorithm operates as follows. For every metastate $\hat{\boldsymbol{q}} \in \hat{\boldsymbol{Q}}$ of the checking machine $\boldsymbol{\mathcal{M}}$ it computes a set $\boldsymbol{label}(\hat{\boldsymbol{q}})$ of all subformulae of $\boldsymbol{\varphi}$ satisfied in $\hat{\boldsymbol{q}}$. More formally, let $\boldsymbol{Sub}(\boldsymbol{\varphi})$ be the minimal set of $\boldsymbol{\mathcal{LP}\text{-}CTL}$ formulae such that:

1. $\varphi \in Sub(\varphi)$;
2. if $\neg\psi \in Sub(\varphi)$ then $\psi \in Sub(\varphi)$;
3. if $\psi \wedge \chi \in Sub(\varphi)$ then $\psi,\ \chi \in Sub(\varphi)$;
4. if $\mathbf{EX}_c\psi \in Sub(\varphi)$, $\mathbf{EY}_c\psi \in Sub(\varphi)$ or $\mathbf{EG}_L\psi \in Sub(\varphi)$ then $\psi \in Sub(\varphi)$;
5. if $\mathbf{E}[\psi \mathbf{U}_L \chi] \in Sub(\varphi)$ then $\psi,\ \chi \in Sub(\varphi)$.

The algorithm builds incrementally the sets $label(\hat{\boldsymbol{q}})$ of all those $\psi \in Sub(\varphi)$ for which $\hat{\boldsymbol{q}} \vDash_0 \psi$ holds. At the first step every $label(\hat{\boldsymbol{q}})$ contains only basic predicates, i. e. $label(\hat{\boldsymbol{q}}) \subseteq Sub(\varphi) \cap \mathcal{P}$. Then, at $step\ i$ the algorithm processes those subformulae ψ whose nesting depth is $i - 1$. Every time when the algorithm adds a subformula ψ to $label(\hat{q})$ it thus detects that $\hat{q} \vDash_0 \psi$.

All we need now is to describe how the algorithm should process formulae of 7 types: basic predicate $\boldsymbol{P}$, $\boldsymbol{\neg\psi}$, $\boldsymbol{\psi_1 \wedge \psi_2}$, $\boldsymbol{EX_c\psi}$, $\boldsymbol{\mathbf{EY}_c\psi}$, $\boldsymbol{EG_L\psi}$ and $\boldsymbol{\mathbf{E}[\psi \mathbf{U}_L \chi]}$.

- A basic predicate P_i is added to $label(\hat{q})$ iff $\hat{q} = (q_0, q_1, \dots, q_i, \dots, q_k)$ and $\hat{q}_i \in F_{P_i},\ i \geq 1$;
- A subformula $\neg\psi$ is added to $label(\hat{q})$ iff $\psi \notin label(\hat{q})$;
- A subformula $\psi_1 \wedge \psi_2$ is added to $label(\hat{q})$ iff both $\psi_1, \psi_2 \in label(\hat{q})$;

- A subformula $\boldsymbol{EX}_c\psi$ is added to $label(\hat{q})$ iff there exists a transition $\hat{q} \overset{c,h}{\rightarrow} \hat{q}'$ such that $\psi \in label(\hat{q}')$;
- A subformula $\boldsymbol{EY}_c\psi$ is added to $label(\hat{q})$ iff there exists a transition $\hat{q} \overset{c,h}{\rightarrow} \hat{q}'$ such that $\psi \in label(\hat{q}')$ or a transition $\hat{q} \overset{c,h}{\rightarrow} \hat{q}'$ such that $c' \neq c$;
- To handle a subformula $\boldsymbol{E}[\psi\, \boldsymbol{U}_L \chi]$ we construct a directed labeled graph (DLG) $\Gamma_U(\mathcal{M}, L)$ as follows. Let $A_L = (Q_L, \mathcal{C}, init_L, \delta_L, F_L)$ be a minimal DFA that recognizes the language L. Then the nodes of $\Gamma_U(\mathcal{M}, L)$ are all pairs $(\hat{q}, q_L) \in \hat{Q} \times Q_L$. This DLG has an arc of the form $(\hat{q}', q_L') \overset{c,h}{\rightarrow} (\hat{q}'', q_L'')$ iff $\hat{q}' \overset{c,h}{\rightarrow} \hat{q}"$ is a transition of $\mathcal{M}$ and $\delta_L(q_L', c) = q_L''$.
  We then delete all those nodes $(\hat{q}, q_L)$ of $\Gamma_U(\mathcal{M}, L)$ for which the relations $\psi \notin label(\hat{q})$, $\chi \notin label(\hat{q})$ and $q_L \in F_L$ hold simultaneously and discard all arcs incoming to or outcoming from such nodes. A DLG thus reduced is denoted by $\Gamma'_U(\mathcal{M}, L)$.
  A subformula $E[\psi\, U_L \chi]$ is added to the set $label(\hat{q})$ iff $\Gamma'_U(\mathcal{M}, L)$ includes the node $(\hat{q}, init_L)$ and there exists a directed path in this graph from this node to some node $(\hat{q}', q_L')$ such that $\chi \in label(\hat{q}')$ and $q_L' \in F_L$.
- For a subformula $EG_L\psi$ we construct a DLG $\Gamma_G(\mathcal{M}, L)$ in the same fashion and delete all the nodes $(\hat{q}, q_L)$ for which the relations $\psi \notin label(\hat{q})$ and $q_L \in F_L$ hold simultaneously. As the result we obtain the reduced DLG $\Gamma'\_G\ (\mathcal{M}, L)$.
  The subformula $\mathbf{EG}_L\psi$ is added to the set $label(\hat{\boldsymbol{q}})$ iff $\Gamma'_G(\mathcal{M}, L)$ includes the node $(\hat{\boldsymbol{q}}, init_L)$ and there exists a directed path in this graph from this node to some nontrivial *strongly connected component* (SCC), that is, to a subgraph, every node of which is reachable from any other node by some non-empty path.

As soon as all the subformulae from $Sub(\varphi)$ (including the formula φ) are processed we obtain the result of the model checking as

$$\boldsymbol{\Pi} \vDash \boldsymbol{\varphi} \quad \Leftrightarrow \quad \boldsymbol{\varphi} \in \boldsymbol{label}(\hat{\boldsymbol{q}}_{\boldsymbol{init}}).$$

The correctness of this assertion is based on the following relationship: $\hat{\boldsymbol{q}} \vDash_{\mathbf{0}} \boldsymbol{\varphi} \Leftrightarrow \boldsymbol{\varphi} \in \boldsymbol{label}(\hat{\boldsymbol{q}})$. It can be proved by applying induction on the nesting depth of formulae with the help of Proposition 3. We also need Propositions 4 and 5 to justify the induction step for formulae of the form $\mathbf{E}[\boldsymbol{\psi}\, \mathbf{U}_L \boldsymbol{\chi}]$ and $\mathbf{EG}_L\boldsymbol{\psi}$.
Suppose, that for every metastate $\hat{\boldsymbol{q}} \in \hat{\boldsymbol{Q}}$ it is true that $\hat{\boldsymbol{q}} \vDash_{\mathbf{0}} \boldsymbol{\psi} \Leftrightarrow \boldsymbol{\psi} \in \boldsymbol{label}(\hat{\boldsymbol{q}})$ and $\hat{\boldsymbol{q}} \vDash_{\mathbf{0}} \boldsymbol{\chi} \Leftrightarrow \boldsymbol{\chi} \in \boldsymbol{label}(\hat{\boldsymbol{q}})$. This statement is used as an inductive hypothesis.

Proposition 4. *Let $\hat{\boldsymbol{q}}_{\mathbf{0}} \in \hat{\boldsymbol{Q}}$ be an arbitrary metastate in $\boldsymbol{\mathcal{M}}$. Then $\hat{\boldsymbol{q}}_{\mathbf{0}} \vDash_{\mathbf{0}} \boldsymbol{E}[\boldsymbol{\psi}\, \boldsymbol{U}_L \boldsymbol{\chi}]$ iff some node $(\hat{\boldsymbol{q}}', \boldsymbol{q}_L')$ in DLG $\boldsymbol{\Gamma}'_U(\boldsymbol{\mathcal{M}}, \boldsymbol{L})$, such that $\hat{\boldsymbol{q}}' \vDash_{\mathbf{0}} \boldsymbol{\chi}$ and $\boldsymbol{q}_L' \in \boldsymbol{F}_L$, is reachable from the node $(\hat{\boldsymbol{q}}_{\mathbf{0}}, \boldsymbol{init}_L)$ by a directed path.*

Proposition 5. *Let* $\widehat{\boldsymbol{q}}_{\boldsymbol{0}} \in \widehat{\boldsymbol{Q}}$ *be an arbitrary metastate in* $\boldsymbol{\mathcal{M}}$. *Then* $\widehat{\boldsymbol{q}} \vDash_{\boldsymbol{0}} \boldsymbol{EG_L\psi}$ *iff some nontrivial strongly connected component is reachable from the node* $(\widehat{\boldsymbol{q}}, \boldsymbol{init_L})$ *in DLG* $\boldsymbol{\Gamma'_G(\mathcal{M}, L)}$ *by a directed path.*

The proofs of these Propositions are straightforward adaptations of the correctness proof of the tabular model checking algorithm for $\boldsymbol{CTL}$ which is discussed in much details in [8]. However, for completeness of the exposition we give here a proof of Proposition 5. The proof of Proposition 4 follows the similar line of reasoning.

Proof of Proposition 5 (Sketch).
(⇒) Suppose, that $\widehat{\boldsymbol{q}}_{\boldsymbol{0}} \vDash_{\boldsymbol{0}} \mathbf{EG}_{\boldsymbol{L}}\boldsymbol{\psi}$. Consider an arbitrary state $\boldsymbol{d_0} \in \boldsymbol{D}_{\widehat{\boldsymbol{q}}_{\boldsymbol{0}}}$. Then, by definition of $\vDash_{\boldsymbol{0}}$ and by Proposition 3, it is true that $\boldsymbol{M}, \boldsymbol{d_0} \vDash \mathbf{EG}_{\boldsymbol{L}}\boldsymbol{\psi}$. This means that there is a trajectory $\boldsymbol{tr} = (\boldsymbol{d_0}, \boldsymbol{\alpha})$, where $\boldsymbol{\alpha} = (\boldsymbol{c_1}, \boldsymbol{d_1}), (\boldsymbol{c_2}, \boldsymbol{d_2}), ...$, such that $\boldsymbol{M}, \boldsymbol{tr} \vDash \mathbf{G}_{\boldsymbol{L}}\boldsymbol{\psi}$. By the semantics of $\boldsymbol{\mathcal{LP}}$-$\boldsymbol{CTL}^*$, $\boldsymbol{M}, \boldsymbol{d_i} \vDash \boldsymbol{\psi}$ holds for every $\boldsymbol{i}$ such that $\boldsymbol{c_1 c_2} \ldots \boldsymbol{c_i} \in \boldsymbol{L}$.
Consider now the corresponding metatrajectory $\widehat{\boldsymbol{tr}} = (\widehat{\boldsymbol{q}}_{\boldsymbol{0}}, \widehat{\boldsymbol{\alpha}})$ in the checking machine, where $\widehat{\boldsymbol{\alpha}} = (\boldsymbol{c_1}, \widehat{\boldsymbol{q}}_{\boldsymbol{1}}), (\boldsymbol{c_2}, \widehat{\boldsymbol{q}}_{\boldsymbol{2}}), ...$, and let

$$\boldsymbol{\pi} = (\widehat{\boldsymbol{q}}_{\boldsymbol{0}}, \boldsymbol{init_L}) \xrightarrow{\boldsymbol{c_1,h_1}} (\widehat{\boldsymbol{q}}_{\boldsymbol{1}}, \boldsymbol{q_{1L}}) \xrightarrow{\boldsymbol{c_2,h_2}} (\widehat{\boldsymbol{q}}_{\boldsymbol{2}}, \boldsymbol{q_{2L}}) \xrightarrow{\boldsymbol{c_3,h_3}} \ldots,$$

be the respective path in the DLG $\boldsymbol{\Gamma_G(\mathcal{M}, L)}$ which originates in the node $(\widehat{\boldsymbol{q}}_{\boldsymbol{0}}, \boldsymbol{init_L})$. Relying on Proposition 3 and taking into account the fact that $\boldsymbol{q_{iL}} = \boldsymbol{\delta_L(init_L, c_1 c_2 \ldots c_i)}$ for every $\boldsymbol{i}, \boldsymbol{i} \geq \mathbf{0}$, we may conclude that $\widehat{\boldsymbol{q}}_{\boldsymbol{i}} \vDash_{\boldsymbol{0}} \boldsymbol{\psi}$ holds for every $\boldsymbol{i}$ such that $\boldsymbol{q_{iL}} \in \boldsymbol{F}$. By induction hypothesis, $\widehat{\boldsymbol{q}}_{\boldsymbol{i}} \vDash_{\boldsymbol{0}} \boldsymbol{\psi}$ is equivalent to $\boldsymbol{\psi} \in \boldsymbol{label}(\widehat{\boldsymbol{q}}_{\boldsymbol{i}})$. Therefore, by definition of DLG $\boldsymbol{\Gamma_G(\mathcal{M}, L)}$ the path $\boldsymbol{\pi}$ is the infinite path which is entirely contained in the $\boldsymbol{\Gamma'_G(\mathcal{M}, L)}$. Due to the finiteness of $\boldsymbol{\Gamma'_G(\mathcal{M}, L)}$, this path may be represented as a concatenation $\boldsymbol{\pi} = \boldsymbol{\pi_1 \pi_2}$, where $\boldsymbol{\pi_1}$ is a finite path, and $\boldsymbol{\pi_2}$ is an infinite path passing through each of its nodes infinitely often. It is clear that the set $\boldsymbol{V(\pi_2)}$ of all nodes of $\boldsymbol{\pi_2}$ is included in some strongly connected component. Thus, a nontrivial strongly connected component is reachable from the node $(\widehat{\boldsymbol{q}}_{\boldsymbol{0}}, \boldsymbol{init_L})$ in DLG $\boldsymbol{\Gamma'_G(\mathcal{M}, L)}$.

(⇐) Suppose, that a nontrivial strongly connected component is reachable from the node $(\widehat{\boldsymbol{q}}_{\boldsymbol{0}}, \boldsymbol{init_L})$ in DLG $\boldsymbol{\Gamma'_G(\mathcal{M}, L)}$. Then there exists an infinite path

$$\boldsymbol{\pi} = (\widehat{\boldsymbol{q}}_{\boldsymbol{0}}, \boldsymbol{init_L}) \xrightarrow{\boldsymbol{c_1,h_1}} (\widehat{\boldsymbol{q}}_{\boldsymbol{1}}, \boldsymbol{q_{1L}}) \xrightarrow{\boldsymbol{c_2,h_2}} (\widehat{\boldsymbol{q}}_{\boldsymbol{2}}, \boldsymbol{q_{2L}}) \xrightarrow{\boldsymbol{c_3,h_3}} \ldots,$$

in $\boldsymbol{\Gamma'_G(\mathcal{M}, L)}$ from the node $(\widehat{\boldsymbol{q}}_{\boldsymbol{0}}, \boldsymbol{init_L})$ Consider now the sequence of the first components $\widehat{\boldsymbol{q}}_{\boldsymbol{i}}$ of all nodes $(\widehat{\boldsymbol{q}}_{\boldsymbol{i}}, \boldsymbol{q_{iL}}), \boldsymbol{i} \geq \mathbf{0}$, occurred in this path. By the definition of the DLG $\boldsymbol{\Gamma'_G(\mathcal{M}, L)}$,

1. this sequence is a metatrajectory $\widehat{tr}$ in the checking machine $\mathcal{M}$,
2. $\psi \in label(\hat{q}_i)$ holds for every node $(\hat{q}_i, q_{iL})$ such that $q_{iL} \in F_L$.

By the induction hypothesis, the latter implies $\hat{q}_i \vDash_0 \psi$ for every metastate $\widehat{\boldsymbol{q}}_{\boldsymbol{i}}$ in this trajectory such that $c_1 c_2 \ldots c_i \in L$. Consider an arbitrary state $d_0 \in D_{\hat{q}_0}$ and a

trajectory $tr = (d_0, \alpha)$ in M, where $\alpha = (c_1, d_1), (c_2, d_2), \ldots$, which corresponds to $\widehat{tr}$. By definition of $\models_0$ and Proposition 3, $M, d_i \models \psi$ holds for every i such that $c_1 c_2 \ldots c_i \in L$. Then, according to the semantics of $\mathcal{LP}$-$\mathcal{CTL}^*$, $M, tr \models \mathbf{G}_L \psi$, and, hence, $M, d_0 \models \mathbf{EG}_L \psi$. Thus, by referring once again to definition of $\models_0$, we arrive at the conclusion that $\widehat{\boldsymbol{q}}_0 \models_0 \mathbf{EG}_L \psi$. ■

Now we estimate the complexity of the model checking algorithm for $\mathcal{LP}$-$\mathcal{CTL}$ described above. By the *size of a transducer* $\boldsymbol{\Pi} = (\boldsymbol{Q}, \boldsymbol{\mathcal{C}}, \boldsymbol{\mathcal{A}}, \boldsymbol{q_{init}}, \boldsymbol{T})$ we will mean the sum $\| \boldsymbol{\Pi} \| = |\boldsymbol{Q}| + |\boldsymbol{T}|$. The *size of a formula* $\boldsymbol{\varphi}$ is defined as follows. Suppose that basic predicates $\{\boldsymbol{P_i}\}_{\boldsymbol{i=1}}^{\boldsymbol{k}}$ occurred in $\boldsymbol{\varphi}$ are recognized by minimal DFAs $\{\boldsymbol{A_{P_i}} = \boldsymbol{Q_{P_i}}, \boldsymbol{\mathcal{A}}, \boldsymbol{init_{P_i}}, \boldsymbol{\delta_{P_i}}, \boldsymbol{F_{P_i}})\}_{\boldsymbol{i=1}}^{\boldsymbol{k}}$. Suppose also that environment patterns $\{\boldsymbol{L_i}\}_{\boldsymbol{i=1}}^{\boldsymbol{s}}$ used in $\boldsymbol{\varphi}$ are recognized by minimal DFAs $\{\mathbf{A}_{\mathbf{L_i}} = (\mathbf{Q}_{\mathbf{L_i}}, \boldsymbol{\mathcal{A}}, \mathbf{init}_{\mathbf{L_i}}, \boldsymbol{\delta}_{\mathbf{L_i}}, \mathbf{F}_{\mathbf{L_i}})\}_{\boldsymbol{i=1}}^{\boldsymbol{s}}$. Then the size of $\boldsymbol{\varphi}$ is the sum $\| \boldsymbol{\varphi} \| = |\boldsymbol{Sub}(\boldsymbol{\varphi})| + \sum_{\boldsymbol{i=1}}^{\boldsymbol{k}} |\boldsymbol{Q_{P_i}}| + \sum_{\boldsymbol{i=1}}^{\boldsymbol{s}} |\boldsymbol{Q_{L_i}}|$.

As it can be seen from the description of our model checking algorithm, the size of auxiliary graphs $\boldsymbol{\Gamma'_U}(\boldsymbol{\mathcal{M}}, \boldsymbol{L})$ and $\boldsymbol{\Gamma'_G}(\boldsymbol{\mathcal{M}}, \boldsymbol{L})$ used in this algorithm does not exceed the value $\| \boldsymbol{\Pi} \| \cdot \left(\prod_{\boldsymbol{i=0}}^{\boldsymbol{k}} |\boldsymbol{Q_{P_i}}| \right) \cdot \mathbf{max}(|\boldsymbol{Q_{L_i}}| : \mathbf{1} \leq \boldsymbol{i} \leq \boldsymbol{s})$. These graphs are processed in no more than $|\boldsymbol{Sub}(\boldsymbol{\varphi})|$ steps. So, the total time complexity of our model checking algorithm does not exceed the value $\| \boldsymbol{\Pi} \| \cdot |\boldsymbol{Sub}(\boldsymbol{\varphi})| \left(\prod_{\boldsymbol{i=0}}^{\boldsymbol{k}} |\boldsymbol{Q_{P_i}}| \right) \cdot \mathbf{max}(|\boldsymbol{Q_{L_i}}| : \mathbf{1} \leq \boldsymbol{i} \leq \boldsymbol{s})$ which is $\boldsymbol{O}(\| \boldsymbol{\Pi} \| \cdot \mathbf{2}^{\| \boldsymbol{\varphi} \|})$.

Because of these considerations, we get the following

Theorem 1. *Model checking of a finite state transducer* $\boldsymbol{\Pi}$ *operating over a free monoid against a formula* $\boldsymbol{\varphi} \in \mathcal{LP}$-$\mathcal{CTL}$ *can be performed in time* $\boldsymbol{O}(\| \boldsymbol{\Pi} \| \cdot \mathbf{2}^{\| \boldsymbol{\varphi} \|})$.

When a more general case of model checking problem of FSTs against $\mathcal{LP}$-$\mathcal{CTL}^*$ formulae is concerned we can rely on the well-known combining approach which is based on the interleaving application of model checking algorithms for $\mathcal{CTL}$ and $\mathcal{LTL}$. The details can be found in [8]. The similar procedure for $\mathcal{LP}$-$\mathcal{CTL}^*$ can be obtained in the same fashion by means of $\mathcal{LP}$-$\mathcal{CTL}$ model checking algorithm described above and $\mathcal{LP}$-$\mathcal{LTL}$ model checking algorithm developed in . Since this approach does not take into account any specific features of $\mathcal{LP}$-$\mathcal{CTL}^*$ formulae, we will not give a complete description of it.

5. $\mathcal{LP}$-$\mathcal{LTL}^*$ and ordinary Kripke structures

In this section, we consider the model checking problem for two subfamilies of $\mathcal{LP}$-$\mathcal{CTL}^*$ whose semantics can be defined on ordinary Kripke structures.

Recall, that a *Kripke structure* over a finite set $\boldsymbol{AP}$ of atomic propositions is a quadruple $\boldsymbol{M} = (\boldsymbol{Q}, \boldsymbol{q_{init}}, \boldsymbol{R}, \boldsymbol{\rho})$, where $\boldsymbol{Q}$ is a finite set of states which includes an initial state $\boldsymbol{q_{init}}$, $\boldsymbol{R} \subseteq \boldsymbol{Q} \times \boldsymbol{Q}$ is a transition relation and $\boldsymbol{\rho}: \boldsymbol{Q} \to \boldsymbol{2^{AP}}$ is a *labeling function* which for each state $\boldsymbol{q}$ gives a matching set $\boldsymbol{\rho(q)} \subseteq \boldsymbol{AP}$ of all atomic propositions that are evaluated to $\boldsymbol{true}$ in this state. As usual, the *size* of $\boldsymbol{M}$ is the sum $\| \boldsymbol{M} \| = |\boldsymbol{Q}| + |\boldsymbol{R}|$. Below we present two modifications of $\boldsymbol{\mathcal{LP}}$-$\boldsymbol{CTL}^*$ that are well suited for model checking of Kripke structures.

Given a Kripke structure $\boldsymbol{M} = (\boldsymbol{Q}, \boldsymbol{q_{init}}, \boldsymbol{R}, \boldsymbol{L})$, consider a set of $\boldsymbol{\mathcal{LP}}$-$\boldsymbol{CTL}^*$ formulae where $\boldsymbol{\mathcal{L}}$is a family of regular languages over one-letter alphabet $\{\boldsymbol{c}\}$ and $\boldsymbol{\mathcal{P}} = \boldsymbol{AP}$ (we denote this formulae by $\boldsymbol{\mathcal{LP}}$-1-$\boldsymbol{CTL}^*$) and a transition system $\boldsymbol{M_c} = (\boldsymbol{Q}, \{\boldsymbol{c}\}, \boldsymbol{q_{init}}, \boldsymbol{R_c}, \boldsymbol{L})$ where $(\boldsymbol{q'}, \boldsymbol{c}, \boldsymbol{q''}) \in \boldsymbol{R_c}$ iff $(\boldsymbol{q'}, \boldsymbol{q''}) \in \boldsymbol{R}$. Then for $\boldsymbol{q} \in \boldsymbol{Q}$ the relation $\boldsymbol{q} \vDash \boldsymbol{P}$ holds iff $\boldsymbol{P} \in \boldsymbol{\rho(q)}$. The semantics of more complex formulae is defined exactly as in Section 3.

Some $\boldsymbol{\mathcal{LP}}$-1-$\boldsymbol{CTL}^*$ formulae have an ability to keep track of the number of steps of the run. For example, an $\boldsymbol{\mathcal{LP}}$-1-$\boldsymbol{LTL}$ formula $\mathbf{AG}_{\boldsymbol{L}}\, \boldsymbol{\varphi}$, where $\boldsymbol{L} = \{\boldsymbol{c^{2n}}\}$ is a regular language which contains all 1-letter words of even length, expresses the assertion that $\boldsymbol{\varphi}$ holds at every even step of a run. By using the techniques of Ehrenfeucht-Fraisse games for Temporal Logics developed and studied in [11] one can prove that this property cannot be specified by means of usual $\boldsymbol{LTL}$. This certifies that $\boldsymbol{\mathcal{LP}}$-1-$\boldsymbol{CTL}^*$ is more expressive than $\boldsymbol{CTL}^*$ and justifies its use as a new specification language for finite state transducers and Kripke structures.

Observe, that given a set $\boldsymbol{AP}$ of all atomic propositions used in formulae we can use the $\boldsymbol{M_c}$ directly as a checking machine $\boldsymbol{\mathcal{M}}$ for the algorithm described in Section 4. Suppose that formula $\boldsymbol{\varphi}$ refers to 1-letter regular languages $\boldsymbol{L_1}, \boldsymbol{L_2}, \dots, \boldsymbol{L_s}$ as the parameters of temporal operators, and every language $\boldsymbol{L_i}$, $\boldsymbol{1 \le i \le s}$, is recognized by a DFA with a set of states $\boldsymbol{Q_{L_i}}$. Then the size of the graphs used in this algorithm does not exceed the value $\| \boldsymbol{M} \| \cdot \mathbf{max}(|\boldsymbol{Q_{L_i}}| : \boldsymbol{1 \le i \le s})$ which is $\boldsymbol{O}(\| \boldsymbol{M} \| \cdot \| \boldsymbol{\varphi} \|)$, where $\| \boldsymbol{\varphi} \| = |\boldsymbol{Sub}(\boldsymbol{\varphi})| + \sum_{\boldsymbol{i=0}}^{\boldsymbol{s}} |\boldsymbol{Q_{L_i}}|$.

Another modification of the Kripke structure $\boldsymbol{M}$ allows one to encode more detailed information of the computation flow. Let $\boldsymbol{\Sigma} = \boldsymbol{2^{AP}}$. For each state $\boldsymbol{q}$ in $\boldsymbol{M}$ there exists a letter $\boldsymbol{\sigma_{\rho(q)}} \in \boldsymbol{\Sigma}$ corresponding to the label $\boldsymbol{\rho(q)}$ assigned to this state.

Let $\boldsymbol{M_{AP}} = (\boldsymbol{Q} \cup \{\boldsymbol{err}\}, \boldsymbol{q_{init}}, \boldsymbol{R_{AP}}, \boldsymbol{\rho_{AP}})$ be a transition system for $\boldsymbol{M}$, where for every $\boldsymbol{q} \in \boldsymbol{Q}$ the following equalities hold: $\boldsymbol{\rho_{AP}(q)} = \boldsymbol{\rho(q)}$, $\boldsymbol{\rho_{AP}(err)} = \{\boldsymbol{err}\}$ and $\boldsymbol{R_{AP}} \subseteq \boldsymbol{Q} \times \boldsymbol{2^{AP}} \times \boldsymbol{Q}$ is a minimal transition relation such that:

- for each transition (q', q'') of the Kripke structure M there exists a *fair transition* $(q', \sigma_{\rho(q'')}, q'')$ and *erroneous transitions* (q', σ, err) for each $\sigma \neq \sigma_{\rho(q'')}$;
- $(err, \sigma, err) \in R_{AP}$ holds for each $\sigma \in \Sigma$ and $(err, \sigma, q) \notin R_{AP}$ holds for each $q \neq err$.

Then consider a specification language $\boldsymbol{\mathcal{LP}}$-$\boldsymbol{n}$-$\boldsymbol{CTL}^*$ which is a set of all such formulae where $\mathcal{L}$ is a family of regular languages over Σ and $\mathcal{P}$ = AP. To model

check a transition system M_{AP} against these formulae one needs to process only the states in Q and only the fair transitions. To do so, we replace all state formulae of type $\mathbf{A}\varphi$ with $\boldsymbol{A}(\boldsymbol{G}\neg err \longrightarrow \varphi)$ and all state formulae of type $\boldsymbol{E}\varphi$ with $\boldsymbol{E}(\boldsymbol{G}\neg err \wedge \varphi)$. The transition system M_{AP} thus obtained may as well be used as a checking machine for the model checking algorithm described in Section 4.
Thereby, the following theorem holds.

Theorem 2.

1. *There exists an algorithm for model checking of a Kripke structure M against a formula $\varphi \in \boldsymbol{\mathcal{LP}\text{-}1\text{-}CTL}$ with time complexity $O(\| M \| \cdot \| \varphi \|^2)$.*
2. *There exists an algorithm for model checking of a Kripke structure M against a formula $\varphi \in \boldsymbol{\mathcal{LP}\text{-}n\text{-}CTL}$ with time complexity $O\left(\| M \| \cdot \| \varphi \|^2 \cdot 2^{|AP|}\right)$.*

As it can be seen from this theorem, the exponential complexity of model checking procedure described in Section 4 is due to the language-theoretic nature of basic predicates used in $\boldsymbol{\mathcal{LP}\text{-}CTL^*}$.

6. Related papers and conclusion

Actually, the idea of providing parameterization of temporal operators is not new. In [27] right-linear grammar patterns were offered to define new temporal operators. The same kind of temporal patterns but specified by means of finite state automata were introduced in [18, 24]. For these extensions it was proved that they have the same expressiveness as $\boldsymbol{S1S}$ and that satisfiability checking problem in these logics is PSPACE-complete. We did not pursue a goal of merely expanding the expressive possibilities of $\boldsymbol{CTL^*}$; our aim was to make $\boldsymbol{CTL^*}$ more adequate for describing the behaviour of reactive systems. Almost the same kind of parametrization is used in Dynamic $\boldsymbol{LTL}$. However, our extension of $\boldsymbol{CTL^*}$ differs from that which was developed in [14], since in our logic basic predicates are also parameterized.
The $\boldsymbol{\mathcal{LP}\text{-}CTL^*}$ formulae allows one to specify and verify the behaviour of finite state transducers that operate over semigroups as well as classical Kripke structures. Moreover, when Kripke structures are concerned $\boldsymbol{\mathcal{LP}\text{-}CTL^*}$ has more expressive power than conventional temporal logics. But the place of $\boldsymbol{\mathcal{LP}\text{-}CTL^*}$ in the expressive hierarchy of specification languages, such as $\boldsymbol{S1S}$, $\boldsymbol{PDL}$ or $\boldsymbol{\mu}$-calculus, has not yet been established and remains a matter for our further research.
The results of this paper combined with the results of [17] provide positive solution to model checking for transducers over free semigroups. Free semigroups is the most simple algebraic structure which can be used for interpretation of basic actions performed by transducers when they are regarded as formal models of sequential reactive systems. Next, we are going to find out whether model checking algorithms could be built for transducers operating over more specific semigroups. Some preliminary results showed that this is not an easy problem. In [12] we proved that it is undecidable for the case of Abelian groups and free commutative semigroups.

It is also interesting how much the complexity of model checking algorithms for $\mathcal{LP}$-CTL^* depends on languages that are used as parameters of temporal operators. We assume that model checking problem becomes undecidable when context-free languages are allowed for this purpose. The complexity issues of model checking for regular variant of $\mathcal{LP}$-CTL^* also need further research. We assume that even for regular $\mathcal{LP}$-CTL this problem is PSPACE-complete.

As for practical application of the results obtained, the most important issue is that of adapting the existing means of working with finite automata to widely known model checking tools (like SPIN, ν-SMV, etc.) in order to be able to effectively implement the proposed model checking algorithms for $\mathcal{LP}$-CTL^*.

Acknowledgments

The authors of the article thank the anonymous reviewers for their valuable comments and advice on improving the article. This work was supported by the Russian Foundation for Basic Research, Grant N 18-01-00854.

References

[1]. Alur R., Cerny P. Streaming transducers for algorithmic verification of single-pass list-processing programs. In Proceedings of 38-th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, 2011, pp. 599-610

[2]. Alur R., Moarref S., and Topcu U. Pattern-based refinement of assume-guarantee specifications in reactive synthesis. In Proceedings of 21-st International Conference on Tools and Algorithms for the Construction and Analysis of Systems, 2015, pp. 501-516.

[3]. Baier C., Katoen J. Principles of Model Checking, 2008, MIT Press.

[4]. Blattner M., Head T. The decidability of equivalence for deterministic finite transducers. Journal of Computer and System Sciences, 1979, vol. 1, pp. 45-49.

[5]. Blattner M., T. Head T. Single-valued a-transducers. Journal of Computer and System Sciences, vol. 15, 1977, pp. 310-327.

[6]. Culik K, Karhumaki J. The equivalence of finite-valued transducers (on HDTOL languages) is decidable. Theoretical Computer Science, 1986, vol. 47, pp. 71-84.

[7]. Bouajjani A., Jonsson B., Nilsson M., Touili T. Regular Model Checking. Proceedings of 12-th International Conference on Computer Aided Verification, 2000, p. 403-418.

[8]. Clarke (Jr.) E. M., Grumberg O., Peled D. A. Model Checking. MIT Press, 1999.

[9]. Diekert V., Rozenberg G. eds. The Book of Traces, 1995, World Scientific, Singapore.

[10]. Emerson E.A., Halpern J.Y. Decision procedures and expressiveness in the temporal logic of branching time. Journal of Computer and System Sciences, vol. 30, N 1, 1985, pp. 1–24.

[11]. Etessami K., WilkeT. An Until Hierarchy and Other Applications of and Ehrenfeucht-Fraisse Game for Temporal Logic. Information and Computation, vol. 160, 2000, pp. 88-108.

[12]. Gnatenko A.R., Zakharov V. A. On the complexity of verification of finite state machines over commutative semigroups. In Proceedings of the 18-th International Conference "Problems of Theoretical Cybernetics", 2017, pp. 68-70 (in Russian).

[13]. Griffiths T. The unsolvability of the equivalence problem for free nondeterministic generalized machines. Journal of the ACM, vol. 15, 1968, pp. 409-413.
[14]. HenriksenJ. G., Thiagarajan P.S. Dynamic linear time temporal logic. Annals of Pure and Applied Logic, vol. 96, 1999, pp.187-207.
[15]. Hu Q., D'Antoni L. Automatic Program Inversion using Symbolic Transducers. In Proceedings of the 38-th ACM SIGPLAN Conference on Programming Language Design and Implementation, 2017, pp. 376-389.
[16]. Ibarra O. The unsolvability of the equivalence problem for Efree NGSM's with unary input (output) alphabet and applications. SIAM Journal on Computing, vol. 4, 1978, pp. 524-532.
[17]. Kozlova D. G., Zakharov V. A. On the model checking of sequential reactive systems. Proceedings of the25-th International Workshop on Concurrency, Specification and Programming (CS&P 2016), CEUR Workshop Proceedings, vol. 1698, 2016, pp. 233-244.
[18]. Kupferman O., Piterman N., Vardi M.Y. Extended Temporal Logic Revisited. In Proceedings of 12-th International Conference on Concurrency Theory, 2001, pp. 519-535.
[19]. Schutzenberger M. P. Sur les relations rationnelles. In Proceedings of Conference on Automata Theory and Formal Languages, 1975, pp. 209-213.
[20]. Sakarovitch J., de Souza R. On the decomposition of k-valued rational relations. In Proceedings of 25-th International Symposium on Theoretical Aspects of Computer Science, 2008, pp. 621-632.
[21]. Sakarovitch J., de Souza R. On the decidability of bounded valuedness for transducers. In Proceedings of the 33-rd International Symposium on Mathematical Foundations of Computer Science, 2008, pp. 588-600.
[22]. De Souza R. On the decidability of the equivalence for k-valued transducers. In Proceedings of 12-th International Conference on Developments in Language Theory, 2008, pp. 252-263.
[23]. Thakkar J., Kanade A., Alur R. A transducer-based algorithmic verification of retransmission protocols over noisy channels. In Proceedings of IFIP Joint International Conference on Formal Techniques for Distributed Systems, 2013, pp. 209-224.
[24]. Vardi M.Y., Wolper P. Yet Another Process Logic. Logic of Programs, 1983, pp. 501-512.
[25]. Veanes M., Hooimeijer P., Livshits B. et al. Symbolic finite state transducers: algorithms and applications. In Proceedings of the 39-th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, ACM SIGPLAN Notices, vol. 147, 2012, pp. 137-150.
[26]. Weber A. Decomposing finite-valued transducers and deciding their equivalence. SIAM Journal on Computing. vol. 22, 1993, pp. 175-202.
[27]. Wolper P. Temporal Logic Can Be More Expressive. Information and Control, vol. 56, N 1/2, 1983, pp. 72-99.
[28]. Wolper P., Boigelot B. Verifying systems with infinite but regular state spaces. In Proceedings of the 10-th Int. Conf. on Computer Aided Verification (CAV-1998), 1998, pp. 88-97.
[29]. Zakharov V.A. Equivalence checking problem for finite state transducers over semigroups. In Proceedings of the~6-th International Conference on Algebraic Informatics (CAI-2015), 2015, pp. 208-221.

О верификации конечных автоматов-преобразователей над полугруппами

[1] *А.Р. Гнатенко<gnatenko.cmc@gmail.com>*
[2] *В.А. Захаров<zakh@cs.msu.su>*
[1] *Московский государственный университет им. М. В. Ломоносова, 119991, Российская Федерация, Москва, Ленинские горы, д. 1*
[2] *Национальный исследовательский университет Высшая школа экономики, 101000, Российская Федерация, Москва, ул. Мясницкая, д. 20*

Аннотация. Последовательные реагирующие системы – это программы, которые взаимодействуют с окружением, получая от него сигналы или запросы, и реагируют на эти запросы, проводя операции с данными. Подобные системы могут служить моделью для многих программ: драйверов, систем реального времени, сетевых протоколов и др. В статье исследуются задача верификации программ такого вида. В качестве формальных моделей для реагирующих систем мы используем конечные автоматы-преобразователи, работающие над полугруппами. Для описания поведения автоматов-преобразователей введён новый язык спецификаций LP-CTL*. В его основу положена темпоральная логика CTL*. Этот язык спецификаций имеет две характерные особенности: 1) каждый темпоральный оператор снабжён регулярным выражением над входным алфавитом автомата, и 2) каждое атомарное высказывание задается регулярным выражением над выходным алфавитом автомата-преобразователя. В данной работе представлен табличный алгоритм проверки выполнимости формул LP-CTL* на моделях конечных автоматов-преобразователей, работающих над свободными полугруппами. Доказана корректность предложенного алгоритма и получена оценка его сложности. Кроме того, рассмотрен специальный фрагмент языка LP-CTL*, содержащий в качестве параметров темпоральных операторов только регулярные выражения над однобуквенным алфавитом. Показано, что этот фрагмента применим для спецификаций обычных моделей Крипке, и при этом его выразительные возможности превосходят обычную логику CTL*.

Ключевые слова: реагирующая система, автомат-преобразователь, верификация, проверка на модели, темпоральная логика, конечный автомат, регулярный язык.

DOI: 10.15514/ISPRAS-2018-30(3)-21

Для цитирования: Гнатенко А.Р., Захаров В.А. О верификации конечных автоматов-преобразователей над полугруппами. Труды ИСП РАН, том 30, вып. 3, 2018 г., стр. 303-324 (на английском языке). DOI: 10.15514/ISPRAS-2018-30(3)-21

Список литературы

[1]. Alur R., Cerny P. Streaming transducers for algorithmic verification of single-pass list-processing programs. In Proceedings of 38-th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, 2011, pp. 599-610

[2]. Alur R., Moarref S., and Topcu U. Pattern-based refinement of assume-guarantee specifications in reactive synthesis. In Proceedings of 21-st International Conference on Tools and Algorithms for the Construction and Analysis of Systems, 2015, pp. 501-516.
[3]. Baier C., Katoen J. Principles of Model Checking, 2008, MIT Press.
[4]. Blattner M., Head T. The decidability of equivalence for deterministic finite transducers. Journal of Computer and System Sciences, 1979, vol. 1, pp. 45-49.
[5]. Blattner M., T. Head T. Single-valued a-transducers. Journal of Computer and System Sciences, vol. 15, 1977, pp. 310-327.
[6]. Culik K, Karhumaki J. The equivalence of finite-valued transducers (on HDTOL languages) is decidable. Theoretical Computer Science, 1986, vol. 47, pp. 71-84.
[7]. Bouajjani A., Jonsson B., Nilsson M., Touili T. Regular Model Checking. Proceedings of 12-th International Conference on Computer Aided Verification, 2000, p. 403-418.
[8]. Clarke (Jr.) E. M., Grumberg O., Peled D. A. Model Checking. MIT Press, 1999.
[9]. Diekert V., Rozenberg G. eds. The Book of Traces, 1995, World Scientific, Singapore.
[10]. Emerson E.A., Halpern J.Y. Decision procedures and expressiveness in the temporal logic of branching time. Journal of Computer and System Sciences, vol. 30, N 1, 1985, pp. 1–24.
[11]. Etessami K., WilkeT. An Until Hierarchy and Other Applications of and Ehrenfeucht-Fraisse Game for Temporal Logic. Information and Computation, vol. 160, 2000, pp. 88-108.
[12]. Гнатенко А.Р., Захаров В.А. О сложности верификации конечных автоматов-преобразователей над коммутативными полугруппами. Материалы XVIII международной конференции «Проблемы теоретической кибернетики»' (Пенза, 20-24 июня, 2017), стр. 68-70.
[13]. Griffiths T. The unsolvability of the equivalence problem for free nondeterministic generalized machines. Journal of the ACM, vol. 15, 1968, pp. 409-413.
[14]. HenriksenJ. G., Thiagarajan P.S. Dynamic linear time temporal logic. Annals of Pure and Applied Logic, vol. 96, 1999, pp.187-207.
[15]. Hu Q., D'Antoni L. Automatic Program Inversion using Symbolic Transducers. In Proceedings of the 38-th ACM SIGPLAN Conference on Programming Language Design and Implementation, 2017, pp. 376-389.
[16]. Ibarra O. The unsolvability of the equivalence problem for Efree NGSM's with unary input (output) alphabet and applications. SIAM Journal on Computing, vol. 4, 1978, pp. 524-532.
[17]. Kozlova D. G., Zakharov V. A. On the model checking of sequential reactive systems. Proceedings of the25-th International Workshop on Concurrency, Specification and Programming (CS&P 2016), CEUR Workshop Proceedings, vol. 1698, 2016, pp. 233-244.
[18]. Kupferman O., Piterman N., Vardi M.Y. Extended Temporal Logic Revisited. In Proceedings of 12-th International Conference on Concurrency Theory, 2001, pp. 519-535.
[19]. Schutzenberger M. P. Sur les relations rationnelles. In Proceedings of Conference on Automata Theory and Formal Languages, 1975, pp. 209-213.
[20]. Sakarovitch J., de Souza R. On the decomposition of k-valued rational relations. In Proceedings of 25-th International Symposium on Theoretical Aspects of Computer Science, 2008, pp. 621-632.

[21]. Sakarovitch J., de Souza R. On the decidability of bounded valuedness for transducers. In Proceedings of the 33-rd International Symposium on Mathematical Foundations of Computer Science, 2008, pp. 588-600.

[22]. De Souza R. On the decidability of the equivalence for k-valued transducers. In Proceedings of 12-th International Conference on Developments in Language Theory, 2008, pp. 252-263.

[23]. Thakkar J., Kanade A., Alur R. A transducer-based algorithmic verification of retransmission protocols over noisy channels. In Proceedings of IFIP Joint International Conference on Formal Techniques for Distributed Systems, 2013, pp. 209-224.

[24]. Vardi M.Y., Wolper P. Yet Another Process Logic. Logic of Programs, 1983, pp. 501-512.

[25]. Veanes M., Hooimeijer P., Livshits B. et al. Symbolic finite state transducers: algorithms and applications. In Proceedings of the 39-th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, ACM SIGPLAN Notices, vol. 147, 2012, pp. 137-150.

[26]. Weber A. Decomposing finite-valued transducers and deciding their equivalence. SIAM Journal on Computing. vol. 22, 1993, pp. 175-202.

[27]. Wolper P. Temporal Logic Can Be More Expressive. Information and Control, vol. 56, N 1/2, 1983, pp. 72-99.

[28]. Wolper P., Boigelot B. Verifying systems with infinite but regular state spaces. In Proceedings of the 10-th Int. Conf. on Computer Aided Verification (CAV-1998), 1998, pp. 88-97.

[29]. Zakharov V.A. Equivalence checking problem for finite state transducers over semigroups. In Proceedings of the~6-th International Conference on Algebraic Informatics (CAI-2015), 2015, pp. 208-221.