UDC 004.042 + 519.713.2

V.V. Dorozhinsky

Kharkiv national university named after V.N. Karazin, Kharkiv

REGULAR COMPLEX EVENT PROCESSING MACHINES

In the paper Complex Event Processing Systems are considered. Complex Event Processing 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. The survey of problems and approaches to their solutions associated with this technology that was done by the author with co-authors in their earlier paper led to the development of the mathematical model of such systems. This model is called a CEP-machine. Some properties of CEP-machines related to regular languages are considered in the paper. It is shown that for some classes of complex events that can be represented as a regular language a CEP-machine can be transformed to the automaton that is able to recognize these events.

Keywords: *pre-automaton, complex event, mathematical model, event processing, response function, computable function, regular set.*

Introduction

A modern development of information technologies shows that systems based on events are being widely used in all branches of the industry. Therefore, a problem of efficient event processing becomes very important when designing such systems. In general an event can be described as some phenomena of interest [4]. Events usually occur asynchronously and may vary in size and complexity ranging from simple hardware interrupts to sophisticated database updates. It is known that the handling and processing of asynchronous events is a well-proven and established method used in many computing domains where it supplements synchronous techniques or even represents the norm. Here some examples of event applications:

— Interrupts. Hardware interrupts [9, 11] are low level events by which, for example, an input/output device electrically signals the processor that it needs service. Modern operating systems also leverage interrupts to implement context switches by which they efficiently handle multiple tasks and processes running concurrently.

— Graphical user interfaces. Graphical user interfaces (GUIs) [5, 6] are inherently event-driven. Input data is provided asynchronously in form of user events such as key strokes, mouse clicks or finger taps.

— Active databases. Active databases supplement regular database management systems with the capability to specify reactive behavior[2]. For this purpose, event-condition-action (ECA) rules are often used for specification [8,10].

Another important type of events that currently used in the modern event driven systems is a Complex Events. Complex Event Processing (CEP) is a method of tracking and analysing streams of data about things that happen (they are called events) [7, p. 3] and about effects caused by them. The goal of complex event processing is to identify meaningful events (such as opportunities or threats) and respond to them as quickly as possible [1]. In this paper a mathematical model of Complex Event Processing system is considered. Such a model is called a CEP- machine. Section 2 introduces basic notion and notation that used in the paper. In the sections 3 and 4 a mathematical model of the Complex Event Processing machine (CEP-machine), its properties and relation to the pre-automata are being recalled. These results were first introduced in [12]. Then in section 5 some special class of Complex Event Processing handlers is considered. And finally, section 6 contains some important properties of the CEP-machines which all handlers are regular.

1. Preliminaries

Let X and Y be sets and f be a partial mapping from X into Y then this fact is denoted below by $f : X \xrightarrow{\text{partial}} Y$.

This notation is used in contrast to the notation $f : X \to Y$, which is specified that f is a totally defined mapping.

For a partial mapping $f : X \xrightarrow{\text{partial}} Y$ and some element $x \in X$ the notation $f(x)\downarrow$ is used to indicate that x belongs to the domain of the mapping f. Moreover, if x belongs to the domain of the mapping f and it is known that $f(x) = y$ then the denotation $f(x)\downarrow = y$ is used to express this fact.

Similarly, the notation $f(x)\uparrow$ is used to indicate that x does not belong to the domain of the mapping f.

The domain of the mapping f is denoted below by Df.

Further let Σ be a finite alphabet.

Then we use the following notation:

Σ^* to refer to the free monoid generated by Σ;

ε to denote the unit of this monoid;

Σ^+ to refer to the semigroup of all non-empty words over an alphabet Σ, i.e.

$$\Sigma^+ = \Sigma^* \setminus \{\varepsilon\};$$

$|u|$ to denote the length (a number of symbols) of any $u \in \Sigma^*$.

Any set L consisting of words over alphabet Σ is called prefix-free if the assertions $uv \in L$ and $u \in L$ for $u, v \in \Sigma^*$ imply the equality $v = \varepsilon$.

In other words if for any $L \subset \Sigma^*$ we denote by $C(L)$ the following set $C(L) = \{w \in L \,|\, w = uv$ and $u \in L$ imply $v = \varepsilon\}$.

Then one can easily see that $L \subset \Sigma^+$ is prefix-free if and only if $C(L) = L$.

We consider also infinite sequences of alphabet symbols. To denote the set of all one-way infinite sequences of symbols belonging to the alphabet Σ the notation Σ^ω is used.

For $\pi \in \Sigma^\omega$ and $n \in \mathbb{N}$ we use the notation $\pi[1..n]$ to refer to the word that has a length n and coincides with the beginning of the sequence π.

We denote also by $\pi(n..)$ the sequence belonging to Σ^ω that is defined by the equality

$$\pi = \pi[1..n]\pi(n..).$$

And, finally, we use the notation $\pi[n]$ to refer to the n-th symbol of the sequence π.

2. Formal Model of Complex Event Processing System

In this section we remind the mathematical model of an abstract machine for complex event processing proposed in [12].

Let us choose and fix two finite alphabets Σ and A to describe atomic events and system responses respectively.

Definition 1 (a handler). A partially defined map $h : \Sigma^+ \xrightarrow{\text{partial}} A$ is called a handler if its domain Dh is a prefix-free set.

The equation $C(Dh) = Dh$ in the definition ensures the possibility to identify uniquely if the current input word can be processed or not.

Definition 2 (a CEP-machine). An abstract machine for complex event processing (below a CEP-machine) is a quintuple $(\Sigma, A, H, h_*, \delta)$ where Σ and A are finite alphabets of events and responses respectively, H is a finite set of handlers, $h_* \in H$ is some fixed initial handler, and $\delta : A \to H$ is a total map, which is called a transition map.

This definition specifies the structure of CEP-machines but it does not determine their behaviour. Therefore we need in a series of definitions to specify such a behaviour.

Thus, we suppose that some CEP-machine $(\Sigma, A, H, h_*, \delta)$ has been selected.

Below we call an infinite sequence of elements belonging Σ by event streams and denote the set of all event streams by Σ^ω.

Definition 3 (a configuration). A pair $\langle h, \pi \rangle \in H \times \Sigma^\omega$ is called a configuration.

Now for each handler $h \in H$ let us define the partial function $T_h : \Sigma^\omega \xrightarrow{\text{partial}} \mathbb{N}$ by the following way:

$$T_h(\pi)\downarrow = t \quad \text{if} \quad \pi[1..t] \in Dh;$$
$$T_h(\pi)\uparrow \quad \text{if} \quad (\forall t \in \mathbb{N}) \quad h(\pi[1..t])\uparrow. \qquad (1)$$

Informally, $T_h(\pi)$ equals duration between the present time-point and the first time-point when the handler h can process the corresponding prefix of the event stream π.

Definition 4 (the evolutionary operator). The partial mapping S from the set of configurations into itself defined by the conditions

$$S\langle h, \pi \rangle\downarrow = \langle \delta(h(\pi[1..t]), \pi(t..) \rangle \quad \text{if} \quad T_h(\pi)\downarrow = t$$
$$S\langle h, \pi \rangle\uparrow \quad \text{if} \quad T_h(\pi)\uparrow$$

is called the evolutionary operator of the CEP-machine.

The evolutionary operator is a tool to define correct scenarios of the CEP- machine behaviour.

Definition 5 (a work-flow). An element $w \in (H \times \Sigma^\omega)^\infty$ is called a work-flow for processing of the event stream π if the following conditions are satisfied:

1) $w[0] = \langle h_*, \pi \rangle$ for some $\pi \in \Sigma^\omega$;

2) $w[t+1]$ is defined, if and only if $w[t]$ is defined and belongs to the evolutionary operator domain and, in this case, $w[t+1] = Sw[t]$.

The work-flow definition describes feasible scenarios for the CEP-machine to process streams and, therefore, determines its behaviour.

Thus, Def. 2 and Def. 5 give the complete specification of the CEP-machine.

3. CEP-machines and Pre-automata

As it shown in [12], the concept of a pre-automaton is closely related with the concept of a CEP-machine. This section is devoted to recalling of this relation.

Definition 6 (see [3]). A triple (X,Σ,μ), where X is a set, Σ is a finite alphabet, and

$$\mu : X\times\Sigma^* \xrightarrow{\text{partial}} X$$

is a partial mapping, is called a pre-automaton if the following conditions hold^

1) the equality $\mu(x,\varepsilon)\downarrow = x$ holds for all $x\in X$;

2) if the assertions $\mu(x,u)\downarrow$ and $\mu(\mu(x,u),v)\downarrow$ are true for some $x\in X$ and $u,v\in\Sigma^*$ then the assertion $\mu(x,uv)\downarrow = \mu(\mu(x,u)v)$ is true too;

3) if the assertions $\mu(x,u)\downarrow$ and $\mu(x,uv)\downarrow$ are true for some $x\in X$ and $u,v\in\Sigma^*$ then the assertion $\mu(\mu(x,u),v)\downarrow = \mu(x,uv)$ is true too.

In this context μ, is called a transition function.

Now we describe a manner to associate a pre-automaton with a CEP-machine.

The origin point for our construction is a CEP-machine $M=(\Sigma,A,H,\delta)$. We will find the target pre-automaton as the triple $\widehat{M}=(H,\Sigma,\mu)$ such that the partial mapping $\mu : H\times\Sigma^* \xrightarrow{\text{partial}} H$ satisfies the following condition

for any $h\in H$, $w\in Dh$, *and* $\pi\in\Sigma^\omega$
the conjunction of $\mu(h,w)\downarrow$ and

$$S\langle h, w\cdot\pi\rangle\downarrow = \langle\mu(h,w),\pi\rangle \text{ is true,} \qquad (2)$$

where S is the evolutionary operator of the CEP-machine M.

Theorem 1. Let $M=(\Sigma,A,H,\delta)$ be a CEP-machine and $\mu : H\times\Sigma^* \xrightarrow{\text{partial}} H$ be defined in the following manner:

1) $\mu(h,\varepsilon)=h$ for any $h\in H$;

2) for any $h\in H$, $u\in Dh$, and $v\in\Sigma^*$ the truth of the assertion $\mu(\delta(h(u)),v)\downarrow$ implies that $\mu(h,uv)\downarrow$ is true and in this case $\mu(h,uv)=\mu(\delta(h(u)),v)$;

3) $\mu(h,w)\downarrow$ for all other cases, then the triple $\widehat{M}=(H,\Sigma,\mu)$, is a pre-automaton and the mapping μ satisfies *(2).*

Proof of the Theorem 1 can be found in [12].

Definition 7 (the dual pre-automaton for a CEP-machine).

Let $M=(\Sigma,A,H,\delta)$ be a CEP-machine and $\widehat{M}=(H,\Sigma,\mu)$ be the pre-automaton determined by Theorem 1 then $\widehat{M}$ is called the dual pre-automaton for the CEP-machine M.

We can carry out the converse construction too and associate with any pre- automaton P a CEP-machine M so as to satisfy the equality $P=M$.

Proposition 1. Suppose that a pre-machine $P=(X,\Sigma,\mu)$, is given and let us define the quadruple $\hat{P}=(X,\Sigma,H,\delta)$ such that

$$H=\left\{h_x : \Sigma^+ \xrightarrow{\text{partial}} X \,\middle|\, x\in X\right\} \text{ where}$$

$$Dh_x=\left\{w\in\Sigma^+ \,\middle|\, \mu(x,w)\downarrow \text{ and if } w=uv\wedge\mu(x,u)\downarrow \text{ then } v=\varepsilon\right\},$$

$$h_x(w)=\mu(x,w) \text{ for } w\in Dh_x;$$

$$\delta(x)=h_x.$$

Then $\hat{P}$ is a CEP-machine and $\hat{P}$ is isomorphic to P. Proof of the proposition can be found in [12].

4. Regular Handlers

In this section we consider and study some class of handlers that is closely connected with finite automata. Handlers of this class we call regular handlers.

Definition 8 (a regular handler). A handler $h : \Sigma^* \xrightarrow{\text{partial}} A$ is called regular if its domain Dh is a regular set.

Example 1. *Let us consider a finite automaton* $(S,\Sigma,\gamma : S\times\Sigma\to S)$ *and select some initial state* s_* *and the set of accepting states* $S_{acc}\subset S$. *Our selection should satisfy the following conditions:*

1) $s_*\notin S_{acc}$;

2) $(\bigcup_{a\in\Sigma}\gamma(S_{acc},a))\cap S_{acc}=\varnothing$;

3) *each element of* S_{acc} *is unreachable from* $\bigcup_{a\in\Sigma}\gamma(S_{acc},a)$/

Now let us give $\beta : S_{acc}\to A$ and define $h(u)=\alpha\in A$ if $\gamma^*(s_*,u)\in S_{acc}$ and $\beta(\gamma^*(s_*,u))=\alpha$.

We claim that h is a handler.

Proof. Truly, let us consider a set of words $L=\left\{w \,\middle|\, \gamma^*(s_*,w)\in S_{acc}\right\}$. One can easily see that the set L is a prefix-free set. Indeed, let some words $u, w=uv\in L$ then word w can be accepted by the

automaton only if $v = \varepsilon$. Thus $\forall w \in L$, $\beta(\gamma^*(s_*, w)) \in A$ and $Dh \subseteq L$.

Therefore Dh is a prefix-free set and hence h is a handler (see definition 1).□

Definition 9 (A realization). Let $h : \Sigma^+ \xrightarrow{\text{partial}} A$ be a handler such that exists a finite automaton

$$(S, \Sigma, \gamma : S \times \Sigma \to S), s_*, S_{acc}, \beta$$

as in Example 1, and the equation $h(\cdot) = \beta(\gamma^*(s_*, \cdot))$ is satisfied then $((S, \Sigma, \gamma), s_*, S_{acc}, \beta)$ *is* called a realization of h.

The principal result is the following.

Theorem 2. A handler $h : \Sigma^+ \xrightarrow{\text{partial}} A$ is regular if and only if there exists a realization for it.

Proof. First let us show that if handler is regular then there is exist its realization. Indeed, if some handler $h \in H$ is regular then its domain Dh is a regular set. Thus there exists an automaton $(S, \Sigma, \gamma : S \times \Sigma \to S)$ such that $\forall w \in Dh, \gamma^*(s_*, w) \in S_{acc}$ and one can define mapping $\beta : S_{acc} \to A$ such that

$$\beta(\gamma^*(s_*, w)) = h(w) = \alpha \in A.$$

And thus there exists the realization of the regular handler h such that $((S, \Sigma, \gamma), s_*, S_{acc}, \beta)$. Next let us show that if for some handler h there is exist a realization $((S, \Sigma, \gamma), s_*, S_{acc}, \beta)$ then the handler h is regular.

Really, if $\forall w \in Dh$ the equation $\beta(\gamma^*(s_*, w)) = h(w)$ is fulfilled then Dh is a regular set and thus h is a regular handler□.

5. Regular CEP-machines

In this section we are going to consider some subclass of the CEP-machines which handlers are regular. This leads to the following definition.

Definition 10 (A regular CEP-machine). A CEP-machine $(\Sigma, A, H, h_*, \delta)_R$ *will be called regular if all its handlers are regular.*

Theorem 3 (Regular CEP-machine transformation). A regular CEP-machine

$$M = (\Sigma, A, H, h_*, \delta)_R$$

can be transformed to the finite automaton that is a globalization (for definition of the globalization see [3]) of its dual pre-automaton

$$\widehat{M} = (X_H, \Sigma, \mu : X_H \times \Sigma^* \xrightarrow{\text{partial}} X_H).$$

Proof. To prove that regular CEP-machine M can be transformed to the finite automaton let us first build its dual pre-automaton $\widehat{M}$ and then build this pre-automaton's globalization.

1. To build a dual pre-automaton

$$\widehat{M} = (X_H, \Sigma, \mu : X_H \times \Sigma^* \xrightarrow{\text{partial}} X_H)$$

let us define a total map $\varphi : H \to X_H$ such that for any $h \in H$ $\varphi(h) = x_h \in X_H$ and the following equation is fulfilled:

$\forall w \in Dh$, $\varphi(\delta(h(w))) = \mu(x_h, w)$ and using theorem 1 one can show that such pre-automaton $\widehat{M}$ is dual to the regular CEP-machine M.

2. Now let us build a globalization

$$(S^R, \Sigma, \gamma^R : S^R \times \Sigma \to S^R)$$

to the preautomaton $\widehat{M}$. To do this let us note that each regular handler $h \in H$ has the realization $((S, \Sigma, \gamma), s_*, S_{acc}, \beta)_h$. So for all $h \in H$ let us assign for initial state s_* of the realization $s_* = x_h \in X_H$ and $\mu(x_h, w)\downarrow = \gamma^*(s_*, w) \in S_{acc}$ for any $w \in Dh$.

Thus $(S^R, \Sigma, \gamma^R) = \bigcup_{h \in H} (S, \Sigma, \gamma)_h$

More precisely:

- $S^R = \bigcup_{h \in H} (S \setminus S_{acc})_h$
- $\gamma^R = \bigcup_{h \in H} \gamma_h$

Also one can easily see that $X_H \subseteq S^R$.

Therefore the automaton (S^R, Σ, γ^R) is a globalization for the pre-automaton

$$\widehat{M} = (X_H, \Sigma, \mu).$$

It is evident that converse statement "a finite automaton can be transformed to the regular CEP-machine" is true too. Since any automaton can be represented as a pre-automaton (see [3]) that is dual to the regular CEP-machine.

Conclusion

In the paper the mathematical model of Complex Event Processing system was considered. The concepts of the CEP-machine and pre-automaton were used. The section 2 gave some basic notion and notations that were used in the paper. In the sections 3, 4 the mathematical model of the CEP-machine, its properties and relation to the pre-automata was recalled. Then in the section 5 some specific class of the Complex Event Processing handlers was considered. These handlers were called regular handlers.

The theorem 2 gives a principal result that a regular handler can be realized by the automaton.

Finally, section 5 presented important sub-class of CEP-machines called by the author Regular CEP-machines.

A theorem 3 proofs that regular CEP-machine can be transformed to the automaton that is a globalization to the dual pre-automaton of this regular CEP-machine.

The results obtained in the paper may be useful in modeling and designing of some class of the Complex Event Processing systems that should be capable to recognize the events such that can be described by the regular language.

References

1. Bates J. Secrets Revealed / J. Bates //Trading Tools Uncover Hidden Opportunities. Global Trading, May 14, (2012). [Електронний ресурс]. – Режим доступу до ресурсу: http://fixglobal.com/home/secrets-revealed-trading-tools-uncover-hidden-opportunities/

2. Dittrich K.R. The Active Database Management System Manifesto. A Rulebase of ADBMS Features / K.R. Dittrich, S. Gatziu, A. Geppert // Proceedings of the 2nd International Workshop on Rules in Database Systems (RIDS '95). Ed. by T. Sellis. Vol. 985. Lecture Notes in Computer Science. Glyfada, Athens, Greece. – Springer, Sept. 1995. – P. 3-20.

3. Dokuchaev M. Partial Actions and Automata / M. Dokuchaev, B. Novikov, G. Zholtkevych // Alg. Discr. Math. – 2011. – 11(2).

4. Gehani N.H. Composit Event Specification in Active Databases. Model & Implementation / N.H. Gehani, H.V. Jagadish, O. Shmueli // Proceedings of the 18th International Conference on Very Large Data Bases (VLDB'92). Ed. by L.-Y. Yuan. Vancouver, Canada: Morgan Kaufmann, Aug. 1992. – P. 327-338.

5. Krasner G.E. A Cookbook for Using the Model-View-Controller User Interface Paradigm in Smalltalk-80 / G.E. Krasner, S.T. Pope // Journal of Object Oriented Programming 1.3 (Aug. 1988). – P. 26-49.

6. Linton M.A. Composing User Interfaces with InterViews. Tech. rep. CSL-TR-88-369 / M.A. Linton, J.M. Vlissides, P.R. Calder. – Stanford, CA, USA: Stanford University, Nov. 1988.

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

8. McCarthy D. The Architecture of an Active Database Management System / D. McCarthy, U. Dayal; ed. by J. Clifford, B. Lindsay, D. Maier. – Portland, OR, USA: ACM Press, 1989. – P. 215-224.

9. Mersel J. Program Interrupt on the Univac Scientific Computer / J. Mersel // Proceedings of the Western Joint Computer Conference (WJCC'56). – San Francisco, CA, USA: ACM, Feb. 1956. – P. 52-53.

10. Paton N.W. Active Database Systems / N.W. Paton, O. Diaz // ACM Computing Surveys 31.1 (Mar. 1999). – P. 63-103.

11. Rosen S. Electronic Computers: A Historical Survey / S. Rosen // ACM Computing Surveys 1.1. – Mar. 1969. – P. 7-36.

12. Zholtkevych G. Pre-Automata and Complex Event Processing / G. Zholtkevych, B. Novikov, V. Dorozhinsky. – Springer International Publishing Switzerland V. Ermolayev et al. (Eds.): ICTERI 2014, CCIS 469, 2014. – P. 100-116.

Надійшла до редколегії 16.06.2015

Рецензент: д-р техн. наук, проф. Г.М. Жолткевич, Харківський національний університет ім. В.Н. Каразіна, Харків.

РЕГУЛЯРНІ МАШИНИ ОБРОБКИ СКЛАДНИХ ПОДІЙ

В.В. Дорожинський

Стаття присвячена системам обробки складних подій. Обробка складних подій визначається алгоритмічними методами розпізнавання подій, що базуються на отриманні більш загальної інформації, або складних подій, за допомогою неперервного аналізу набору конкретних даних, або простих подій, за прийнятний час. Дослідження проблем та методи їх розв'язання, що пов'язанні з даною технологією, проведене автором разом з співавторами в в їх попередній статті, призвело до розробки математичної моделі таких систем. Ця модель називається машиною обробки складних подій (CEP-machine). Також в статті розглядаються деякі властивості машин обробки складних подій, пов'язанні з регулярними мовами. Показано, що для деяких класів складних подій, що можуть бути описані як регулярна мова, машина обробки складних подій може бути перетворена на автомат, що може розпізнавати ці події.

Ключові слова: *перед-автомат, складна подія, математична модель, обробка подій, функція відгуку, обчислювальна функція, регулярна множина.*

РЕГУЛЯРНЫЕ МАШИНЫ ОБРАБОТКИ СЛОЖНЫХ СОБЫТИЙ

В.В. Дорожинский

В статье рассматриваются системы обработки сложных событий. Обработка сложных событий определяется алгоритмическими методами распознавания событий, основанными на получении более общей информации (комплексных событий) посредствам непрерывного анализа набора конкретных данных (простых событий) за приемлемое время. Исследование проблем и подходов к их решению, связанных с этой технологией, проведенное автором с соавторами в их предыдущей статье, привело к разработке математической модели таких систем. Эта модель называется машиной обработки сложных событий (CEP-machine). В статье также рассматриваются некоторые свойства машин обработки сложных событий, связанные с регулярными языками. Показано, что для некоторых классов сложных событий, которые могут быть описаны регулярным языком, машина обработки сложных событий может быть преобразована в автомат, который способен распознавать эти события.

Ключевые слова: *пред-автомат, сложное событие, математическая модель, обработка событий, функция отклика, вычислимая функция, регулярное множество.*