Fifth IFSA World Congress (1993), 1330-1333

FUZZY PETRI NETS AND THEIR APPLICATIONS TO FUZZY REASONING SYSTEMS CONTROL

Tadashi MATSUMOTO, Atsushi SAKAGUCHI, and Kohkichi TSUJI
Department of Electrical and Electronics Engineering
Faculty of Engineering, Fukui University, Fukui 910, JAPAN

ABSTRACT

In this paper, first, the fuzzy Petri net inference mechanism with learning function is proposed by using the extended fuzzy Petri nets. Secondly, a control system with this new inference engine is proposed. This system can do automatically and easily the knowledge acquisition from the operator's empirical data and can also be controlled adaptively under the big parameter change.

1 INTRODUCTION

A fuzzy control has a distinguished feature that the control is capable of incorporating expert's emperical control rules using his linguistic description. [4] ~ [11] But, automated extraction or identification of the fuzzy if - then rules is one of big issues in a fuzzy control. Several researchers have done on the identification capability as well as the learning capability of neural networks for their applications to fuzzy control. [7] ~ [10]

However, another model for a fuzzy control has been desired because the neural network model has the linkage explosion and the computation of exponentials for each linkage of neurons. [4], [5]

In this paper, first, the fuzzy Petri net inference mechanism with learning function is proposed by using the extended fuzzy Petri nets. The new controller can automatically identify the if - then rules and tune the membership functions by utilizing expert's control data. Identification capability of the new fuzzy controller is also examined using numerical data. Secondly, an adaptive control system with this new inference engine is proposed. It is shown that this system can be controlled adaptively under the big parameter change.

2 FUZZY PETRI NETS AND FUZZY INFFRENCE MECHANISM

2.1 Definitions and Notations for the Extended Fuzzy Petri nets[1]~[5].

First, let us define the ordinary Petri nets[1].

[Definition 1] Ordinary Petri Nets[1],[2]

The structure of an ordinary Petri net can be defined as a directed bipartite graph with two disjoint sets of nodes, S and T, called a set S of places (symbol:○) and a set T of transitions (symbol:|). Assume that cardinality of S is n. Marking $M \in N^n$ is defined as a non-negative integral vector whose component $M(p)$ is a number of tokens on the place p. For a subset U of S, the symbol ${}^\bullet U$ denotes the set of all transitions t's such that there exists an arc from t to $p \in U$. The symbol ${}^\bullet U$ is called the set of input transitions of U. Similarly, $U^\bullet$ denotes the set of all transitions t's such that there exists an arc from $p \in U$ to t and is called the set of output transitions of U. For subset Q of T, the set of input places ${}^\bullet Q$ and the set of output places $Q^\bullet$ of Q are similarly defined. For a transition t, t is said to be firable at M iff $M(p) > 0$ for each $p \in {}^\bullet t$. When the firable transition t at M fires, the tokens are moved as follows and the new resulting marking $M' \geq 0$ is defined as

$$\begin{aligned} M'(p) &= M(p) + 1 : p \in t^\bullet \cap p \notin {}^\bullet t \\ M'(p) &= M(p) - 1 : p \in {}^\bullet t \cap p \notin t^\bullet \\ M'(p) &= M(p) : \text{otherwise.} \end{aligned} \quad (1)$$

Next, we review the new extended fuzzy Petri nats[4][5]. The definition of ordinary Petri nets is included in that of the extended fuzzy Petri nets. For modeling of vague discrete events, let us define the extended fuzzy Petri nets by changing the definition of ordinary Petri nets.

[Definition 2] Extended fuzzy Petri nets (EFPNs) [5]

• Place: The same definition and symbol of a place as those in ordinary Petri nets are used.

• Transition: Symbol | :Free firing transition, but this firing is selective.

• Token: Let the symbol $\bullet^W$ be the token with the weight w which is any positive real number. Let us define the virtual token when W equals zero and denote it $\bigcirc = \bullet^0$. The virtual token is useful to distinguish the following two cases:

① The premise of if-then rule is denied, then the place has no token.

② The if-then rule has not worked at that time, then the place has no token.

The virtual token implies the case ①.

• Firing rule: By a firing of firable transition t at M, the token is moved and the new resulted marking $M' \geq 0$ is defined as follows, where $a = \min_i M(p_i)$ and $p_i \in {}^\bullet t$.

$$\begin{aligned} M'(p) &= M(p) + a : p \in t^\bullet \cap p \notin {}^\bullet t, \\ M'(p) &= M(p) - a : p \in {}^\bullet t \cap p \notin t^\bullet, \\ M'(p) &= M(p) : \text{otherwise.} \end{aligned} \quad (2)$$

• Arc: There exists no weight on it. See also Remark 1. ∎

[Definition 3] The max.transition

Let the max.transition be the transition t which transfers the maximum token-weight of all the input places p's $\in {}^\bullet t$ into all the output places p''s $\in t^\bullet$. ∎

However, if we use the virtual tokens, some inhibitors ($-\circ$), and the regular transitions which obey the firing rule of Definition 2, we can represent the max transition by using EFPNs.

[Remark 1]

Although it is defined that an arc has no weight in the extended fuzzy Petri nets as in Definition 2, it is simple to use the weight on the arc (t, p) as shown in §2 · 2. However, the weight on the arc is transformed into the net of Difinition2 because of the firing rule of the regular transitions. ∎

Hereafter, let us denote the extented fuzzy Petri

nets EFPNs for simiplicity.

2.2 Fuzzy Inference Mechanism using EFPNs[5]

Let us consider the if-then rule; If X is A, then Y is C, where X and Y are fuzzy variables. First,let us explain how to model the discrete membership functions of both the premise and the consequence by using examples.

If X has the discrete membership functions of Fig.1(a), then the EFPN model for Fig.1(a) is obtained as in Fig.1(b), where "¶" implies "↑↓", i.e., the double arc. When the input for X is 1.0, then the weights for P and Z is 0.5 and 0.3, respectivey.

If $Y = C$ has the discrete membership function of Fig.2(a) after Mamdani's min-max opetrations, then we have the EFPN model of Fig.2(b). When the resulted weights for P, Z, and N is 0.8, 0.2, and 0.0, we have the value of 0.8, 0.5, 0.2, 0.2, and 0.0 on the discrete universe 2, 1, 0, -1, and -2, respectively.

Secondly, let us consider the fuzzy inference mechanism of two input variables, X_1 and X_2, and one output variable, Y, by using Mamdani's method, where each variable has three lingistic labels, P, Z, and N and the decision rules are shown in Fig. 6(a). Then, we have nine if-then rules.

If X_1 is A_i and X_2 is B_i, then Y is C_i, $i = 1, 2, \cdots, 9$. Moreover, the resultant fuzzy number from the above each rule is determined by the max. operation (i.e., the max. transition).

Therefore, we have the EFPN model shown in Fig.3(b) for the above fuzzy inference mechanism.

If we adopt gravity calculation as a defuzzification method, the output Y^* of the fuzzy inference mechanism is determined by $Y^* = \int YC^*(Y)dy / \int C^*(Y)dy$, where $C^*(Y) = C_1(Y) \vee C_2(Y) \vee \cdots \vee C_9(Y)$ and $\vee$ denotes maximum. Note that we can use the resultant fuzzy numbers, P, Z,and N, of Fig.3(b) for the above defuzzification.

From Fig.3, we can easily understand each reasoning process and can do easily and effectively the knowledge acquisition because the extended fuzzy Petri nets EFPNs avoid the linkage explosion of the usual neural networks by linking only conditions that combine into rules, whereas neural networks link every neuron cell in a layer with every other neuron in adjacent layers. Further, the outputs of EFPN transitions are extermely simple and efficient to compute, while the output functions of the usual neural networks involve the computation of exponentials for each linkage of neurons[4],[5].

3 AUTOMATED EXTRACTION OF FUZZY IF-THEN RULES

3.1 Automated Rule-Extraction Method

The automated extraction or identification of the fuzzy if-then rules is one of big issues in a fuzzy control or a fuzzy inference engine.

In this subsection, let us show an automated rule-extraction method for the EFPN modelled fuzzy inference engine.

(1) The setting of the membership functions for the premise of a rule;

The membership functions for the premise of a rule is fixed such that each membership function has the equi-width and that it is regularly spaced on the universe, where each fuzzy gain is adjusted to cover all the input data.

(2) Initialization of the consequence of a rule;

All the membership functions of the consquence are set zero at the biginning.

(3) Automated extraction of the membership functions for the consequence of a rule:

The consequence of an if-then rule is characterized by the weights of arcs from the transitions, of which input places imply the resultant conditions of each rule, to the places which imply the discrete points on the universe. Then, we can generate the membership functions of the consequence adjusting the above weights to the expert's input data. Let X_s and Y_s be the input-output pair at time t and let the token weight on the place which implies the adaptability of the premise of the i-th rule be $O_i(s)$. Then, the token weight of the consequence at time $t+1$, W_{ij}, is determined by the next equation;

$$W_{ij}(t+1) = W_{ij}(t) - (Y_s - Y^*) \cdot (Y^* - j) \cdot O_i(s) \cdot LF, \quad (3)$$

where j is the discrete value on the universe,$W_{ij}(0) = 0$,and LF =0.001 is the learning factor.

(4) The performance evaluation of a rule;

Measure the infered output to the expert's input data by using the new extracted rules and calculate the error between the infered output and the expert's output. If the error is too big, adjust the interval of the premise and extract again the membership functions of the consequence .

3.2 Performance Evaluation

In order to check the usefulness of the learning fuzzy inference engine, we use the following nonlinear system with three inputs (X_1, X_2, X_3) and an output (Y), which was used to verify the learning capability in Refs.[7],[8],[11];

$$Y = (1 + X_1^{0.5} + X_2^{-1} + X_3^{-1.5})^2 \quad (4)$$

Table 1 is the obtained results about evaluation, where X_4 is a dummy variable. In Table 1, E_1 is the average error for identification data (No.1∼No.20) and E_2 is the average error for evaluation data (No.21∼No.40). In our experiment, E_1 is very good compared with others [7], [8], [11], but E_2 is not so good. As a whole, we can say that our fuzzy engine has the same identification capability as that in [7], [8], [11].

4 SIMULATION

In this section, the simulated results for the controlled object of the first-order lagging system with dead time are given. The control system used is Fig.6 without the output trajectry estimator, where the learning fuzzy controller has two inputs $(e, \Delta e)$ and an output (Δu) with seven premise membership functions.

We adopted the input and output data of PI control as the expert's data for automated extraction of rules, where $K_p = 4.2$, $K_I = 0.12$, and the sampling period $\tau = 0.2$. The number of input and output data for identification is 250 from $t = 0.0$ to $t = 50.0$ (sec). Moreover, the fuzzy gains are $g_e = 0.02$, $g_{\Delta e} = 0.5$, and $g_{\Delta u} = 1.7$, where $e, \Delta e$, and Δu is the error, the change of error, and the change of manipulated variable, respectively.

Table 2 shows the obtained fuzzy rule table in which the number implies each membership function. In this simulation, 35 rules out of 49 are automatically generated.

One of controlled results under the automatically extracted rules is shown in Fig.4 together with PI control, which are step responses of a feedback control system which incorprates the above fuzzy engine and a controlled object of a first order lagging system with a dead time : $G(s) = \frac{e^{-2s}}{1+20s}$.

5 CONTROL SYSTEM WITH FUZZY PETRI NET MODEL INFERECE ENGINE

Let us consider the adaptive modification of if-then rules to overcome the system paramater change or the disturbance by using the knowledge about the output trajectory which implies a kind of the higher rank knowledges than the expert's operating knowledges. This problem is another important issue in a fuzzy control[6]. Fig.6 is the blockdiagram of the fuzzy control system, where the Fuzzy controller in Section 4 is used and the output trajectory estimation and the performance evaluation are as follows.

5.1 Output Trajectory Estimator

The fuzzy if-then rule for the output trajectory estimator has two inputs (e and Δe) and an output (the change of error at the next time ; Δne) and the range of the universe of both the premise and the consequence is from -9 to +9 and it is discretized as the integer. The number of rules is $9 \times 9 = 81$. The estimation process is as follows.

(1) Collect the operator's input and output data and determine each fuzzy gain for e, Δe, and Δne.

(2) The initial learning or identification;

Extract the knowledge for each fuzzy if-then rule for the output trajectry estimator from the operator's input and output data by using the same automated rule-extraction method as that in Section 3.

(3) On line adaptation;

Each consequence of rules of the fuzzy controller is modified such that $\Delta u(t_2-l)$ equals $D_{\Delta}u(t_2-l)$ by using the method of §3.1 three times and the next equation;

$$D\_\Delta u(t_2-l) = \frac{D\_\Delta e(t_2) - \Delta e(t_1)}{\Delta e(t_2) - \Delta e(t_1)} \cdot \Delta u(t_2-l), \qquad (5)$$

where $\Delta e(t_1)$, $\Delta e(t_2)$, and $D\_\Delta e(t_2)$ are defined in Fig.5 and $\Delta u(t_2-l)$ is the change of manipulating variable at time $t_2 - l$.

Each fuzzy gain for e, Δe, and Δne is determined as follows; $g_e = 0.1$, $g_{\Delta e} = 2.0$, and $\Delta ne = 40.0$.

The idenification of the 9×9 rules for the output trajectory estimator is done as the same way as that in Section 4 and 48 rules out of 81 are automatically generated, where the maximum learning repetitions is 2000 times.

5.2 Simulation for Fuzzy Control System

In order to check the usefulness for the system parameter change in Fig.6, we simulate the next case, where the parameter is changed at time $t = 20$(sec).

$G(s) = \frac{e^{-2s}}{1+20s} \rightarrow G(s) = \frac{e^{-2s}}{1+10s}$

The simulated result is shown in Fig.7 together with PI control results.

Fig.8 is the simulated result with the disturbance $d = 80$ at time $t = 25$(sec), where $G = \frac{e^{-2s}}{1+20s}$. We could recognize the usefulness of adaptation for the disturbance of being less than or equal to 80, but for more than $d > 80$, it was difficult to control the disturbance because the knowledge of the output trajectory estimator was not enough to control those bigger disturbances. If we identify directly the knowledge of the estimator from the desired output trajectory instead of the operator's input and output data, it seems that we can improve the adaptivity for bigger disturbance.

6 CONCLUSIONS

In this paper, we proposed a learning fuzzy inference mechanism, or a fuzzy controller, by using the extented fuzzy Petri nets. This fuzzy engine could acquisite easily and effectively the knowledge for each if-then rule from the expert's operating input and output data.

For demonstrating the capability of the new fuzzy engine, we also simulated step responses of a feedback control system which incorporated the fuzzy engine and a controlled object of a first order lagging system with a dead time. Moreover, we applied this new engine to a control system with parameter change and disterbance. Then, we comfirmed that the system could be controlled adaptively under rather big parameter change and disterbance.

Therefore we can conclude that the fuzzy Petri net inference engine has some superiorities to neural networks; the net structure as well as computation is simpler and each reasoning process can be easily understood. Then the knowledge acquisition can also be done easily and effectively.

References

[1] J.L.Peterson : *"Petri Net Theory and the Modelling of Systems"*, Prentice-Hall(1981)

[2] T.Murata : "Petri nets : properties, analysis and applications", Proc.of the IEEE, Vol.77, No.4, pp.541-580 (1989).

[3] K.Jensen : "Coloured Petri Nets and the Invariant-Method", Theoret.Comput. Sci., Vol.14,pp.317-336 (1985).

[4] C.G.Looney : "Fuzzy Petri Nets for Rule Based Decisionmaking", IEEE Trans.on Systems,Man,and Cibernetics,Vol.18,pp.178-183 (1988).

[5] T.Tsuji and T.Matsumoto : "Extended Petri Net Models for Neural Networks and Fuzzy Inference Engines",Proc. of ISCAS'90,pp.2670-2673 (1990).

[6] T.Terano,K.Asai,and M.Sugeno : *"Fuzzy Systems Theory and Its Applications"*, Ohm-Sha(1987).

[7] I.Hayashi and H.Takagi : "Formulation of Fuzzy Reasoning by Neural Network",Proc.of 4th Fuzzy System Symposium, pp.55-60,Tokyo,May30-31(1988).

[8] S.Horikawa,T.Fukuhashi, S.Okuma, and Y.Uchikawa : "A Learing Fuzzy Controller Using a Neural Network", Proc. of SICE, Vol.27, No2, pp2, pp.208-215(1991).

[9] Y.Hayashi and M.Nakai : "Automated Extraction of Fuzzy IF-THEN Rules Using Neural Networks", Trans. IEE Japan, Vol.110-c, No.3, pp.198-206(1990).

[10] T.Yamaguchi,N.Imazaki, and K.Haruki : "A Reasoning and Learining Method for Fuzzy Rules with Associative Memory", ibid., pp.207-214(1990).

[11] G.T.Kang and M.Sugeno : "Fuzzy Modelling", Proc.of SICE, Vol.26, No.6,pp.106-108(1987).

[12] Y.Deng and S.-K Chang : "A G-Net Model for Knowledge Representation and Reasoning", IEEE Trans. Knowledge and Data Engeneering, Vol.2,3,pp.295-310(1990).

[13] S.-M.Chen,J.-S Ke and J.-F Chang : "Knowledge Representation Using Fuzzy Petri Nets", ibid., pp.313-319(1990).

[14] T.Matsumoto,A.Sakaguchi and K.Tsuji : "Automated Extraction of Fuzzy If-Then Rules Using Fuzzy Petri Nets and its Application to Adaptive System Control", Procs. of Korea-Japan Joint Conf. on Fuzzy Systems and Engeneering, pp.37-40(1992).

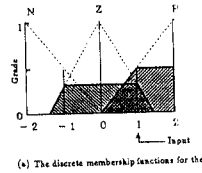
[15] M.-G Chun and Z.Bien : "Expert's Fuzzy Knowledge Representation and Inference Methods via Fuzzy Petri Nets", ibid., pp.64-67(1992).

Table 1 Precision of identified model

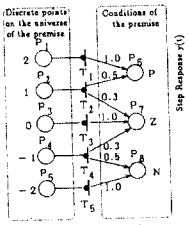
	Input variables	E_1 (%)	E_2 (%)
Fuzzy Controller using EFPN	X_1, X_2, X_3, X_4	$0.11*10^{-3}$	19.13
	X_1, X_2, X_3	$0.38*10^{-3}$	5.41
Fuzzy Controller using neural network[8]	X_1, X_2, X_3, X_4	0.07	14.69
	X_1, X_2, X_3	0.60	2.41
Fuzzy Controller using neural network[7]	X_1, X_2, X_3	0.47	4.79
Sugeno's fuzzy model *I* [13]	X_1, X_2, X_3	1.5	2.1
Sugeno's fuzzy model *II* [13]	X_1, X_2, X_3	1.1	3.6

Table 2 Identified control rules

Δe \ e	PB	PM	PS	ZO	NS	NM	NB
PB	—	—	—	—	—	—	—
PM	—	⑥	⑫	⑱	㉔	㉚	—
PS	①	⑦	⑬	⑲	㉕	㉛	—
ZO	②	⑧	⑭	⑳	㉖	㉜	—
NS	③	⑨	⑮	㉑	㉗	㉝	—
NM	④	⑩	⑯	㉒	㉘	㉞	—
NB	⑤	⑪	⑰	㉓	㉙	㉟	—

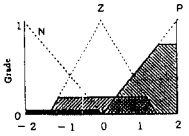


(a) The discrete membership functions for the premise.

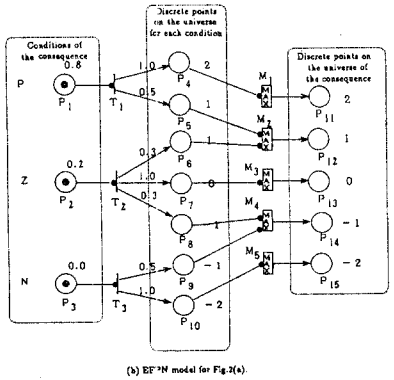


(b) EFPN model for Fig.1(a).

Fig.1 EFPN model for the discrete membership functions for the premise.

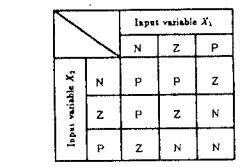


(a) The resultant discrete membership functions of the consequence

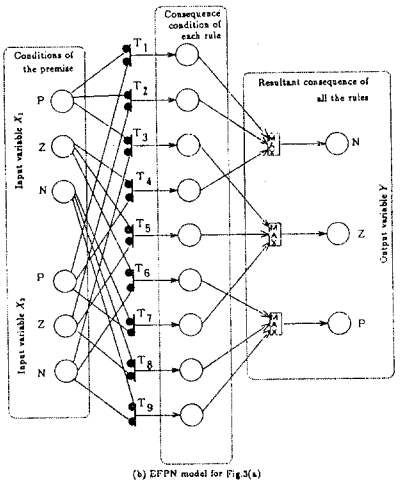


(b) EFPN model for Fig.2(a).

Fig.2 EFPN model for the discrete membership functions for the consequence.



(a) Fuzzy rule table with two inputs and an output.



(b) EFPN model for Fig.3(a)

Fig.3 EFPN model fuzzy inference engine; each fuzzy variable has three lingistic labels.

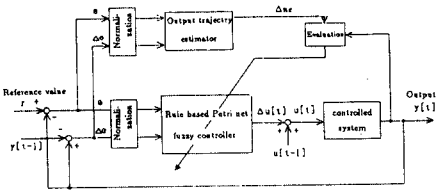
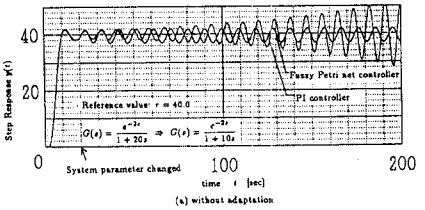
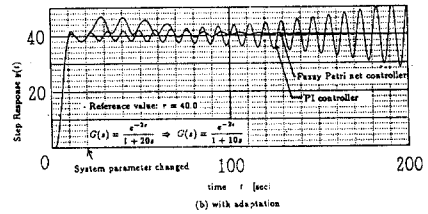


Fig. 6 System configuration of an adaptive control system



(a) without adaptation



(b) with adaptation

Fig. 7 Experimented results with PI controller or fuzzy Petri net controller.

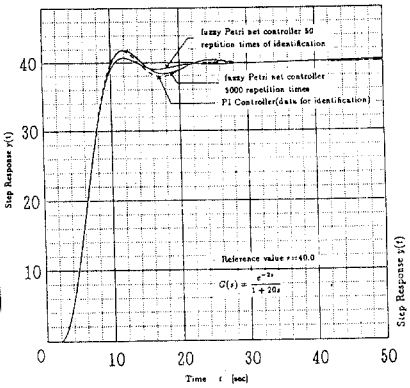


Fig. 4 Step responses of a control system with PI controller or fuzzy Petri net controller

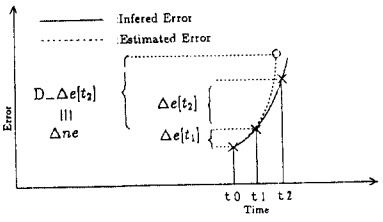


Fig. 5 Illustration of eq.(7).

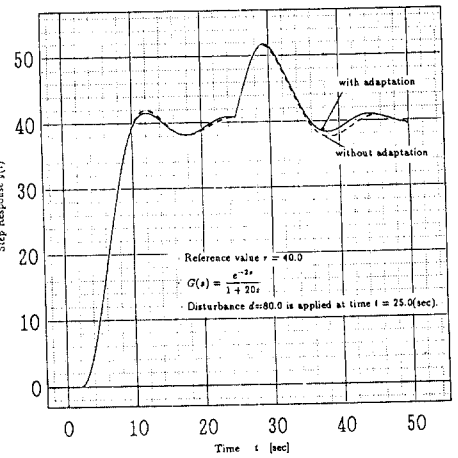


Fig. 8 Step responses of a control system with or without fuzzy adaptation under disterbance