CEUR-WS.org/Vol-2362/paper23.pdf

Machine Learning Technique for Regular Pattern Detector Synthesis: toward Mathematical Rationale

Grygoriy Zholtkevych[0000−0002−7515−2143] and Nataliya Polyakovska[0000−0003−2855−7970]

V. N. Karazin Kharkiv National University, Kharkiv, 61022, Ukraine
g.zholtkevych@karazin.ua, natalipolyakovska@gmail.com

Abstract. In the paper, the machine learning method for synthesis regular pattern detectors proposed early by authors is studied. This method can be used in event processing systems. This paper, in contrast to the previous one, is focused on studying the mathematical properties of concepts related to the method. The main result of the paper is proofs of theorems grounding that the method leads to the required result sometimes. The estimation of a probability for success is not considered in the paper.

Keywords: event· event stream processing· event pattern· pattern detector· regular pattern detector· regular pattern acceptor

1 Introduction

The widespread development of cloud computing and Internet of Things (IoT) leads to an increase in the share of cyber-physical systems of all the variety of software systems. The distribution, heterogeneity, and the variability of the composite of components and relationships between them are distinctive features for systems of this class. Thus, the architecture of these systems should provide mechanisms to plug and unplug components, reconfiguration of relationships, and synchronisation of component behaviours.

It is known that event-driven architecture is a good choice is known that event-driven architecture is a good choice for meeting the requirements mentioned above. In this context, the problem of controlling event streams is very important and may be considered even as the main one.

Thus, a stream of events is a very important concept for cyber-physical systems and distributed software systems in whole. The validity of this statement is caused by the practice of developing controlled asynchronously distributed systems. This practice led to the reference architecture for event-driven systems (see [7,2], for example). It is no coincidence that these developments are conducted in the context of cloud technologies because message passing is the only way to organise collaboration between components of cloud platforms. One can find a detailed analysis of the event processing technology in [3].

Creating an event processing subsystem for each component of a distributed software system lies in the focus of the system developers. Therefore, efficient tools for specifying the valid system component behaviour and recognising violations of the validity are needed.

We need to underline that there is a contradiction between the complexity to specify and check the behaviour requirements of the system components and the necessity to do fast these processes. An attempt to overcome this contradiction is discussed in the paper.

2 State of Art

The idea to use inductive methods for overcoming the mentioned above contradiction was firstly formulated in [10]. This idea has arisen as a result of developing the mathematical theory of some generalisation of automata called pre-automata and their applications in software engineering [1,12,9].

In [11], the machine learning method for the synthesis of special class event detectors was proposed and experimentally studied.

2.1 Regular Pattern Detectors

Remind that we consider the class of machines, which we call regular pattern detectors. A member of this class is defined as follows.

Definition 1. *A regular pattern detector is a tuple* $\mathcal{R} = \langle \Sigma, Q, q_0, \Pi, \delta \rangle$ *where*

- Σ *is a finite alphabet of input signals;*
- Q *is a finite sets of states;*
- $q_0 \in Q$ *is a fixed state called initial;*
- Π *is some finite alphabet whose elements mark the corresponding patterns;*
- $\delta \colon Q \times \Sigma \to \Pi + Q$ *is called the decision function* [1].

Below we consider non-empty words [2] over alphabet Σ as events (more precisely, messages about event occurrences) and denote the set of events by Σ^+.

To define the Π-indexed family $\mathcal{R}^+ = \{\mathcal{R}^+_s \mid s \in \Pi\}$ of event sets being detected by the pattern detector $\mathcal{R}$ we extend the decision function δ up to the partial mapping [3] $\delta^+ \colon Q \times \Sigma^+ \dashrightarrow \Pi + Q$ defined recursively as follows

$$\begin{array}{rl} \delta^+(q,a)\downarrow= \delta(q,a) & \text{for any } q \in Q \text{ and } a \in \Sigma; \\ \delta^+(q,ua)\uparrow & \text{if either } \delta^+(q,u)\downarrow= s \text{ where } s \in \Pi \text{ or } \delta^+(q,u)\uparrow; \\ \delta^+(q,ua)\downarrow= \delta(q',a) & \text{if } \delta^+(q,u)\downarrow= q' \text{ where } q' \in Q. \end{array}$$

[1] The sign "+" denotes here and below the disjoint union of sets.
[2] The necessary definitions and notation are given in Appx. B.
[3] The necessary definitions and notation see in Appx. A.

Definition 2. *We say that an event $u \in \Sigma^+$ is detected by a regular pattern detector $\mathcal{R}$ as a sample of the pattern $s \in \Pi$ (symbolically, $u \in \mathcal{R}^+_s$) if $\delta^+(q_0, u) \downarrow= s$.*

In other words, $\mathcal{R}^+_s = \{u \in \Sigma^+ \mid \delta^+(q_0, u) \downarrow= s\}$ for all $s \in \Pi$.

Thus, the Π-indexed family $\mathcal{R}^+ = \{\mathcal{R}^+_s \mid s \in \Pi\}$ of subsets of Σ^+ is associated with any regular pattern detector $\mathcal{R}$. Properties of this family are established by the next proposition.

Proposition 1. *For any regular pattern detector $\mathcal{R}$, the Π-indexed family $\mathcal{R}^+$ holds the following properties*

1. *$\mathcal{R}^+$ is a mutually disjoint family;*
2. *each member of $\mathcal{R}^+$ is a regular set*[4]*;*
3. *the set $\bigcup\limits_{s\in\Pi} \mathcal{R}^+_s$ is prefix-free*[5].

Proof. For proving the first item let us suppose that for some $u \in \Sigma^+$, the statements $u \in \mathcal{R}^+_{s_1}$ and $u \in \mathcal{R}^+_{s_2}$ are true where $s_1, s_2 \in \Pi$ and $s_1 \neq s_2$. By definition of the family $\mathcal{R}^+$, it means that $\delta^+(q_0, u) \downarrow= s_1$ and $\delta^+(q_0, u) \downarrow= s_2$ and, therefore (see Def. A1), $s_1 = s_2$. This contradiction proves the first item.
To prove the second item let us construct a finite acceptor $\mathcal{A}_s = \langle Q_s, q_0, F_s, \delta_s\rangle$ that recognises exactly words from $\mathcal{R}^+_s$. We define $Q_s = Q + \Pi + \{\bot\}$ where $\bot$ is used to refer to the special trash-state; the initial state of the detector q_0 is the initial state of the acceptor being constructed; the subset of acceptable states $F_s \subset Q_s$ is the singleton $\{s\}$; the transition function $\delta_s : Q_s \times \Sigma \to Q_s$ acts on a pair (q, a) as follows

$$\begin{aligned} \delta_s(q,a) &= \delta(q,a) && \text{for any } q \in Q \text{ and } a \in \Sigma; \\ \delta_s(s',a) &= \bot && \text{for any } s' \in \Pi \text{ and } a \in \Sigma; \\ \delta_s(\bot,a) &= \bot && \text{for any } a \in \Sigma. \end{aligned}$$

Now one can easily see that $\mathcal{R}^+_s$ coincides with the set of words being accepted by $\mathcal{A}_s$.
To prove the third item let us assume $u \in \bigcup\limits_{s\in\Pi} \mathcal{R}^+_s$ and $uv \in \bigcup\limits_{s\in\Pi} \mathcal{R}^+_s$ for some $v \in \Sigma^+$. The first statement ensures that $\delta^+(q_0, u) \downarrow= s$ for some $s \in \Pi$ and, therefore, for any $v \in \Sigma^+$, we have $\delta^+(q_0, uv)\uparrow$. This contradiction gives the required proof. $\square$

The inverse statement is also true. We will prove this fact later using the technique presented below.

2.2 Synthesis Method

The idea of the synthesis method consists of the follows

[4] See, Def. C3 of Appx. B

[5] See, Def. B5 of Appx. B

- we have some finite alphabet of signals Σ and some finite alphabet of recognisable pattern Π;
- we fix a Π-indexed disjoint family $\{E_s \mid x \in \Pi\}$ of finite subsets of Σ^+ with the prefix-free union and a finite subset $C \subset \Sigma^+$ of event messages being considered as undetected, moreover, $E \bigcap C = \varnothing$ where $E = \bigcup_{s \in \Pi} E_s$;
- we search a regular pattern detector $\mathcal{R} = \langle \Sigma, Q, q_0, \Pi, \delta \rangle$ such that
  1. $E_s \subset \mathcal{R}_s^+$ for all $s \in \Pi$;
  2. $\left(\bigcap_{s \in \Pi} \mathcal{R}_s^+ \right) \bigcap C = \varnothing$;
  3. the number of elements of Q is less or equal of the number of states for any detector satisfying 1 and 2.

This idea led the authors of [11] to Algorithm1.

Algorithm 1: Synthesis Method

```
Data: a $\Pi$-indexed family $E$ of finite sets of $\Sigma^+$ with the prefix-free union and
      a finite set $C$ of $\Sigma^+$ disjointed with the union of $E$
Result: a regular event detector $\mathcal{R}$
1  build the tree-like detector $\langle \Sigma, Q, q_0, \Pi, \delta \rangle$ corresponding to $E$ and denote it by
   $\mathcal{R}$;
2  foreach $q \in Q$ and $a \in \Sigma$ do
3      if $\delta(q, a) = \perp$ then
4          choose randomly $x$ from $(Q + \Pi) \setminus \{q, \perp\}$;
5          redefine $\delta(q, a)$ as $x$;
6          if $u \in \mathcal{R}_s^+$ for some $s \in \Pi$ and $u \in C$ then
7              rollback the redefinition;
8              continue
9          end
10         minimise the modified detector $\mathcal{R}$ using Hoproft's Method (see [4])
11     end
12 end
```

The like-tree detector $\langle \Sigma, Q, q_0, \Pi, \delta \rangle$ mentioned in item 1 of Algorithm 1 is defined as follows

$$\begin{aligned} Q &= \left\{ u \in \Sigma^* \mid uv \in \bigcup_{s \in \Pi} E_s \text{ for some } v \in \Sigma^* \right\} \bigcup \{\perp\}; \\ q_0 &= \epsilon; \\ \delta(\perp, a) &= \perp \quad \text{for all } a \in \Sigma; \\ \delta(u, a) &= ua \quad \text{if } ua \in Q; \\ \delta(u, a) &= \perp \quad \text{otherwise} \end{aligned}$$

followed by minimisation of Hopcroft's Method.

3 Parallel Joint of Regular Pattern Detectors

In this section, we describe a construction that allows simplifying the problem being studied and restricting the studying by the simple detectors called acceptors.

Let us assume that we have n ($n > 1$) regular pattern detectors denoted by $\mathcal{R}_i = \langle \Sigma, Q_i, q_{0,i}, \Pi_i, \delta_i \rangle$ with the same alphabet Σ of input signals for $i = 1, \ldots, n$.
In this case, we define the regular pattern detector $\mathcal{R} = \langle \Sigma, Q, q_0, \Pi, \delta \rangle$ as follows

$$\begin{array}{ll} Q & = \{\bot\} + Q_1 \times \ldots \times Q_n; \\ q_0 & = (q_{0,1}, \ldots, q_{0,n}); \\ \Pi & = \Pi_1 \cup \ldots \cup \Pi_n; \\ \delta(\bot, a) & = \bot \quad \text{for all } a \in \Sigma; \\ \delta((q_1, \ldots, q_n), a) & = (\delta_1(q_1, a), \ldots, \delta_n(q_n, a)) \quad \text{if } \delta_i(q_i, a) \in Q_i \\ & \qquad \text{for all } i = 1, \ldots, n; \end{array}$$

$$\begin{array}{ll} \delta((q_1, \ldots, q_n), a) = s & \text{if } \delta_i(q_i, a) \notin Q_i \text{ implies } \delta_i(q_i, a) = s \\ & \quad \text{for all } i = 1, \ldots, n; \\ \delta((q_1, \ldots, q_n), a) = \bot & \text{otherwise.} \end{array}$$

As above, the symbol $\bot$ is used to refer to a trash-state, which is not a member of any Q_i.

It is easy to understand that $\mathcal{R}$ is really a regular pattern detector. This detector is below called the parallel joint of the detectors $\mathcal{R}_1, \ldots, \mathcal{R}_n$ (symbolically, $\mathcal{R} = \overset{n}{\underset{i=1}{\bowtie}} \mathcal{R}_i$).

Below we use this definition to demonstrate that any Π-indexed family of subsets of Σ^+ satisfying the conditions of Prop. 1 can be represented as $\mathcal{R}^+$ for a regular pattern detector $\mathcal{R}$, which is a parallel joint of primitive, in some sense, regular pattern detectors called regular pattern acceptor.

Definition 3. *A regular pattern detector $\mathcal{A}$ is called a regular pattern acceptor if the correspondence set of patterns is a singleton.*

It is evident that Prop. 1 ensures the following properties of the set $\mathcal{A}^+$ formed by word accepted by $\mathcal{A}$: this set is regular and prefix-free. The following theorem shows that the converse fact is also true.

Theorem 1. *Any regular and prefix-free subset $\mathcal{F} \subset \Sigma^+$ is the set $\mathcal{A}^+$ for some regular pattern acceptor $\mathcal{A}$.*

Proof. Considering $\mathcal{F}$ is regular let us take the minimal finite-state acceptor $\mathcal{M} = \langle \Sigma, Q, q_0, F, \delta_{\mathcal{F}} \rangle$[6] that recognises words belonging to $\mathcal{F}$ and only such words. The minimality of $\mathcal{M}$ ensures the existence exactly one state $q_\bot \notin F$ such that

[6] See App. C

1. $\delta_{\mathcal{F}}(q_\perp, a) = q_\perp$ for all $a \in \Sigma$;
2. for any $q \in Q$ such that $q \neq q_\perp$ and $u \in \Sigma^+$, $\delta^*_{\mathcal{F}}(q_0, u) = q$ implies that $\delta^*_{\mathcal{F}}(q_0, uu') \in F$ for some $u' \in \Sigma^*$.

Further, if $q \in F$ then taking into account that $\mathcal{F}$ is prefix-free one can conclude that $\delta_{\mathcal{F}}(q, a) = q_\perp$ for all $a \in \Sigma$. Hence, the the minimality of $\mathcal{M}$ ensures that $F = \{q_{\mathrm{a}}\}$ for some $q_{\mathrm{a}} \in Q$.
Let us now define $Q' = Q \setminus \{q_{\mathrm{a}}\}$, $\Pi = \{*\}$, and

$$\delta(q,a) = \begin{cases} * & \text{if } \delta_{\mathcal{F}}(q,a) = q_{\mathrm{a}} \\ \delta_{\mathcal{F}}(q,a) & \text{otherwise} \end{cases}$$

Thus, $\mathcal{A} = \{\Sigma, \Pi, Q', \delta\}$ is a regular pattern acceptor and one can easily conclude that $\mathcal{A}^+$ equals $\mathcal{F}$. □

Now we are ready to prove the statement converse to Prop. 1.

Theorem 2. *Let Σ and Π be any finite alphabets, $\mathcal{F}$ be a Π-indexed family of subsets of Σ^+ and this family holds the conditions*

1. *$\mathcal{F}$ is a mutually disjoint family;*
2. *each member of $\mathcal{F}$ is a regular set;*
3. *the set $\bigcup\limits_{s \in \Pi} F_s$ is prefix-free*

then there exists a Π-indexed family $\mathcal{A} = \{\mathcal{A}_s \mid s \in \Pi\}$ of regular pattern acceptors such that $\mathcal{R}^+_s = F_s$ for all $s \in \Pi$ where $\mathcal{R} = \mathop{\bowtie}\limits_{s \in \Pi} \mathcal{A}_s$.

Proof. Since the union of all F_s is prefix-free, each F_s is prefix-free too. Moreover, each F_s is regular therefore Theorem 1 ensures for each $s \in \Pi$, the existence of a regular pattern acceptor $\mathcal{A}_s$ such that $\mathcal{A}^+_s = F_s$. The condition that $\mathcal{F}$ is a mutually disjoint family ensures immediately the equality $\mathcal{R}^+_s = F_s$ for all $s \in \Pi$ where $\mathcal{R} = \mathop{\bowtie}\limits_{s \in \Pi} \mathcal{A}_s$. □

Combining Prop. 1 and Theorem 2 one can easily obtain the following.

Corollary 1. *For any finite alphabets Σ and Π, a Π-indexed family $\mathcal{F}$ of subsets of Σ^+ is the family associated with some regular pattern detectors if and only if the following conditions hold*

1. *$\mathcal{F}$ is a mutually disjoint family;*
2. *each member of $\mathcal{F}$ is a regular set;*
3. *union of all member of $\mathcal{F}$ is a prefix-free subset of Σ^+.*

4 Specification of Regular Prefix-free Sets

The results obtained above allow further us to restrict our consideration by the class of regular pattern acceptors. Therefore, the theorem is proven in this

Algorithm 2: Synthesis Method for Acceptors

```
Data: a prefix-free finite set $E$ of $\Sigma^+$ and a finite set $C$ of $\Sigma^+$ disjointed with $E$
Result: a regular event acceptor $\mathcal{A}$
build the tree-like acceptor $\langle \Sigma, Q, q_0, \delta \rangle$ corresponding to $E$ and denote it by $\mathcal{A}$;
foreach $q \in Q$ and $a \in \Sigma$ do
    if $\delta(q, a) = \perp$ then
        choose randomly $q'$ from $(Q + \{*\}) \setminus \{q, \perp\}$;
        redefine $\delta(q, a)$ as $q'$;
        if $u \in \mathcal{A}^+$ for some $u \in C$ then
            rollback the redefinition;
            continue
        end
        minimise the modified detector $\mathcal{A}$ using Hoproft's Method (see [4])
    end
end
```

section is the Main Theorem of the paper. It gives a method to specify any regular prefix-free set by two finite word subsets.

We begin with a specialised synthesis method, represented by Algorithm 1 for event pattern acceptors (see Algorithm 2 below).

The like-tree acceptor $\mathcal{A}_0 = \langle \Sigma, Q, q_0, \delta \rangle$ mentioned in item 1 of Algorithm 2 is defined as follows

$$\begin{aligned} Q &= \{u \in \Sigma^* \mid uv \in E \text{ for some } v \in \Sigma^*\} \bigcup \{\perp\}; \\ q_0 &= \epsilon; \\ \delta(\perp, a) &= \perp \quad \text{for all } a \in \Sigma; \\ \delta(u, a) &= ua \quad \text{if } ua \in Q; \\ \delta(u, a) &= \perp \quad \text{otherwise} \end{aligned}$$

followed by minimisation of Hopcroft's Method.

Further, we need some auxiliary notion. The definition and proof of two important properties are presented here.

Definition 4. *For any alphabet Σ and a subset $L \subset \Sigma^+$, we say that a word $u \in L$ is L-prime if for any $u_1, u_3 \in \Sigma^*$ and $u_2 \in \Sigma^+$, the equation $u = u_1u_2u_3$ implies $u_1u_3 \notin L$.*

Proposition 2. *For any alphabet Σ and a subset $L \subset \Sigma^+$, the subset $\operatorname{Pm} L \subset L$ containing exactly L-prime words is prefix-free.*

Proof. Indeed, if word $u = u'u''$, where $u' \in \operatorname{Pm} L$ and $u'' \in \Sigma^+$, belongs to $\operatorname{Pm} L$ then the representation $u = u'u''\epsilon$ ensures that $u'\epsilon = u' \notin L$. The obtained contradiction proves the proposition. □

Proposition 3. *For any alphabet Σ and a regular subset $L \subset \Sigma^+$, the subset $\operatorname{Pm} L \subset L$ is finite.*

Proof. According to the pumping lemma for regular languages [6, Lemma 8, p. 119] (see also, [5, Sect. 4.6, p. 166]) there exists an integer $n \geqslant 1$ depending only on L and such that any word $u \in L$ of length at least n can be represented as $u_1u_2u_3$, where $u_1, u_3 \in \Sigma^*$, $u_2 \in \Sigma^+$, $\operatorname{lng} u_1u_2 \leq n$, and $u_1u_3 \in L$. Consequently, all words longer than n are not L-prime. Obviously, there are a finite set of words with length less than n, accordingly set $\operatorname{Pm} L$ is also finite. □

Now we are ready to formulate and obtain the main results of the paper.

Theorem 3. *Any regular pattern acceptor $\mathcal{A} = \langle \Sigma, Q, q_0, \delta \rangle$ can be reached by using Synthesis Method specified by Algorithm 2 with set E equal to* $\operatorname{Pm} \mathcal{A}^+$.

Proof. Since $\operatorname{Pm} \mathcal{A}^+$ is prefix-free and regular according to Propositions 2 and 3, it can be used as set E for Synthesis Method.
Taking into account that after each iteration of the loop 2–12 in Algorithm 2

1. the acceptor $\mathcal{A}$ has at most one inefficient state
2. the number of its states does not increases,

one can use breadth-first search method in the graph of the target acceptor for the sequential redefining transition function of the like-tree acceptor built in accordance with item 1 of Algorithm 2. This process leads evidently to the target acceptor. □

Note that in the proof of Theorem 3 we assigned $C = \varnothing$. Therefore, we do not have any guarantees that Algorithm 2 halts after reaching the target acceptor. However, we prove that there exists some finite subset $C \subset \Sigma^+$ disjointed to $\operatorname{Pm} \mathcal{A}$ that provide the correct termination of the algorithm. We use methods of general topology to choose C. All necessary definitions and facts can be found in any textbook (for example, in [8]).

Theorem 4. *For any regular event pattern acceptor $\mathcal{A}$, a finite set $C \subset \Sigma^+$ that allows to exactly restore acceptor $\mathcal{A}$ with the set* $\operatorname{Pm} \mathcal{A}^+$ *as E using Algorithm 2 exists.*

Proof. Let us consider the set Σ^ω formed by infinite sequences of elements of Σ. It is evident that the family $\{u \cdot \Sigma^\omega \subset \Sigma^\omega \mid u \in \Sigma^+\}$, where $u \cdot \Sigma^\omega$ is formed by sequences with prefix u, is a base of a topology. This topology is Tikhonov topology on Σ^ω considered as the countable power of Σ in the assumption that the last is equipped with the discrete topology. Tikhonov Theorem guarantees that the space under consideration is compact. It is this fact that we need in the proof.
Further, let $\perp$ denotes the unique inefficient state for acceptor $\mathcal{A}$. One can divide Σ^ω into two disjoint components $\Sigma^\omega = C_1 \bigcup C_2$ as follows

- a sequence $s \in \Sigma^\omega$ belongs to C_1 if all and only if for some its prefix $u \in \Sigma^+$ the equation $\delta^+(q_0, u) = \perp$ is fulfilled;
- a sequence $s \in \Sigma^\omega$ belongs to C_2 if all and only if for any its prefix $u \in \Sigma^+$ either $\delta^+(q_0, u)\uparrow$ or the equation $\delta^+(q_0, u) \downarrow = x$ implies $x \neq \perp$.

It is evident that $C_1 = \bigcup_{u \in \mathcal{A}_\perp} u \cdot \Sigma^\omega$ where $\mathcal{A}_\perp$ is formed by such u that

1. $\delta^+(q_0, u) \downarrow = \perp$ and
2. for any proper prefix u' of u the equation $\delta^+(q_0, u') \downarrow = \perp$ is not fulfilled.

Further, one can easily conclude that C_2 is open in Tikhonov topology. Indeed, if a sequence $s \in C_2$ then there exists some prefix u of s and $u' \in \Sigma^*$ such that $\delta^+(q_0, uu') \downarrow = *$. But in this case, any $s' \in uu' \cdot \Sigma^\omega$ belongs to C_2 and, therefore $uu' \cdot \Sigma^\omega \subset C_2$.
Openness of C_2 ensures closedness of C_1. Moreover, the family $\{u \cdot \Sigma^\omega \mid u \in \mathcal{A}_\perp\}$ is an open covering of C_1. Thus, compactness of Σ^ω ensures the finiteness of $\mathcal{A}_\perp$ and, therefore, we can assign $C = \mathcal{A}_\perp$ for Algorithm 2.
It is evident that this choice ensures reaching exactly $\mathcal{A}$ under using Algorithm 2.
$\square$

Theorem 1, Theorem 2, Theorem 3, and Theorem 4 ensure the following fact.

Main Theorem. *For any regular pattern acceptor $\mathcal{A}$, the finite prefix-free set* $\mathrm{Pm}\,\mathcal{A}^+$ *and the finite set $\mathcal{A}_\perp$ (see proof of the previous theorem) determine uniquely the language $\mathcal{A}^+$ by using Algorithm 2.*

5 Conclusion

The paper examines the method of machine learning proposed earlier by the authors, designed to synthesise regular pattern detectors based on training sets.

In the paper, the method for decomposing an arbitrary regular pattern detector into prime ones has been grounded. These prime regular pattern detectors have been called regular pattern acceptors and characterised as detectors with the single output. It has been established that any recognition problem for regular pattern detectors can be reduced to the recognition problems for the corresponding acceptors (Theorems 1 and 2).

The use of these facts allowed to prove the main result of the work, justifying to reach the required acceptor using the previously proposed Synthesis Method specified for the case of acceptors (Theorems 3 and 4).

Thus, we have proven that the Synthesis Method based on Machine Learning Technique proposed in [10,11] leads sometimes to the required result. The next step of studying is to estimate the probability of success of the synthesis process and understand whether a good probability for success gotten experimentally in [11] is grounded.

Appendix A Partial Mappings

This section contains the definitions and notation that are relating to the concept of a partial mapping and used above.

Definition A1. *A partial mapping f from a set X into a set Y (symbolically, $f\colon X \dashrightarrow Y$) is the triple $\langle X, Y, \varGamma_f\rangle$ where $\varGamma_f \subset X \times Y$ if this triple satisfies the condition*

$$\textit{for all } x \in X \textit{ and } y', y'' \in Y,\ \langle x, y'\rangle \in \varGamma_f \textit{ and } \langle x, y''\rangle \in \varGamma_f \textit{ imply } y' = y''.$$

Notation A1. Let X and Y be sets, and $f\colon X \dashrightarrow Y$ then for any $x \in X$,

$$\begin{array}{ll} f(x)\!\uparrow & \text{means that } \langle x, y\rangle \notin \varGamma_f \text{ for any } y \in Y; \\ f(x)\!\downarrow & \text{means that } \langle x, y\rangle \in \varGamma_f \text{ for some } y \in Y; \\ f(x)\downarrow= y & \text{means that } \langle x, y\rangle \in \varGamma_f \text{ where } y \in Y. \end{array}$$

We use the symbol ϵ to refer to any partial mapping of the form $\langle X, Y, \varnothing\rangle$.

Appendix B Alphabets and Words

This section contains the definitions, notation, and facts that are relating to the concept of a word and used above.

Definition B1. *An alphabet is a finite set whose elements called tokens or symbols.*

Definition B2. *A word over an alphabet $\varSigma$ is a partial mapping $u\colon \mathbb{N} \dashrightarrow \varSigma$ such that*

$$\begin{array}{l} \textit{there exists } n \in \mathbb{N} \textit{ such that } u(n)\!\uparrow; \\ \textit{for any } m, n \in \mathbb{N} \textit{ such that } m \le n,\ u(n)\!\downarrow \textit{ implies } u(m)\!\downarrow. \end{array}$$

Notation B1. The set of words over an alphabet $\varSigma$ is denoted by $\varSigma^*$, and the set $\varSigma^* \setminus \{\epsilon\}$ is denoted by $\varSigma^+$.
If $u \in \varSigma^+$ and $n \in \mathbb{N}$ such that $u(n)\!\downarrow$ then $u[n]$ is the token from $\varSigma$ satisfying the condition $u(n)\downarrow= u[n]$.

Definition B3. *The length of a word $u \in \varSigma^*$ is defined as follows*

$$\operatorname{lng} u = \min\{n \in \mathbb{N} \mid u(n)\!\uparrow\}.$$

Definition B4. *For any alphabet $\varSigma$ and $u, v \in \varSigma^*$, the word uv is defined as follows*

$$\begin{array}{lll} (uv)(n)\downarrow= u[n] & \textit{whenever} & 0 \le n < \operatorname{lng} u; \\ (uv)(n)\downarrow= v[n - \operatorname{lng} u] & \textit{whenever} & \operatorname{lng} u \le n < \operatorname{lng} u + \operatorname{lng} v; \\ (uv)(n)\!\uparrow & \textit{whenever} & n \ge \operatorname{lng} u + \operatorname{lng} v. \end{array}$$

This word is called concatenation of u and v.

Definition B5. *For any alphabet $\varSigma$, a subset $L \subset \varSigma^*$ is called prefix-free if $uv \in L$ ensures $v = \epsilon$ for any $u \in L$ and $v \in \varSigma^*$.*

Appendix C Finite-State Acceptors and Regular Sets of Words

Definition C1. *A finite-state acceptor is a pentacle* $\mathcal{M} = \langle \Sigma, Q, q_0, F, \delta \rangle$ *where*

Σ	*is a finite input alphabet;*
Q	*is a finite set of states;*
$q_0 \in Q$	*is some fixed state called initial;*
$F \subset Q$	*is some fixed subset of states that are called acceptable;*
$\delta : Q \times \Sigma \to Q$	*is a mapping called a transition function.*

For a finite-state acceptor $\mathcal{M} = \langle \Sigma, Q, q_0, F, \delta \rangle$, one can define the following extension $\delta^* : Q \times \Sigma^* \to Q$ of the mapping δ

$$\begin{array}{lll} \delta^*(q, \epsilon) & = q & \text{for any } q \in Q\,; \\ \delta^*(q, ua) & = \delta(\delta^*(q, u), a) & \text{for any } q \in Q\,,\ u \in \Sigma^*\,,\ \text{and } a \in \Sigma\,. \end{array}$$

Definition C2. *A finite state acceptor* $\mathcal{M} = \langle \Sigma, Q, q_0, F, \delta \rangle$ *recognises a word* $u \in \Sigma^*$ *if* $\delta^*(q_0, u) \in F$.

Definition C3. *For any alphabet* Σ*, a subset* $L \subset \Sigma^*$ *is called regular if there is a finite-state acceptor that recognises exactly words from* L*.*

More detailed information about regular set and finite-state acceptors one can find in, for example, [5].

References

1. Dokuchaev, M., Novikov, B., Zholtkevych, G.: Partial actions and automata. Algebra and Discrete Mathematics **11**(2), 51–63 (2011)
2. Event-driven reference architecture, https://www.ibm.com/cloud/garage/architectures/eventDrivenArchitecture/reference-architecture, (last accessed 1.03.2019)
3. Etzion, O., Niblett, P.: Event Processing in Action. Manning (2011)
4. Hopcroft, J.: Theory of Machines and Computations, chap. An $n \log n$ Algorithm for Minimizing States in a Finite Automaton, pp. 189–196. Academic Press, New York (1971)
5. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Addison Wesley (2003)
6. Rabin, M., Scott, D.: Finite Automata and Their Decision Problems. IBM J. Res. Dev. **3**(2), 114–125 (1959)
7. Reference Architecture: Event-Driven Microservices with Apache Kafka, https://devcenter.heroku.com/articles/event-driven-microservices-with-apache-kafka, (last accessed 1.03.2019)
8. Willard, S.: General Topology. Addison-Wesley Publishing Company (1970)
9. Zholtkevych, G.: Realisation of Synchronous and Asynchronous Black Boxes Using Machines. In: Ermolayev, V., et al. (eds.) Information and Communication Technologies in Education, Research, and Industrial Applications, CCIS, vol. 594, pp. 124–139. Springer International Publishing (2016)

10. Zholtkevych, G., Dorozhinsky, V., Khadikov, A.: Regular Event Processing and Machine Learning Correctness Verification. Information processing systems **34**(9), 162–166 (2016)
11. Zholtkevych, G., Lukyanenko, S., Polyakovska, N.: Toward Synthesis of Event-Pattern Detectors for Event Complex Processing with Using Machine Learning. Volume II: Workshops. In: Ermolaev, V., et al. (eds.) ICT in Education, Research and Industrial Applications. Integration, Harmonization and Knowledge Transfer. pp. 707–715. Kyiv, Ukraine (May 14–17 2018)
12. Zholtkevych, G., Novikov, B., Dorozhinsky, V.: Pre-automata and Complex Event Processing. In: Ermolayev, V., et al. (eds.) Information and Communication Technologies in Education, Research, and Industrial Applications, CCIS, vol. 469, pp. 100–116. Springer International Publishing (2014)