UDC 004.042 + 519.713.2

G.M. Zholtkevych, V.V. Dorozhinsky, A.V. Khadikov

V.N. Karazin Kharkiv National University, Kharkiv

"REAL-TIME" EVENT PROCESSING AND MACHINE LEARNING METHODS

In the paper some machine-learning method for synthesis an on-fly event acceptor is proposed. Such an acceptor appears to be necessary for the technique of the on-fly event processing. This approach is increasingly being used in the design of real-time control and monitoring systems. In the paper the synthesis problem for such an acceptor is considered in a rigorous mathematical formulation. The method for solution of the problem and the computer experiment to study the method are described. The directions for the further research are proposed.

Keywords: *on-fly event processing, acceptor, regular language, prefix- free language.*

Introduction

Trends in the last decade show that technologies like Cloud Computing, Big Data Processing and Internet of Things (IoT) become widely used in the IT-industry and business management for the process monitoring and control in real-time. Usually such systems designed using Event Driven Architecture (EDA). In particular a concept of Complex Event Processing (CEP) is used to manage and control a behaviour of such systems. CEP denotes algorithmic methods for making sense of events by deriving higher-level knowledge, or complex events, from lower-level events in a timely fashion and permanently [6]. CEP uses a number of techniques, including: event-pattern detection; event abstraction; event filtering; event aggregation and transformation; modeling event hierarchies; detecting relationships (such as causality, membership or timing) between events; abstracting event-driven processes [4]. CEP has a lot of applications in the systems where reaction time is critical [7]. For example: Financial systems, operations and services [1]; Fraud detection [9]; Security; Transportation; Health care [8]; Energy; Consumer relations management (including online sales and marketing); Operational intelligence (in business and government); Location-based services; Military applications (monitoring, control and analysis real-time systems).

For modeling CEP systems, the authors proposed in their previous papers a mathematical model called CEP-machine [10] that closely related to the pre-automata [2]. This model takes as its input an infinite stream of atomic events and tries to derive some complex events to make an appropriate action. And as it was shown in [3] by restricting set of complex events to regular, this set can be recognized by the Regular CEP-machine based on the automaton. Further mathematical fundamentals to analyse component behaviour of such a system have been established in the paper [11]. In this paper the authors propose a solution to the problem of CEP system building by the way of computer-aided learning. This approach helps to overcome a complexity of event pattern definitions. The advantage of this technique is needlessness to construct a general theory, covering all logically possible occasions. Instead, it creates a mechanism to adapt the system to new situation that is not consistent with the current knowledge of the system. In the paper the authors present the principal suggestions of our approach and obtained preliminary results in the process of implementing this approach. The rest of the paper is organized as follows. First, section 2 gives basic notation and definitions. Also it contains detailed explanation of CEP-machine concept and its behaviour algorithm. In section 3, we outline the Problem Statement and Key Hypothesis of our study. Section 4 discusses proposed Method for Solution of the Problem. Section 5 outlines possibility of Computer-aided Study of the Method and describes experiment schema. Finally, section 6 draws conclusions and further work.

1. Basic Mathematical Model

In this section we describe the model proposed in [10] for a component of a system designed using EDA. Below the following notation is used:

$f : X \xrightarrow{\text{partial}} Y$ denotes that f is a partial mapping from X into Y;

$f(x)\uparrow$ denotes that $f(x)$ is not defined for the member x of X;

$f(x)\downarrow$ denotes that $f(x)$ is defined for the member x of X;

$f(x)\downarrow= y$ denotes that $f(x)\downarrow$ and $y = f(x)$ for the member y of Y ;

ε denotes the empty (zero-length) sequence;

X^+ denotes the set of all non-empty finite sequences composed of elements of X;

X^* denotes the set $\{\varepsilon\} \cup X^+$;

X^ω denotes the set of all infinite sequences composed of elements of X;

X^∞ denotes the set $X^* \cup X^\omega$;

$|x|$ denotes the length of the finite sequence x;

$x[0]$ denotes the first element of a finite or infinite sequence x;

$x[1:]$ denotes the sequence obtained by removing the first element of the sequence x.

Here some basic definitions needed to describe the mathematical model of a CEP-machine:

Definition 1. The finite set X is called an alphabet and its elements are called symbols. Further, the subset L of X^* is called a language in this context.

The symbols are interpreted as a prime messages informing that the corresponding elementary event has happened. Some finite symbol sequences carry information about the complex events. Below these sequences are called events. In contrast, other finite symbol sequences do not carry information about any complex event. Below they are called words.

For the sets of complex events the certain conditions should be fulfilled. The most important one is that any stream of elementary events is uniquely subdivided into a series of complex events by directed viewing the stream from left to right. This condition leads to the following definition.

Definition 2. Let L be a language with the alphabet X then L is prefix- free if for any $u \in L$ and $v \in X^*$ the assumption $uv \in L$ ensures the equality $v = \varepsilon$.

Now we can fix that any set of complex events related with a system would be prefix-free. Let us briefly recall the principal idea of the used model.

Any component of EDA based system is modeled by using the concept of a CEP-machine.

Definition 3 (CEP-machine Structure). Any CEP-machine is a quintuple $M = (X, Y, H, h_0, \alpha)$ with the following constituents:

– the finite set that is alphabet of atomic messages X;

– the finite set that is alphabet of machine responses Y respectively;

– the finite set of handlers H,.such that each its member is a partial mapping $h : X^+ \xrightarrow{\text{partial}} Y$ with the domain defined as a prefix-free language;

– the initial handler $h_0 \in H$;

– the response function $\alpha : Y \to H$, that is a mapping with domain Y and codomain H.

Remark 1. This definition postulates the presence of two sets of events: the set of events being received from a CEP-machine surroundings and the set of the CEP-machine responses i.e. events being generated by the CEP-machine as reactions on received events.

Further, the set of handlers determines the tool-kit for operating with received events. Each tool of the tool-kit establishes rules to specify how a CEP-machine reacts on one or the other sequence of external events. The condition that a handler domain is prefix-free set allows to identify uniquely such sequences of events that require a certain reaction.

The initial handler is needed to determine how a CEP-machine begins its activity.

The response function supports the feedback - it determines the behaviour of a CEP-machine after some event sequence has been recognized as such that requires some reaction. The behaviour of any CEP-machine is determined by Algorithm 1. This algorithm is simplified, as it may hung in step 5.

Algorithm 1. Operational model of a CEP-machine:

```
1  def run (M,s):
Require :the studied CEP-machine $M = (X, Y, H, h_0, \alpha)$ and some stream of elementary events $s \in X^\omega$
Ensure: printing of the corresponding response stream
2  handler, buff = $h_0$,[]
3  while True :
4      new_event, s = s[0], s[1:]
5      buff.append (new_event)
6      If handler (buff) $\uparrow$: continue
7  else:
8      response = handler(buff)
9      print(response) # printing of the current response
10     handler, buff = $\alpha$(response),[]
```

More precise the algorithm can be found in [10]. But in our study we consider more simple case, which was first considered in [3]. In this case the situation, when the algorithm is hanging in step 5, is impossible.

To specify this special case the following definition is needed.

Definition 4 (Regular handler). A handler $h : X^+ \xrightarrow{\text{partial}} Y$ is called regular if there exist

- some finite set Z with the marked element $z_0 \in Z$ and some mapping $\delta : Z \times X \to Z \bigcup Y$ such that for any $u \in X^+$ and $y \in Y$ the condition $h(u) \downarrow = y$ is fulfilled iff there exist $z_1, ..., z_{|u|-1} \in Z$ such that $z_{i+1} = \delta(z_i, u[i])$ for $0 \le i < |u| - 1$ and $y = \delta(z_{|u|-1}, u[|u|-1])$.

In this case we say that the handler h is realized by the triple (Z, z_0, δ).

Remark 2. It is evident that a handler is regular if there exist a finite state machine realising it.

Definition 5 (Regular CEP-machine). A CEP-machine is called regular if all its handlers are regular.

2. The Problem Statement

In practice we propose restrict our technique by methods of synthesis for regular CEP-machines. This restriction is caused by technical and theoretical difficulties of more general approach.

Further, it is evident that the synthesis problem for a regular CEP- machine is decomposed into series of synthesis problems for regular handlers each of which has only one possible response "accepted". In this case we use the term "a regular acceptor" instead the term "a regular handler".

Therefore, a machine-learning problem for synthesis process of a regular handler can be formulated in the following way.

Problem. Let $E=\{u_1,...,u_M\}$ be a finite prefix-free set of finite sequences composed by elements of X and $C=\{v_1,...,v_N\}$ be a finite set of finite sequences composed by elements of X such that $E\bigcap C=\varnothing$ then we interpret elements of set E as examples and elements of set C as counterexamples;

we need to find a regular acceptor $h:X^+ \xrightarrow{\text{partial}} \{\text{accepted})$ such that the following conditions are fulfilled:

1. $h(u_i)\downarrow=\text{accepted}$ for all $0\le i<M$;
2. $h(v_i)\uparrow$ for all $0\le i<N$; and
3. the corresponding set Z has the least number of elements among all possible. Another words the regular acceptor is minimal.

3. The Problem Solution

Here we present a method to build a regular acceptor and describe a computer experiment that instills confidence in the existence of a mathematical justification for this method. Let us give some theoretical background. We premise our presentation with a few simple theoretical results. The techniques used to prove these results are quite common in automata theory therefore we omit the corresponding proofs for simplicity of the presentation.

First of all, let us return to Remark 2 and discuss in details the interrelation between regular acceptors and finite state machines. In particular, if we consider for any regular acceptor $h:X^+ \xrightarrow{\text{partial}} Y$ that realized by the triple (Z,z_0,δ) the machine such that

1. its set of states Q is equal to $Z\bigcup Y\bigcup\{q_{trap}\}$, where q_{trap} is some state that does not belong to Z and Y, and
2. its transition function $\overline{\delta}:Q\times X\to Q$ continues the mapping δ in the following manner

(a) $\overline{\delta}(y,x)=q_{trap}$ for any $x\in X$ and $y\in Y$;

(b) $\overline{\delta}(q_{trap},x)=q_{trap}$ for any $x\in X$;

(c) its initial state is z_0;

(d) its set of acceptable states is Y

then the regular language accepted by this machine coincides with the set of events accepted by the regular acceptor.

General automata theory guarantees that the built machine can be uniquely minimized without changing language that is accepted by the machine. Note, that the corresponding minimal machine has only one non-acceptable state q such that $\delta(q,x)=q$ for any $x\in X$. We will call this state the trap and denote by trap. Moreover, $\delta(q,x)=\text{trap}$ for any acceptable state q and any $x\in X$.

After this brief theoretical review, we can return to our problem. Now let us describe the proposed method. The general view of the method to find an acceptor is presented by Alg. 2. This method consists of the series of redirections for acceptor transitions leading into the trap starting with the minimal acceptor for the set of examples. Two functions init and modify used the algorithm are specified separately.

Algorithm 2. Specification of the proposed method:

```
1  def learning_method (E,C):
     Require: the finite prefix-free set of events E
              the finite set of words that are not events C
     Ensure : the required acceptor
2    do that:
3      initiate the learning process by applying function init to the set E
       and denoting the result by acceptor
       # initialize the set of transitions that cannot be redirected
4      frozen_transitions =set()
5      while halting condition is not fulfilled:
6        do that:
7          modify acceptor by redirecting a transition leading into the trap and minimize the resulting acceptor wherein the redirected transition cannot belong to frozen_transitions
8        do that:
9          check that acceptor is admissible in the sense that it does not accept any word from C
10       if the checking is successful: continue
11       else:
12         do that:
13           roll-back the modification and add the redirected transition into frozen_transitions
```

To complete the specification of the proposed method we need to describe algorithms for the function init (see item 3 of Alg. 2) and for function modify (see item 7 of Alg. 2). To build the minimal acceptor with the init function the following ideas are used:

1. states of the acceptor are defined recursively as special sets of words;
2. we choose the set E as z_0 and add it to Z;
3. we choose the empty set as trap;
4. if for $x\in X$ in E there is not a word with the first symbol x then assign $\delta(z_0,x)=\text{trap}$ else the set $\{u\in X^*\,|xu\in E\}$ add to Z;

5. repeat recursively this consideration for all member of Z until Z is stabilized;

6. denote the set $\{\varepsilon\}$ by accepted.

The acceptor obtained in this manner is assigned as result of the function init. The function modify is developed using the following idea: To select a transition for redirection we use the following simple remark: the minimal regular acceptor has at most one state that is an attractor, i.e.any transition that goes out from this state has it as a target. Moreover, if this acceptor accepts a finite language then the existence of the attractor is guaranteed. Further, to minimize the new acceptor we use standard Hopcroft's algorithm [5]. The general view of the method to build an acceptor is presented as UML activity diagram in Fig. 1.

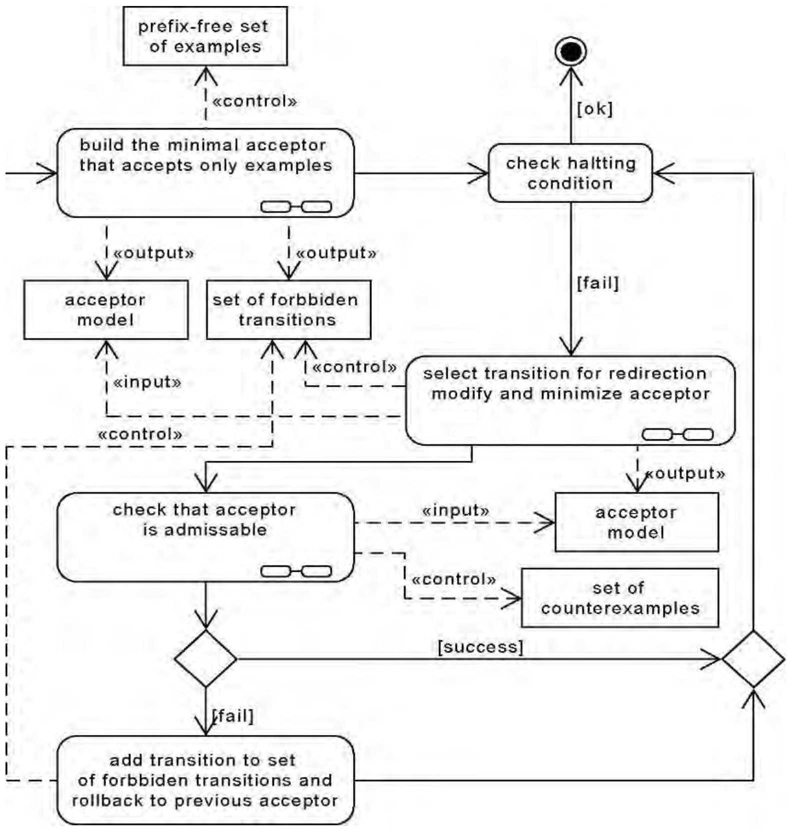
prefix-free set of examples
«control»
build the minimal acceptor that accepts only examples
«output»
«output»
acceptor model
set of forbbiden transitions
«input»
«control»
«control»
[ok]
check haltting condition
[fail]
select transition for redirection modify and minimize acceptor
«output»
check that acceptor is admissable
«input»
acceptor model
«control»
set of counterexamples
[success]
[fail]
add transition to set of forbbiden transitions and rollback to previous acceptor

Fig. 1. The general schema of the method

4. Computer Experiment Schema

Here, we assume that the described above method gives a solution of our problem. To check reasonability of this assumption we designed the computer experiment for searching counterexamples to the assumption. Alg. 3 specifies the schema of the experiment.

Algorithm 3. Schema of computer experiment

```
1  for _ in range(given_number_of_experiments):
2 do that:
3  | generate randomly a regular acceptor  acceptor
4 do that:
5  | generate randomly sets E and C using acceptor
6 acceptor′ = learning _ method (E, C)
7 do that:
8  | compare acceptor′ and acceptor
```

The schema of the experiment is described by the UML diagram in Fig. 2. The results of more than 10,000 experiments have shown that for a randomly generated acceptors we have always been sets E and C such that the presented method of learning was restoring the acceptor using these sets. Thus, we can assume that the proposed method is precise on regular acceptors. The last assumption can be considered as evidence in favour of the validity of the proposed method of machine learning.

Conclusion

The presented paper cannot be considered as complete research. The obtained results are preliminary, but they are very important because they demonstrate a chance to substitute the complete logical analysis of the situation during software development by the learning using the examples and counterexamples for on-fly processing systems.

Also they show that further research in the following directions is needed:

– further experimental study of the proposed method in order to clarify the boundaries of its applicability;

– finding rigorous formulations of the method convergence conditions;

– mathematical justification of the method;

– evaluation of the effectiveness of the method.

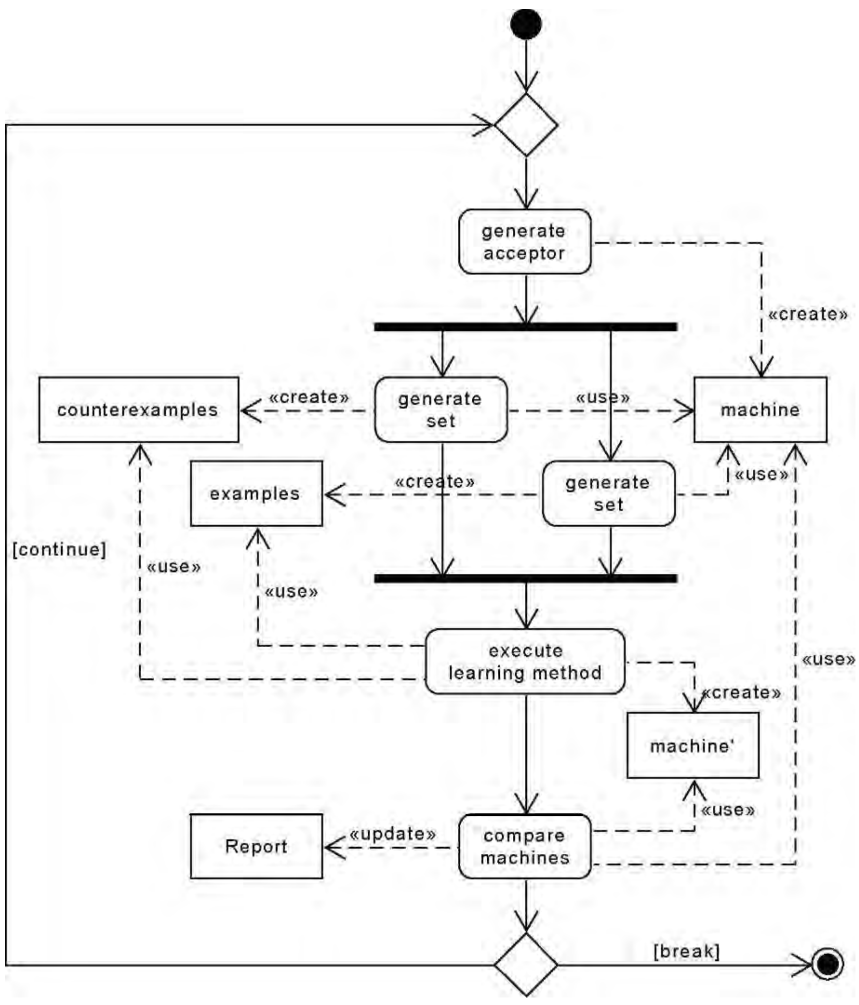

Fig. 2. The general schema of the experiment

References

1. Adi, A., Botzer, D., Nechushtai, G., Sharo, G.: Complex Event Processing for Financial Services. In Proceedings of the IEEE Services Computing Workshops (SCW'06) 0-7695-2681-0/06, (IEEE) (2006).

2. Dokuchaev, M., Novikov, B., Zholtkevych G.: Partial Actions and Automata. Alg. Discr. Math. 11(2), 51-63 (2011).

3. Dorozhinsky, V.: Regular Complex Event Processing Machines. Systemy obrobky informacii. 8, 82-86 (2015).

4. Etzion, O., Niblett, P.: Event Processing in Action. Manning Publications (2010).

5. Hopcroft, J.: An nlogn algorithm for minimizing states in a finite automaton. In: Proc. Internat. Sympos., Technion, Haifa. Theory of machines and computations, pp. 189-196. Academic Press, New York (1971).

6. Luckham, D.: The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems. Addison-Wesley (2002).

7. Luckham, D.C.: Event Processing for Business: Organizing the Real-Time Enterprise. John Wiley & Sons Inc. Hoboken New Jersey (2012).

8. Wang, D., Rundensteiner, E, A.: Active Complex Event Processing over Event Streams. Proceedings of the VLDB Endowment, Seattle, Washington, Ellison III, Richard T. (September 2011).

9. Widder, A., Ammon, R., Hagemann, G., Schnfeld, D.: An Approach for Automatic Fraud Detection in the Insurance Domain. In AAAI Symposia, Spring, SS 09-05017 (2009).

10. Zholtkevych, G., Novikov, B., Dorozhinsky, V.: Pre-Automata and Complex Event Processing. In: V. Ermolayev et al. (eds.) ICTERI 2014. CCIS, vol. 469, pp. 100116. Springer International Publishing, Switzerland (2014).

11. Zholtkevych, G.: Realisation of Synchronous and Asynchronous Black Boxes Using Machines. In: V. Yakovyna et al. (eds.) ICTERI 2015. CCIS, vol. 594, pp 124-139. Springer International Publishing, Switzerland (2016).

Надійшла до редколегії 14.04.2016

Рецензент: д-р техн. наук, проф. Г.А. Кучук, Харківський університет Повітрянх Сил ім. Івана Кожедуба, Харків.

ОБРОБКА ПОДІЙ В РЕАЛЬНОМУ ЧАСІ ТА МЕТОДИ МАШИННОГО НАВЧАННЯ

Г.М. Жолткевич, В.В. Дорожинський, А.В. Хадіков

У статті запропонований метод машинного навчання для синтезу акцепторів у системах обробки складних подій реального часу "на льоту". Поняття акцептор є необхідним у техніці обробки подій реального часу. Такі методи обробки складних подій реального часу все частіше використовуються у розробці систем контролю та моніторингу. Позначена в статті проблема синтезу таких акцепторів викладена із чітким математичним обґрунтуванням. Запропоновані метод розв'язання проблеми та схема комп'ютерного експерименту. Визначені напрямки подальших досліджень у зазначеній галузі.

Ключові слова: *обробка подій реального часу "на льоту", акцептор, регулярна мова, безпрефіксна мова.*

ОБРАБОТКА СОБЫТИЙ РЕАЛЬНОГО ВРЕМЕНИ И МАШИННОЕ ОБУЧЕНИЕ

Г.Н. Жолткевич, В.В. Дорожинский, А.В. Хадиков

В статье предложен метод машинного обучения для синтеза акцепторов в системах обработки сложными событий реального времени "на лету". Понятие акцептор необходимо для техники обработки сложных событий реального времени. Такие методы обработки сложных событий реального времени все в большей мере используются для разработок систем контроля и мониторинга. Рассматриваемая в статье проблема синтеза таких акцепторов представлена строгим математическим обоснованием. Представлены метод решения проблемы и схема компьютерного эксперимента. Обозначены направления дальнейших исследований в указанной области.

Ключевые слова: *обработка событий "на лету", акцептор, регулярный язык, префиксно-свободный язык.*