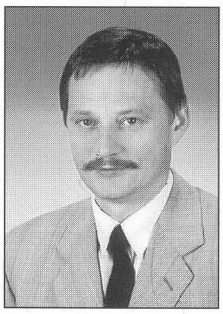
Jacek KLUSKA, Lesław GNIEWEK
POLITECHNIKA RZESZOWSKA, KATEDRA INFORMATYKI I AUTOMATYKI

Fuzzy Petri nets as control systems

Dr hab. inż. Jacek KLUSKA



Absolwent Wydziału Elektroniki Politechniki Wrocławskiej (1977, specjalność systemy cybernetyki technicznej). Od 1977 r. zatrudniony w Politechnice Rzeszowskiej. W 1983 r. uzyskał stopień doktora w Politechnice Wrocławskiej. Stopień doktora hab. uzyskał w 1995 r. w AGH w Krakowie. Obecnie profesor nadzwyczajny Politechniki Rzeszowskiej w Katedrze Informatyki i Automatyki. Jego publikacje dotyczą projektowania układów regulacji rozmytej, sztucznej inteligencji, sieci Petriego i projektowania systemów czasu rzeczywistego.

e-mail: jacklu@prz-rzeszow.pl

Dr inż. Lesław GNIEWEK

Absolwent Wydziału Elektrycznego Politechniki Rzeszowskiej (1991, specjalność automatyka i metrologia). Od 1991 r. zatrudniony w Politechnice Rzeszowskiej. W 1999 r. uzyskał stopień doktora na Wydziale Informatyki i Zarządzania Politechniki Wrocławskiej. Obecnie adiunkt w Politechnice Rzeszowskiej w Katedrze Informatyki i Automatyki. Jego publikacje dotyczą projektowania sprzętu cyfrowego, logiki rozmytej, sieci Petriego i programowalnych sterowników logicznych.

e-mail: lgniewek@prz-rzeszow.pl

Abstract

Industrial processes can often be modelled using Petri nets. If all the process variables are assumed to be two-valued signals, then it is possible to obtain a control device, which works according to the algorithm described by conventional Petri net. However, the values of real signals are contained in some bounded interval, and therefore they can be interpreted as events, which are true in some degree from the interval [0,1]. Such a natural interpretation concerns sensor outputs, control signals, time expiration, etc. It leads to the idea of the fuzzy Petri net as a controller, which is able to process both analog, and binary signals. Such the net is presented in this paper.

Streszczenie

Procesy przemysłowe często mogą być zamodelowane za pomocą sieci Petriego. Jeżeli zmienne procesowe są traktowane jako dwuwartościowe, to można otrzymać urządzenie sterujące, które działa według algorytmu opisanego przez konwencjonalną sieć Petriego. Jednak wartości sygnałów rzeczywistych zawarte są w pewnym skończonym przedziale, więc mogą być zinterpretowane jako zdarzenia prawdziwe w pewnym stopniu z przedziału [0,1]. Taka naturalna interpretacja dotyczy wyjść czujników, sygnałów sterujących, upływu czasu itd. Prowadzi to do idei rozmytych sieci Petriego jako układów sterowania, przetwarzających zarówno sygnały analogowe, jak i binarne. Takie sieci są przedstawione w niniejszym artykule.

1. Introduction

Petri nets are widely used to model computer systems, digital circuits, design of control systems and real-time software, communication protocols, information systems, knowledge-based systems, man-machine interfaces, flexible manufacturing systems, transport systems, etc. Modeling a system using Petri nets has many advantages in comparison with the other schemes; the language provided by such nets is a great help in representation of the behavior of concurrent systems. The graphical aspect makes it easier to represent the different interactions between discrete events: parallelism, synchronization, sequence, alternatives, nondeterminism and so on [8, 20, 24, 25]. Owing to Petri nets it is possible to exact modeling such interactions and the analysis of the investigated system.

Until now several extensions for improving different aspects of the Petri nets: high-level, hierarchical, temporal or stochastic Petri nets [8, 20], have been proposed. There has been an increasing interest in extending the fundamental concepts of Petri nets to incorporate the capabilities for handling fuzziness [27] in systems modeling. The studies carried out in this way have originated a new class of models which are called fuzzy Petri nets [2, 3, 5, 6, 18, 22]. In 1988, Looney as a first researcher proposed a Petri net model using fuzzy logic in [18], and developed a fuzzy reasoning algorithm for rule-based decision making by propositional logic. Since then, several authors from artificial intelligence and Petri nets communities, have proposed different kinds of fuzzy Petri nets. However, under the same name, these models are based on different approaches, which combine Petri nets and fuzzy logic. Some researchers used such the nets to deal with the knowledge representation and fuzzy reasoning [2, 5, 6, 22]. The other new conceptions appeared, as well. For example in [21], so called continuous fuzzy Petri net tool, which integrates the three technologies of fuzzy control, Petri nets and real-time knowledge-based systems was developed. This tools was implemented in the real-time expert system environment of Imperial Oil Ltd. (ESSO Canada).

Industrial processes can be often decomposed into many parallel subprocesses, which can, in turn, be modelled using Petri nets. For this reason, Petri net formalism seems to be more and more promising in the area of automatic control systems design. Some useful concepts of Petri nets has been developed and used for modeling of dynamic systems [8], and so called "interpreted Petri nets", which has been used for real-time software design for the programmable logic controllers (Grafcet, SFCs) [7, 11]. The interpreted Petri net is slightly close to the idea of the net discussed in this paper, but in this case only, when we use classical logic.

As a result of the use of Petri nets and design methods based on Boolean logic, a number of procedures have been developed for the synthesis of the so called reconfigurable logic controllers [23], and parallel controllers [4, 10]. The constructed models based on Petri nets and traditional logic can be directly applied for implementing control system software or hardware [1, 19]. Unlike microprocessor-based software implementations, the hardware-based methods offer enough speed to control fast plants at low cost. The method described in [19] offers an attractive tool for engineers, because it provides a simple transformation procedure of the Petri net into the logic circuit. However, the resulting system as a computer program or hardware device, is capable of processing binary information only. The net described in this paper is a step forward in comparison with [19], because it is based on fuzzy logic. As a result it would be possible to design the fuzzy net, which has clear and natural interpretation as a formal description of a control algorithm. One will be able to design this fuzzy net as a computer program or - what is more important, as a hardware device, using new fuzzy hardware components [12, 14]. From the engineer-designer point of view it is important that we propose to use fuzzy logic in a very natural way. This net will be different from those described in the literature. We will describe the idea of the net, give the basic features of this net, and give a simple example of control system modeling.

2. Conception of fuzzy Petri net

First we will give a definition and basic assumptions. The fuzzy Petri net (FPN for short) is a system:

$$FPN = (P, T, D, G, R, \Delta, \Gamma, \Theta, M_0) \qquad (1)$$

which contains the following sets:
- places $P=\{p_1, \ldots, p_r\}$,
- transitions $T=\{t_1, \ldots, t_s\}$,
- statements $D=\{d_1, \ldots, d_r\}$,
- conditions $G=\{g_1, \ldots, g_s\}$,

where any two of the sets P, T, D, G have no common elements,
- incidence relation $R \subseteq (P\times T) \cup (T\times P)$,
- the function Δ: $P\rightarrow D$, assigning the statement for any place,
- the function Γ: $T\rightarrow G$, assigning the condition for any transition,
- the function Θ: $T\rightarrow[0,1]$, defining the degree to which the conditions corresponding to the transitions are satisfied, and
- the initial marking function M_0: $P\rightarrow[0,1]$.

We restrict ourselves to the specific case of the nets. Let us denote by ${}^\circ t = \{p \mid (p, t)\in R\}$ - the set of input places for the transition t, and by $t^\circ = \{p \mid (t, p)\in R\}$ - the set of its output places. The considered net is assumed to be clean, i.e. without such pairs (t, p) or (p, t), that $p\in {}^\circ t\cap t^\circ$. All capacities of the places and all weights of the arcs are assumed to be 1. We allow conflict in the net, i.e. the situation, when the number of input or output transitions for a given place is greater than 1. For determining which of two or more events should occur, an additional information is needed, which is not contained in the system description; it should be delivered from the environment [26]. We assume that such information is available.

3. Petri net dynamics

The basis of algebraic description of the net are incidence matrices C^+ and C^-, whose elements are from the set $\{0,1\}$. If the number of transitions is s and the number of places is r, then the matrices $C^+ = \{c^+_{i,j}\}$ and $C^- = \{c^-_{i,j}\}$ have the dimension $s\times r$, and their elements are defined as follows:

$$c^+_{i,j} = 1 \Leftrightarrow (t_i, p_j) \in R, \text{ and } c^+_{i,j} = 0 \Leftrightarrow (t_i, p_j) \notin R, \qquad (2)$$

$$c^-_{i,j} = 1 \Leftrightarrow (p_j, t_i) \in R, \text{ and } c^-_{i,j} = 0 \Leftrightarrow (p_j, t_i) \notin R, \qquad (3)$$

Petri net dynamics defines how new marking is computed from the current marking, when the transitions are fired. Only enabled transitions can be fired.

In the sequel the marks $\wedge$ and $\vee$ will be used for minimum and maximum operations, respectively. Let FPN be the net with the marking M: $P\rightarrow[0,1]$. The transition $t\in T$ is enabled from the moment at which

$$\forall\, p\in {}^\circ t,\ M(p) = 1 \text{ and } \forall\, p\in t^\circ,\ M(p) = 0, \qquad (4)$$

are satisfied, to the moment at which

$$\forall\, p\in {}^\circ t,\ M(p) = 0 \text{ and } \forall\, p\in t^\circ,\ M(p) = 1, \qquad (5)$$

hold.

Let M be the marking in FPN, for which the transition $t\in T$ is enabled, and $\Theta(t) = \vartheta \in [0,1]$ denotes the degree to which the condition corresponding to the enabled transition t is satisfied. New marking M' of the net is computed according to the formula:

$$M'(p) = M(p) \wedge (1 - \vartheta) \quad \text{for} \quad p\in {}^\circ t \setminus t^\circ,$$

$$M'(p) = M(p) \vee \vartheta \quad \text{for} \quad p\in t^\circ \setminus {}^\circ t,$$

$$M'(p) = M(p) \quad \text{otherwise.} \qquad (6)$$

Now let us define for the matrices $A = \{a_{ij}\}_{n\times m}$, $B = \{b_{ij}\}_{n\times m}$, and $C = \{c_{jk}\}_{m\times s}$, the following operations:

$$U = \{u_{ij}\}_{n\times m} = A \wedge B \Leftrightarrow u_{ij}=a_{ij} \wedge b_{ij}, \qquad (7)$$

$$V = \{v_{ij}\}_{n\times m} = A \vee B \Leftrightarrow v_{ij}=a_{ij} \vee b_{ij}, \qquad (8)$$

$$W = \{w_{ij}\}_{n\times m} = \sim A \Leftrightarrow w_{ij} = 1 - a_{ij}, \qquad (9)$$

$$Y = \{y_{ik}\}_{n\times s} = A \blacklozenge C \Leftrightarrow y_{ik} = \max_{j=1,\ldots,m} \{a_{ij} \wedge c_{jk}\}, \qquad (10)$$

for $i = 1, 2, \ldots, n$, $j = 1, 2, \ldots, m$, and $k = 1, 2, \ldots, s$.

In order to allow analysis of the FPN we need a procedure of how to compute new marking of the net from the current marking. A compact procedure in this range delivers a theorem, which we proved in [13].

Let us consider a clean FPN, in which capacities of the places and weights of the arcs are equal to 1, the number of transitions is s, and the number of places is r. Assume that there are some transitions in the net, which by the marking M are enabled according to (4)-(5). After firing the above transitions, the next marking M' is given by:

$$M' = [\sim\Theta' \blacklozenge C^-] \wedge [(\Theta' \blacklozenge C^+) \vee M], \qquad (11)$$

where

$$\Theta' = E \wedge \Theta. \qquad (12)$$

The vector E is of the length s and has coefficients from the set $\{0,1\}$, in which the given coordinate is equal to 1 if and only if, it corresponds to the enabled transition by the marking M in the net. Θ is the vector of the length s with coordinates from the interval $[0,1]$, in which the given coefficient ϑ_i, $(i = 1, 2,..., s)$, describes the degree to which the condition corresponding to the transition $t_i\in T$ is satisfied.

4. Features of the net

According to definition of FPN and assumptions, the net has two important features.

The first one is as follows. Suppose that the transition t is enabled. At any moment and independently from the degree ϑ, the sum of fuzzy markers in any input place $p_i\in {}^\circ t$ and any output place $p_k\in t^\circ$ is equal to 1, i.e.:

$$\Theta(t) = \vartheta \in [0,1] \Rightarrow M(p_i) + M(p_k) = 1. \qquad (13)$$

At the moment at which the transition t is enabled, the above condition is satisfied for any $p_i\in {}^\circ t$ and $p_k\in t^\circ$. If the condition which corresponds to the transition t is satisfied in the degree ϑ, then for the next marking resulting from marker transferring across the transition t, the same condition holds independently from ϑ. This fact follows from the assumptions and de Morgan's law, which is true in fuzzy logic:

$$\begin{aligned} M'(p_i) + M'(p_k) &= [\sim\vartheta \wedge M(p_i)] + [\vartheta \vee M(p_k)] \\ &= [\sim\vartheta \wedge \sim M(p_k)] + [\vartheta \vee M(p_k)] \\ &= \sim[\vartheta \vee M(p_k)] + [\vartheta \vee M(p_k)] = 1. \end{aligned} \qquad (14)$$

Thus, we say about "marker transferring" across the transition.

Now we explain the second feature of the net. If the degree of satisfying the condition which corresponds to the enabled transition t decreases, then the marking of places $p\in {}^\circ t\cup t^\circ$ remains the same. To prove this, let us suppose that, for the given marking M the transition $t\in T$ is enabled. If the condition which corresponds to the transition is satisfied in the degree ϑ, then using (6) one can compute the new marking M'. If for the marking M' the degree of satisfaction the condition which corresponds to transition t will decrease from ϑ to ϑ', $(\vartheta > \vartheta')$, then the next marking M'' one can compute from (6) as the new one from the current marking M'. This will be done for all three cases.

For $p\in {}^\circ t \setminus t^\circ$ we obtain the next marking M'' as follows:

$$M''(p) = M'(p) \wedge \sim\vartheta' = \sim\vartheta \wedge M(p) \wedge \sim\vartheta' = M'(p). \qquad (15)$$

For $p\in t^\circ \setminus {}^\circ t$, the next marking M'' is computed as:

$$M''(p) = M'(p) \vee \vartheta' = \vartheta \vee M(p) \vee \vartheta' = M'(p), \qquad (16)$$

and finally, for $p\notin {}^\circ t \cup t^\circ$, we obviously obtain:

$$M''(p) = M'(p). \qquad (17)$$

This means that decreasing the degree associated with the transition t does not cause changes in the marking of the net, i.e. $M''(p) = M'(p)$. Such behavior prevents the net from direction changing of the marker, which is being transferred across the transition.

5. Example

Let us consider a simple example of two elevators for taking goods from the level I to 0 and from the level I to II as shown in Fig. 1.

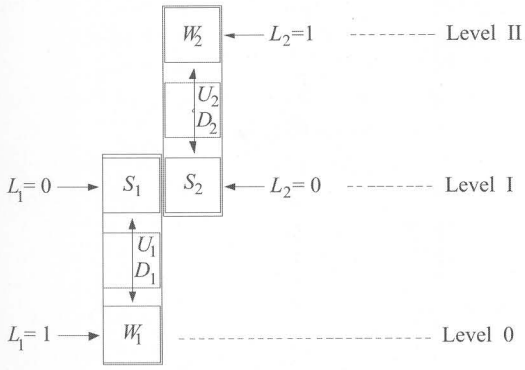


Fig. 1. The system of two elevators.
Rys. 1. Układ dwóch wind towarowych.

At the beginning, both elevators are in the states S_1 and S_2 correspondingly, and they are located on the level I. The sensors indicate: L_1=0 and L_2=0. After a passage of time T, which is the common waiting time for both elevators, the first elevator is moved down due to the signal D_1, which controls its drive (D_1=1). At the same time, the second elevator is moved up due to the signal U_2=1. The sensors L_1 and L_2 indicate the current position of the elevators and we assume that both signals from the sensors are normalized to the interval [0,1]. When the first elevator reaches the level 0, (L_1=1), it goes to the waiting state W_1, which lasts for the time T_1. When the second elevator reaches the level II (L_2=1), it goes to the waiting state W_2 for the time T_2. After the time T_1, the first elevator is moved up (U_1=1) as long as the signal from the sensor reaches the value L_1=0, (~L_1=1). Then the first elevator goes into the waiting state S_1. After the time T_2 the second elevator is moved down (D_2=1) as long as the condition L_2=0 will be satisfied, (~L_2=1). Next, the second elevator goes into the waiting state S_2. When both elevators reach the states S_1 and S_2, correspondingly, the whole control process can be repeated.

The fuzzy Petri net which models the control algorithm is shown in Fig. 2, where:

S_i - initial state of ith elevator, i=1,2,
W_1, W_2 - waiting states of the first and second elevator on levels 0 and II, respectively,
T - common waiting time of both elevators on level I,
T_1 - waiting time of the first elevator on the level 0,
T_2 - waiting time of the second elevator on the level II,
L_i - position sensor signals of ith elevator, i=1,2,
D_i - control signal for ith elevator to move "down", i=1, 2,
U_i - control signal for ith elevator to move "up", i=1, 2.

Note that FPN, which describes the control algorithm enables one to take into consideration the analog sensors. Due to this we can recognize both movement direction, and current position of both elevators. This will cause no additional complication of the net. The structure of both binary, and fuzzy Petri net remains the same (Fig. 2).

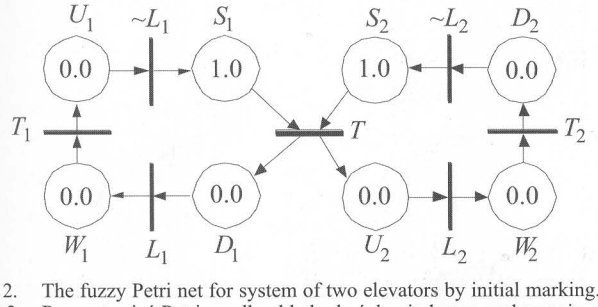


Fig. 2. The fuzzy Petri net for system of two elevators by initial marking.
Rys. 2. Rozmyta sieć Petriego dla układu dwóch wind przy znakowaniu początkowym.

One can describe this net by the incidence matrices C^+ and C^- as follows:

$$C^+ = \begin{array}{c} \\ T \\ L_1 \\ T_1 \\ \sim L_1 \\ L_2 \\ T_2 \\ \sim L_2 \end{array} \begin{array}{c} \begin{array}{cccccccc} S_1 & D_1 & W_1 & U_1 & S_2 & U_2 & W_2 & D_2 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array} \tag{18}$$

$$C^- = \begin{array}{c} \\ T \\ L_1 \\ T_1 \\ \sim L_1 \\ L_2 \\ T_2 \\ \sim L_2 \end{array} \begin{array}{c} \begin{array}{cccccccc} S_1 & D_1 & W_1 & U_1 & S_2 & U_2 & W_2 & D_2 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \end{array} \tag{19}$$

Assume that the common waiting time T has expired and the elevators started from the level I. The vector M which defines the current marking of the net is given by:

$$M = \begin{array}{c} \begin{array}{cccccccc} S_1 & D_1 & W_1 & U_1 & S_2 & U_2 & W_2 & D_2 \end{array} \\ \begin{bmatrix} 0.0 & 1.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 & 0.0 \end{bmatrix} \end{array} \tag{20}$$

In this state the transitions associated with the sensors L_1 and L_2 are enabled. This means that the vector E is equal to:

$$E = \begin{array}{c} \begin{array}{ccccccc} T & L_1 & T_1 \sim L_1 & L_2 & T_2 \sim L_2 \end{array} \\ \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \end{array} \tag{21}$$

Let us suppose that the first elevator covered 20% of its distance, (L_1=0.2), whereas the second one - 70% of its distance, (L_2=0.7). In such case we obtain:

$$\Theta = \begin{bmatrix} 0.0 & 0.2 & 0.0 & 0.8 & 0.7 & 0.0 & 0.3 \end{bmatrix} \tag{22}$$

The marking of the net will be changed as follows:

$$M' = \begin{array}{c} \begin{array}{cccccccc} S_1 & D_1 & W_1 & U_1 & S_2 & U_2 & W_2 & D_2 \end{array} \\ \begin{bmatrix} 0.0 & 0.8 & 0.2 & 0.0 & 0.0 & 0.3 & 0.7 & 0.0 \end{bmatrix} \end{array} \tag{23}$$

There was partial displacement of the marker between places D_1 and W_1 and between U_2 and W_2 (see Fig. 3). The displacement of the marker between D_1 and W_1 is proportional to the distance covered by the first elevator, whereas the displacement between U_2 and W_2 - to the distance covered by the second elevator. The marking changes seem to be rather understandable. It is worth noting that the arithmetic sum in the places D_1 and W_1 and in the places U_2 and W_2 is equal to 1. This general feature of the net was discussed in Section 4 and now it should be clear, why we say about transferring of the fuzzy marker through the transitions.

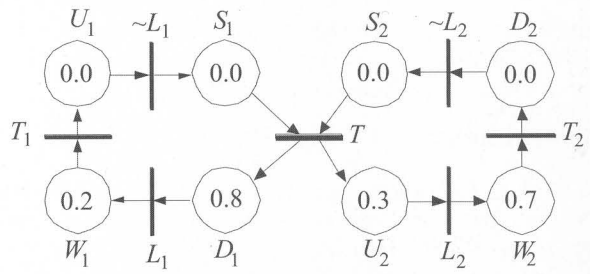


Fig. 3. The next marking of FPN from Fig. 2.
Rys. 3. Znakowanie następne dla FPN z Rys. 2.

The net can describe with any desired accuracy, the movement of the elevators, without necessity of introducing new places or transitions. Moreover, the structure of the binary net is preserved.

6. Summary

In the paper a fuzzy Petri net, which has a natural interpretation from fuzzy logic point of view, if it is a formal description of a control algorithm, was presented. The fuzzy logic was viewed rather as a kind of multi-valued logic. The conditions for transitions, which may be fired, were formulated and the method of transferring fuzzy markers was described. The algebraic method of the net description in the matrix form and the procedure of calculating the next marking based on current marking were presented.

The synthesis method of such a net is dedicated to rather complicated processes, which can be decomposed to a number of parallelly operating subprocesses and modelled using Petri nets. The advantage of using such class of Petri nets is the possibility for control systems design, in which both binary, and multi-valued, i.e. analog signals, occur. Thus, the controlled processes can have both analog, and binary inputs (from the sensors), and outputs (as control signals).

The net described in this paper can be implemented as a software for programmable logic controllers. The method of such implementation was described in [15], where special kind of fuzzy JK flip-flops developed in [12], and other fuzzy hardware components were used. The software for the FPN is much more complicated than this one for binary Petri nets.

The FPN can be assembled as a hardware device using existing [14] or new fuzzy hardware components [12, 13, 16, 17]. Such approach seems to be the most promising in the future, because it leads to low cost and very high-speed controller, which can work as synchronous or asynchronous hardware device [16, 17]. Some methods concerning the design process of such devices using FPGAs were developed and described in other authors' works. New methods concerning rapid prototyping of the nets as hardware devices are being currently investigated.

The proposed Petri net is robust. Namely, changes of the signal determining the degree ϑ to which the condition corresponding to the transition t is satisfied, are not able to change the direction of transferring the marker through this transition. Owing to this, the control system based on FPN is robust with respect to temporary absence of the signals from the sensors. This fact was proved experimentally on several laboratory plants (a system of vehicles, a concrete production process and a technological line) [9, 13, 15, 16, 17].

If we change the analog sensors to binary ones, the FPN-based controller behaves exactly as the one based on binary Petri net - no additional changes in hardware or software are needed. The justification of this fact is simple: all fuzzy hardware or software components work both in the case of multi-valued signals from the interval [0,1], and in the case of binary ones from the set $\{0,1\}$. It is natural, because Boolean logic may be viewed as a special case of fuzzy logic.

In contrast to binary Petri net, owing to fuzziness introduced to our net, we can apply more sophisticated control algorithms. For example, the signals generated by the FPN as a controller, can depend on the fuzzy marking of the net.

7. References

[1] D. Andreu, J. C. Pascal, and R. Valette: Fuzzy Petri Net-Based Programmable Logic Controller, IEEE Trans. Syst., Man, Cybern., vol. 27, pp. 952-961, Dec. 1997.

[2] A. J. Bugarin and S. Barro: Fuzzy Reasoning Supported by Petri Nets, IEEE Trans. Fuzzy Syst., vol. 2, no. 2, pp. 135-150, 1994.

[3] J. Cardoso and H. Camargo (Eds.): Fuzziness in Petrii Nets, Studies in Fuzziness and Soft Computing, vol. 22, New York: Physica-Verlag, 1999.

[4] N. Chang, W. H. Kwon, and J. Park: Hardware implementation of real-time Petri-net-based controllers, Control Eng. Practice, vol. 6, pp. 889-895, 1998.

[5] S. M. Chen: Fuzzy Backward Reasoning Using Fuzzy Petri Nets, IEEE Trans. Syst., Man, Cybern., vol. 30, no. 6, pp. 846-856, Dec. 2000.

[6] S. M. Chen, J. S. Ke, and J. F. Chang: Knowledge Representation Using Fuzzy Petri Nets, IEEE Trans. Knowledge Data Eng., vol. 2, no. 3, pp. 311-319, Sept. 1990.

[7] R. David and H. Alla: Petri Nets and Grafcet: Tools for Modelling Discrete Event Systems. London: Prentice Hall, 1992.

[8] R. David and H. Alla: Petri Nets for Modeling of Dynamic Systems - A Survey, Automatica, vol. 30, no. 2, pp. 175-202, 1994.

[9] G. Dec, Z. Hajduk, D. Zakorczmenny: Model do sterowania linią produkcyjną procesu kucia na gorąco, PAK, Nr 9, s.25-28, 2004.

[10] J. M. Fernandes, M. Adamski, and A. J. Proenca: VHDL Generation from Hierarchical Petri Net Specifications of Parallel Controllers, IEE Proc.-Comput. Digit. Tech., vol. 144, no. 2, pp. 127-137, 1997.

[11] L. Ferrarini and L. Piroddi: Modular design and implementation of a logic control system for a batch process, Computers & Chemical Engineering, vol. 27, pp. 983-996, 2003.

[12] L. Gniewek and J. Kluska: Family of Fuzzy J-K Flip-Flops Based on Bounded Product, Bounded Sum and Complementation, IEEE Trans. Syst., Man, Cybern., Part B, vol. 28, pp. 861-868, Dec. 1998.

[13] L. Gniewek and J. Kluska: Hardware implementation of Fuzzy Petri Net as a controller, IEEE Trans. Syst., Man, Cybern., Part B, vol. 34, pp. 1315-1324, June 2004.

[14] A. Kandel and G. Langholtz (Eds.): Fuzzy Hardware: Architectures and Applications. Norwell, MA: Kluwer 1998.

[15] J. Kluska and L. Gniewek: A New Method of Fuzzy Petri Net Synthesis and its Application for Control Systems Design, in Fuzzy Control, Advances in Soft Computing, R. Hampel, M. Wagenknecht, and N. Chaker, Eds. Heidelberg: Springer-Verlag, 2000, pp. 222-227.

[16] J. Kluska and Z. Hajduk: Hardware implementation of a fuzzy Petri net based on VLSI digital circuits. Proc. of 3rd EUSFLAT Conf. Zittau, pp. 789-793, 2003.

[17] J. Kluska and Z. Hajduk: Digital implementation of fuzzy Petri net based on asynchronous fuzzy RS flip-flop. Proc. of 7th Int. Conf. Artificial Intelligence and Soft Computing ICAISC, Zakopane, pp. 314-319, 2004.

[18] C. G. Looney: Fuzzy Petri Nets for Rule-Based Decisionmaking, IEEE Trans. Syst., Man, Cybern., vol. 18, no. 1, pp. 178-183, Jan./Feb. 1988.

[19] P. Misiurewicz: Zagadnienia projektowania cyfrowych układów sterowania binarnego, VIII Kraj. Konf. Automatyki, tom 1, s. 664-670, Szczecin, 1980.

[20] T. Murata: Petri Nets - Properties, Analysis and Applications, Proc. IEEE, vol. 77, no. 4, pp. 541-580, Apr. 1989.

[21] G. K. H. Pang, R. Tang, and S. Woo: A process-control and Diagnostic Tool based on Continuous Fuzzy Petri Nets, Eng. Appl. Artif. Intellig., vol. 8, no. 6, pp. 643-650, 1995.

[22] W. Pedrycz and F. Gomide: A Generalized Fuzzy Petri Net Model, IEEE Trans. Fuzzy Syst., vol. 2, no. 4, pp. 295-301, Nov. 1994.

[23] E. Park, D. M. Tilbury, and P. P. Khargonekar: A Modeling and Analysis Methodology for Modular Logic Controllers of Machining Systems Using Petri Net Formalism, IEEE Trans. Syst., Man, Cybern., Part C, vol. 31, no. 2, pp. 168-188, May 2001.

[24] J. L. Peterson: Petri Net Theory and the Modelling of Systems, Prentice Hall, Englewood Cliffs, NJ, 1981.

[25] W. Reisig: Petri nets: an introduction. Berlin: Springer-Verlag, 1985.

[26] P. H. Starke: Petri Netze: Grundlagen, Anwendungen, Theorie. Berlin: VEB Verlag, 1980.

[27] L. A. Zadeh: Fuzzy sets, Information and Control, no. 8, pp. 338-353, 1965.

Tytuł: Rozmyte sieci Petriego jako układy sterowania

Artykuł recenzowany