Toward Synthesis of Event-Pattern Detectors for Event Complex Processing with Using Machine Learning

Grygoriy Zholtkevych[1], Stanislav Lukyanenko[2], and Natalya Polyakovska[1]

[1] Math and Comp. Sci. School, V.N. Karazin Kharkiv National University
4, Svobody Sqr., Kharkiv, 61022, Ukraine
g.zholtkevych@karazin.ua; natalipolyakovska@gmail.com
[2] Department of Informatics, Technical University of Munich
3, Boltzmannstr., 85748, Garching, Germany
stlukyanenko@gmail.com

Abstract. The tendency to expand the use of event-driven architecture leads to the need to improve the efficiency of designing components of such systems, in particular, event-pattern detectors. Authors of the paper propose to use machine-learning to automate designing processes of event-pattern detectors. For substantiating their opinion, the authors conducted a series of computer experiments, showing that the idea is not groundless. Summing up, the authors draw attention to the issues that require further research aimed at creating information technology for the synthesis of event-pattern detectors.

Keywords: event-driven-architecture, self-delimiting Turing machine, event stream processing, event-pattern detector, machine learning algorithm, computer experiment

1 Introduction

Global network solutions, based on the Internet platform, which include besides software components various subsystems of information registration and executive physical subsystems form the class of complex systems developing successfully nowadays. This class of systems is called Internet-of-Things (IoT) [1]. It can be considered as the subclass of the more wide class of systems so-called cyber-physical systems (CPS) [7]. From an architectural point of view, systems of this class are distributed systems whose integrity is maintained through the interaction of system components by messaging. This architectural approach is known as Message-Driven Architecture, wich is more known as Event-Driven Architecture (EDA) [2].

Realisation of EDA for information systems is known as Event Stream Processing (ESP). ESP is a complex of technological tools designed for

assisting the development of event-driven information systems [5]. One of the most important units of ESP systems is the event-pattern detector. The main purpose of the detector associated with a system component is the accumulation of a sequence of messages about events that are relevant to this component, and at the moment when the accumulated information becomes sufficient to make a decision about the class of the current situation, it transmits the corresponding message to the executive system.

The mathematical model of such a system is proposed in [10], and the "black box" models associated with such systems is studied in [9].

In this paper, we propose to discuss the possibility of using machine learning methods for automating synthesis process of one simple but important class of event-pattern detectors. We use some ideas contained in [3] and [4].

2 Mathematical Model of Detectors

To build a mathematical model of an event-pattern detector, we need to establish how the detector interacts with other components of the ESP system or, in other words, what is the system context of the detector. This context is presented in Fig. 1. This sequence diagram is the same as the diagram in [10, Fig. 2, p. 105] excepting some insignificant changes. We use this sequence diagram as a framework for building a mathematical model of event-pattern detectors used in ESP systems.

We accept the following series of suggestions, which is the base for building a mathematical model of the detector.

Suggestion 1. *Each message is characterised by its class that is uniquely determined and belongs the finite predefined set of event classes. A message can contain additional information but this information is used by executive systems only.*

Suggestion 2. *There are a finite set of event-patterns and each event-pattern is a language over event classes.*

Suggestion 3. *Any infinite sequence of messages contains at most one message that a detector accepts and, therefore, classifies.*

Suggestion 4. *A detector halts immediately if it has made a decision about belonging to some class of the received word otherwise it continues waits for an additional information to continue the analysis.*

These suggestions lead us to the following mathematical model, which we call a pattern-detecting machine.

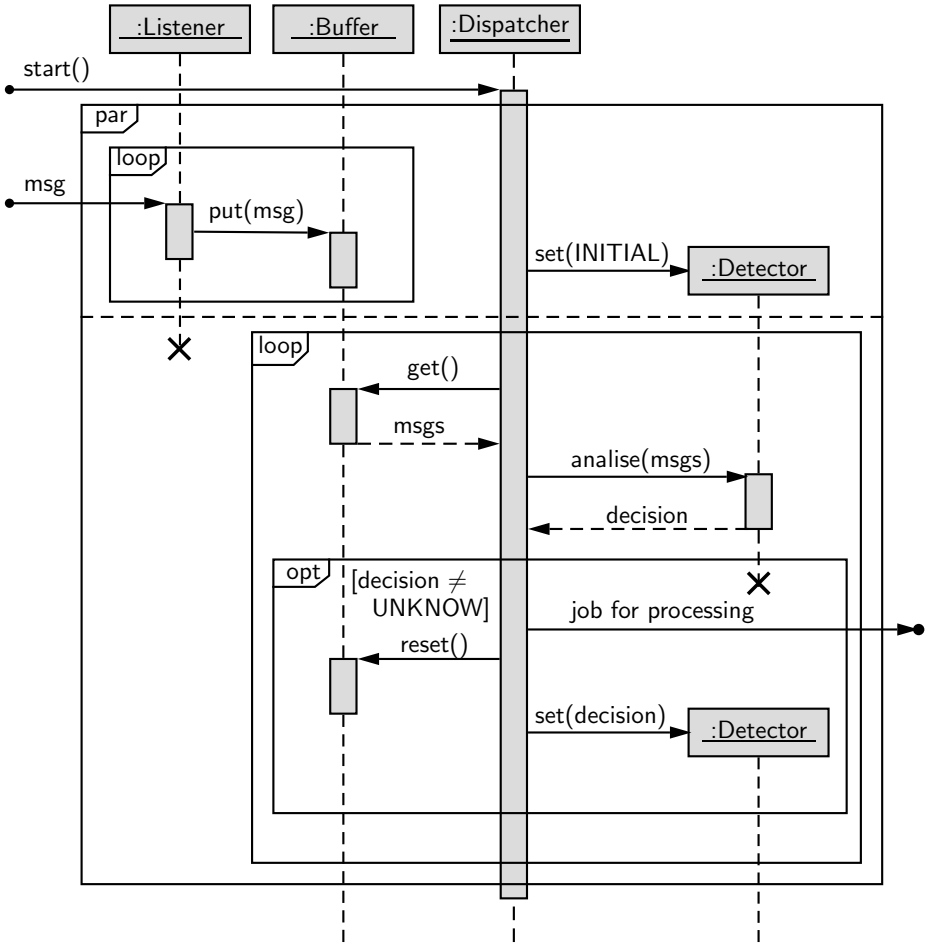

Fig. 1. Detector Context in an ESP System

Definition 1. *Let Σ and Π be finite sets called the event class set and the pattern set respectively then an event detector is a computable prefix-free partial mapping from Σ^+ into Π .*

In this definition the following notation and concepts are used:
Σ^+ refers to the set of all finite non-empty sequences (non-empty words) of elements belonging to Σ;
a prefix-free partial mapping means that this mapping is defined on a subset of words that cannot contain a proper prefix of a word if it contains the word;
a computable mapping means that for the mapping, there exists a Turing

machine that halts on a word if and only if this word belongs to the domain of the mapping and the mapping value on the word coincides with output of the Turing machine.

The above suggestions cause to the conclusion that a self-delimiting Turing machine is adequate and the most general mathematical model of an event-pattern detector. We give the corresponding definition following [8, see Sections 4.3 and 4.4] with minor changes.

Definition 2. *An event-pattern detector with the class set Σ and the pattern set Π where Σ and Π are finite sets is a Turing machine with two tapes, one of which is called the input tape, i.e. it is right-way infinite and its head can move only to the right, and another is called the working tape, i.e. it is two-way infinite and the behaviour of its head is not limited. In addition, Σ is an alphabet of the input type and it is assumed that each its cell has filled before the start of detector operation.*
Finally, elements of Π are uniquely associated with final states of the machine. Thus, the machine makes a decision at the moment of its halting and this decision is the element of Π associated with the corresponding final state.

Bearing in mind that a general self-delimiting Turing machine is an overly complex object, we confine ourselves to some simple subclass of these machines, namely, the subclass of those machines that do not have a working tape.

3 Regular Event-Pattern Detectors

In the case where a self-delimiting Turing machine does not have the working tape, it can be described similarly to a finite state machine.

Definition 3. *A regular event-pattern detector is determined by a quintuple $D = (\Sigma, \Pi, Q, q_0, \delta)$ where*

- *Σ is a finite alphabet of message classes;*
- *Π is a finite alphabet of decisions;*
- *Q is a finite set of machine states;*
- *$q_0 \in Q$ is some selected state, called the initial state;*
- *$\delta\colon Q \times \Sigma \to Q \bigcup \Pi$ is a mapping called the transition function.*

Now we define the relation $\vdash \subset \Sigma^+ \times \Pi$ that models deriving the decision for the given input in the following manner

$$\begin{aligned} &\text{for } a_1 \dots a_n \in \Sigma^+ \text{ and } \alpha \in \Pi\,, \text{ there exists } q_1, \dots, q_{n-1} \in Q \\ &\text{such that} \\ &\quad q_k = \delta(q_{k-1}, a_k) \text{ for all } 0 < k < n\,, \text{ and } \alpha = \delta(q_{n-1}, a_n)\,. \end{aligned} \tag{1}$$

Using mathematical induction gives the following statement.

Proposition 1. *If* $u \vdash \alpha$ *for* $u \in \Sigma^+$ *and* $\alpha \in \Pi$ *then* u *determines uniquely* α *, i.e. if* $u \vdash \alpha$ *and* $u \vdash \alpha'$ *for* $\alpha, \alpha' \in \Pi$ *then* $\alpha' = \alpha$ *.*

Now we can formulate the idea how to use machine learning for the synthesis of regular event-pattern detectors.
Let us assume that for the system being designed, an expert community has found some finite set $T = \{(u_k, \alpha_k) \in \Sigma^+ \times \Pi \mid k = 1, \dots N\}$ such that the subset $\{u_k \mid k = 1, \dots N\} \subset \Sigma^+$ is prefix-free, in other words, for any $0 < k \neq l \leq N$, neither u_k is a prefix of u_l nor u_l is a prefix of u_k . Then we need to find a regular event-pattern detector such that ensures $u_k \vdash \alpha_k$ for all $(u_k, \alpha_k) \in T$ and $k = 1, \dots, N$ under condition that the number of states of the detector is minimal.

4 Regular Detector Synthesis Algorithm

For the synthesis of a regular event-pattern detector according to the aforementioned idea, we use a two-step method. The first step provides to build a tree-like detector D_T and the second step provides to improve this detector reducing it while this is possible. The reduced completely detector D is considered as the required.

The algorithm of the first step is presented in Alg. 1. A simple analysis of this algorithm shows that detector D_T ensures $u \vdash \alpha$ for some $u \in \Sigma^+$ and $\alpha \in \Pi$ if and only if $(u, \alpha) \in T$.

The algorithm of the second step is presented in Alg 2. A simple analysis of the presented algorithm shows

1. for any $(u, \alpha) \in T$, the deriving $u \vdash \alpha$ is preserved for the detector D ;
2. each step of the algorithm loop does not increase the number of states of the detector.

5 Estimation of Algorithm Efficiency

To evaluate the algorithm, we use an experimental environment, that randomly generates a detector D accordingly to Def. 3 based on fixed alphabet Σ, set of decisions Π and a number of possible states M. In these experiments we use common set of possible decisions and a common alphabet. We assign a random $\alpha \in \Pi$ to every possible terminal.

We use such a detector D to generate stochastically paths, ending in terminal nodes. Such paths represents the behaviour of word recognition,

Data: An alphabets Σ and Π and set T
Result: The tree-like detector D_T
$q_0 \leftarrow \epsilon$;
$Q \leftarrow \{q_0\}$;
$T_* = \{u \in \Sigma^+ \mid (u, \alpha) \in T \text{ for some } \alpha \in \Pi\}$;
for $u \in T_*$ **do**
  $Q \leftarrow Q \bigcup \{u' \in \Sigma^+ \mid u' \text{ is any proper prefix of } u\}$
end
$Q \leftarrow Q \bigcup \{\text{trash}\}$;
for $u \in Q$, $a \in \Sigma$ **do**
  if $ua \in Q$ **then**
    def $\delta(u, a) = ua$
  else if $(ua, \alpha) \in T$ **then**
    def $\delta(u, a) = \alpha$
  else
    def $\delta(u, a) = \text{trash}$
  end
end
for $a \in \Sigma$ **do**
  def $\delta(\text{trash}, a) = \text{trash}$
end
return $(\Sigma, \Pi, Q, q_0, \delta)$;

Algorithm 1: Building a tree-like detector

thus for every possible detector D we are able to generate a set of pairs $T = \{(u_k, \alpha_k) \in \Sigma^+ \times \Pi \mid 1 < k \leq N\}$ of recognised words T_* as in Alg. 1.

For every generated detector D from the corresponding D_T using the aforementioned algorithm, we generate a model detector D^*.

The accuracy of the synthesis is estimated as the probability to make the same decision by both detectors D and D^* .

Data: A tree-like detector D_T
Result: The synthesised detector D
$D = D_T$;
for $q \in Q$, $a \in \Sigma$ **do**
  if $\delta(q, a) = \text{trash}$ **then**
    x is the result of random choice from $Q \bigcup \Pi$;
    redef $\delta(q, a) = x$;
    minimise the modified detector D using Hopcroft's algorithms [6]
  end
end
return D

Algorithm 2: Reduction algorithm

After which, we check model's accuracy based on a D. One can see in

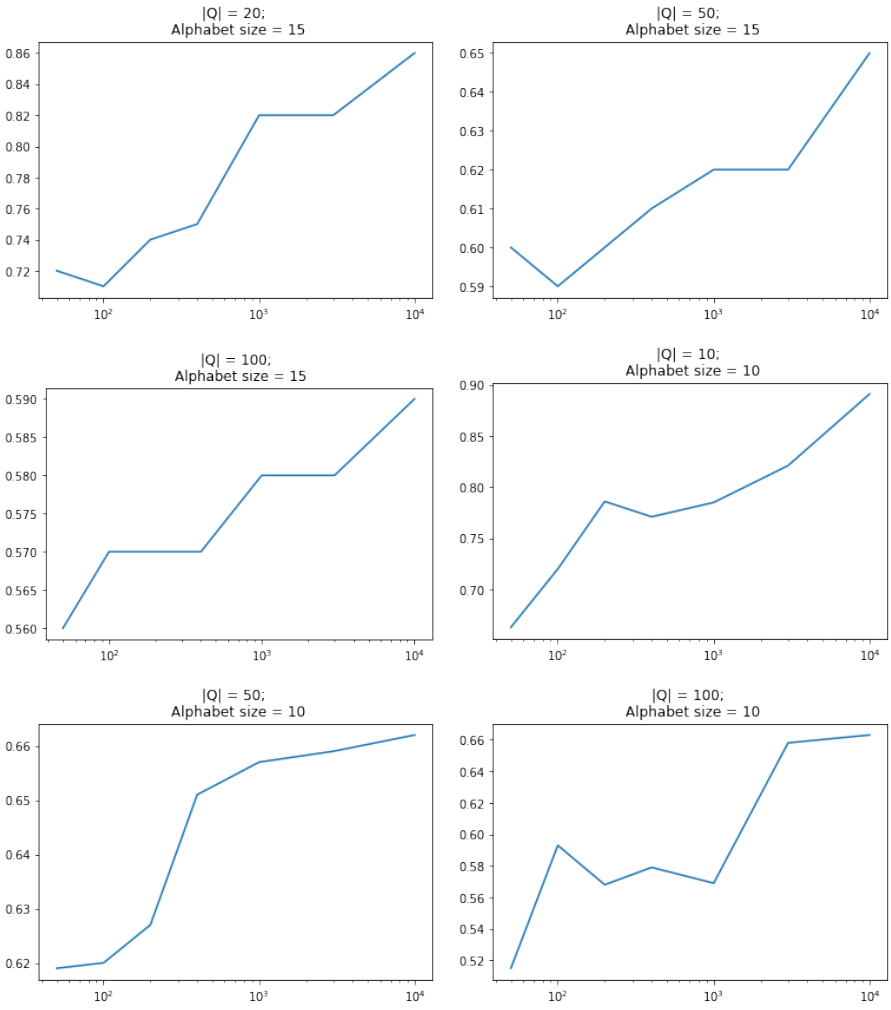

Fig. 2. Consistency between detectors D and D^*

Fig. 2 that the accuracy of the implemented algorithm for a detector with the two-element set of decisions under certain circumstances can reach up to 0.891 .

In the Fig. 3, the dependence of accuracy of synthesis on the number of states of the current detector in the learning process.

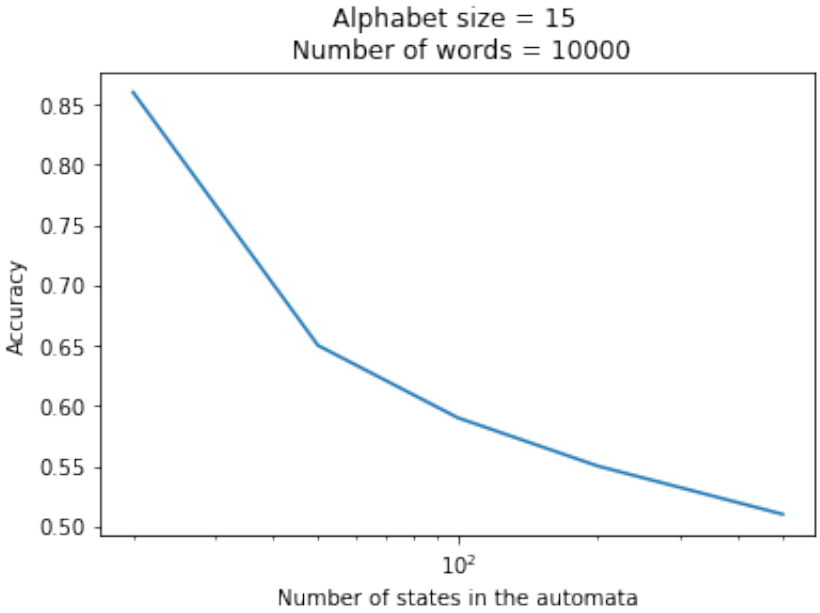

Fig. 3. The dependence of synthesis accuracy on the number of states in the etalon detector

6 Conclusion

Summing up, we can state that the idea of using machine learning algorithms in the design of event detectors is not unreasonable. Nevertheless, this document raises a number of problems that need to be solved in the process of creating information technology for the synthesis of event-pattern detectors on the base machine learning algorithms.

Such problems include

Problem 1. Is it true that $D = \Pr - \lim\limits_{T \to D} D_T$ for any regular event-pattern detector D?

For the special case, when Π contains only two elements this statement transforms into the following statement.

Problem 2. Is it true that $L = \Pr - \lim\limits_{T \to L} T$ where T is finite and $T \subset L$ for any regular language L?

Problem 3. Is it possible to improve the convergence of the algorithm by considering not only the set of confirmatory samples but also the set of counterexamples?

References

1. Overview of the Internet of Things. Recomendation ITU-T Y.4000/Y.2060, International Telecommunication Union (2012)
2. Chandy, K.M., Charpentier, M., Capponi, A.: Towards a theory of events. In: Proceedings of the 2007 Inaugural International Conference on Distributed Event-based Systems. pp. 180–187. DEBS '07, ACM, New York, NY, USA (2007)
3. Dorozhynsky, V.: Regular complex event processing machine. Information Processing Systems 133(8), 82–86 (2015)
4. Dorozhynsky, V.: Mathematical models for specification and analysis of the event-driven system components. PhD Thesis, School of Mathematics and Computer Science at V.N. Karazin Karkiv National University, 4, Svobody sqr., Kharkiv, 61022, Ukraine (2016)
5. Etzion, O., Niblett, P.: Event Processing in Action. Manning Publications Co., Greenwich, CT, USA (2010)
6. Hopcroft, J.: An $n \log n$ algorithm for minimizing states in a finite automaton. In: Theory of machines and computations. pp. 189–196. Academic Press, New York (1971)
7. Khaitan, S.K., McCalley, J.D.: Design techniques and applications of cyber physical systems: A survey. IEEE Systems Journal 9(2), 350–365 (2014)
8. Shen, A., Uspensky, V.A., Vereshchagin, N.: Kolmogorov Complexity and Algorithmic Randomness, Mathematical Surveys and Monographs, vol. 220. American Mathematical Society (2017)
9. Zholtkevych, G.: Realisation of synchronous and asynchronous black boxes using machines. In: V. Yakovyna, et al. (eds.) ICT in Education, Research, and Industrial Applications. CCIS, vol. 594, pp. 124–139. Springer Iternational Publishing, Switzerland (2015)
10. Zholtkevych, G., Novikov, B., Dorozhynsky, V.: Pre-automata and complex event processing. In: V. Ermolayev, et al. (eds.) ICT in Education, Research, and Industrial Applications. pp. 100–116. CCIS, Springer Cham, Heidelberg New York Dordrecht London (2014)