15th Triennial World Congress, Barcelona, Spain

ADAPTIVE FUZZY PETRI NETS FOR SUPERVISORY HYBRID SYSTEMS MODELING

Xiaoou Li * **Wen Yu** ** **Sergio Perez** **

* *Sección de Computación, Departamento de Ingeniería Eléctrica CINVESTAV-IPN, Av.IPN #2508 Mexico D.F., 07360, Mexico*

** *Departamento de Control Automático CINVESTAV-IPN, Av.IPN #2508 Mexico D.F., 07360, Mexico e-mail: yuw@ctrl.cinvestav.mx, fax: +52-5-7477089*

Abstract: A supervisory hybrid system may be modeled from two levels: logic level (upper) and continuous level (lower). In this paper, adaptive fuzzy Petri nets and neural networks are combined together for supervisory hybrid system modeling. Adaptive Fuzzy Petri Net is adopted to model the supervisory logic parts, and dynamic neural networks are applied to continuous parts. Two hybrid system examples are illustrated to show the effective of the method

Keywords: supervisory hybrid system, fuzzy Petri net, neural networks

1. INTRODUCTION

One of the most recent and the most intense efforts in control theory deals with handing hybrid dynamic systems that include not only the continuous process but its supervisory mechanism as well. Hybrid systems have been intensively studied in the past few years both for their mathematical foundations (Lygeros *el al.*, 1999) and engineering design (Koo and Sastry, 1998).

For our purposes, supervisory hybrid systems are considered to be the combination of continuous plants and discrete-event plants. For modeling discrete event plants, Petri nets are widely adopted. Petri nets have been developed original to represent and analyze in an easy way the information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, non-deterministic, and/or stochastic (David and Alla, 1994). PNs have an inherent quality in representing logic in intuitive and visual way. Furthermore, PN approach can be easily combined with other techniques and theories such as programming, fuzzy theory, neural networks, etc. These modified PNs are widely used in computer, manufacturing, robotic, knowledge based systems, process control, as well as other kinds of engineering applications. For example, the reasoning path of expert systems can be reduced to simple sprouting trees if fuzzy Petri nets(FPN)-based algorithms are applied as an inference engine.(Chen *el al.*, 1990), (Yeung and Tsang, 1998).

On the other hand, many results show that neural network techniques seem to be very effective to identify a wide class of continuous plants when we have no complete model information, or even when we consider the controlled plant as a black box (Hunt *el al.*, 1992).

In this paper, hybrid systems can be modeled from two aspects: logic level (supervisory) and continuous process. For example, a chemical process with process-related logic, the logic part can be modeled by fuzzy Petri net and the continuous part may be modeled by neural networks. Two examples will show the effective of our research.

2. MODELING PRODUCTION RULES WITH ADAPTIVE FUZZY PETRI NET

In order to properly describe the real world, fuzzy production rules have been used for knowledge representation (Chen *el al.*, 1990). A fuzzy production rule (FPR) is a rule which describes the fuzzy relation between two propositions. When the antecedent portion of a fuzzy production rule contains AND or OR connectors, a composite fuzzy production rule is introduced. If the relative degree of importance of each antecedents contributing to the consequent is considered, weighted fuzzy production rules (WFPR) may be introduced (Yeung and Tsang, 1998). So, WFPR is the most general case of production rules. Without loss of generality, we use WFPRs to represent logic information of a hybrid system.

Let R be a set of weighted fuzzy production rules $R = \{R_1, R_2, \cdots R_n\}$. The general formulation of the ith weighted fuzzy production rule is as follows:

$$R_i: \text{ IF } a \text{ THEN } c\ (CF = \mu),\ Th,\ w$$

where $a =< a_1, a_2, \cdots, a_n >$ is the antecedent portion which comprises of one or more propositions connected by either AND or OR, c is the consequent proposition, μ is the certainty factor of the rule, Th is the threshold, w is the weight. In general, WFPRs are categorized into three types which are defined as follows (Li *el al.*, 2000):

Type 1: A Simple Fuzzy Production Rule

R: IF a THEN c $(CF = \mu)$, λ, w

Type 2: A Composite Conjunctive Rule

R: IF a_1 AND a_2 AND $\cdots$ AND a_n THEN c $(CF = \mu)$, $\lambda, w_1, w_2, \cdots, w_n$

Type 3: A Composite Disjunctive Rule

R: IF a_1 OR a_2 OR $\cdots$ OR a_n THEN c $(CF = \mu)$, $\lambda_1, \lambda_2, \cdots, \lambda_n, w_1, w_2, \cdots, w_n$

A fuzzy Petri net with learning ability in (Li *el al.*, 2000), which is called adaptive fuzzy Petri net, is used here as the model of the logic information in a hybrid system. An adaptive fuzzy Petri net (AFPN) is a 9-tuple

$$AFPN = (P, T, D, I, O, \alpha, \beta, Th, W)$$

where

$P = \{p_1, p_2, \cdots, p_n\}$ denotes a set of places;

$T = \{t_1, t_2, \cdots, t_m\}$ is a set of transitions;

$D = \{d_1, d_2, \cdots, d_n\}$ is a set of propositions;

$I(O) : T \rightarrow P^{\infty}$ is the input (output) function which defines a mapping from transitions to bags of places;

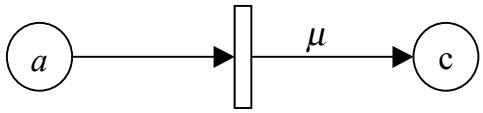

Fig. 1. AFPN of Type 1

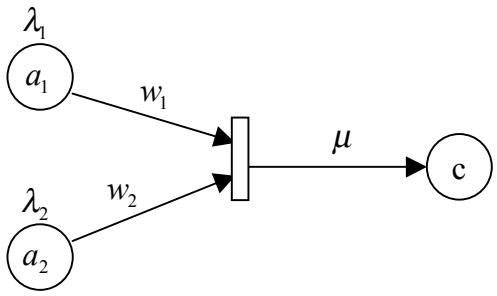

Fig. 2. AFPN of Type 2

$\alpha : P \rightarrow [0, 1]$ is an association function which assigns a real value between zero to one to each place;

$\beta : P \rightarrow D$ is a bijective mapping between the proposition and place label for each node. $P \cap T \cap D = \phi$, $|P| = |D|$;

$Th : T \rightarrow [0, 1]$ is the function which assigns a threshold value λ_i from zero to one to transition t_i.

$W = W_I \cup W_O$. $W_I : I \rightarrow [0, 1]$ and $W_O : O \rightarrow [0, 1]$, are sets of input weights and output weights which assign weights to all the arcs of a net.

The mappings of the three types of weighted fuzzy production rules into adaptive fuzzy Petri net are shown as Fig.1, Fig.2 and Fig.3 respectively. The three types of WFPR may be represented as follows:

Type 1: A Simple Fuzzy Production Rule

R: IF a THEN c $Th(t) = \lambda$, $W_O(t, p_j) = \mu$, $W_I(p_i, t) = w$

Type 2: A Composite Conjunctive Rule

R: IF a_1 AND a_2 AND $\cdots$ AND a_n THEN c, $Th(t) = \lambda$, $W_O(t, p_j) = \mu$, $W_I(p_i, t) = w_i$, $i = 1, \cdots, n$

Type 3: A Composite Disjunctive Rule

R: IF a_1 OR a_2 OR $\cdots$ OR a_n THEN c, $Th(t_i) = \lambda_i$, $W_O(t_i, p_j) = \mu$, $W_I(p_j, t_i) = w_i$, $i = 1, \cdots, n$

The mapping may be understood as each transition corresponds to a simple rule, composite conjunctive rule or a disjunctive branch of a composite disjunctive rule; each place corresponds to a proposition (antecedent or consequent).

Fuzzy Reasoning Algorithm

(1) Build the set of user input places P_{UI}.
(2) Build the set of initially enabled transitions T_p.

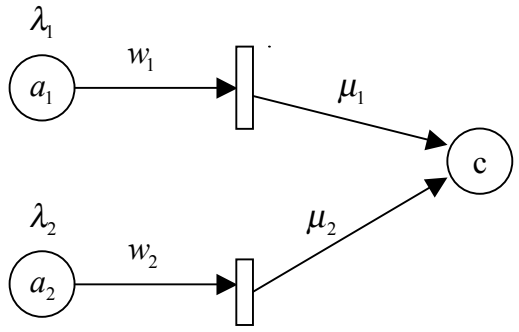

Fig. 3. AFPN of Type 3

(3) Find current enabled transitions $T_f = \{t \in T_p \mid for\ \forall p_i \in^{\cdot} t,\ \alpha(p_i) \geq \lambda_i\}$.
(4) Fire all current enabled transitions and calculate new certainty factors which are produced by fired transitions according to Definition 2.
(5) Make token transmission. Assume p is one of the output places of a fired transition
   - If $|^{\cdot}p| = 1$, then add a token to p with the certainty factor which is produced by its input transition;
   - If $|^{\cdot}p| > 1$ and more than one of its input transitions fired, then select the transition with the maximum output weight, and add a token to p with the certainty factor produced by this transition .
(6) Let $T = T - T_f$, $P = P -^{\cdot} T_f$
(7) Go to Step2 and repeat, until $T_f = \emptyset$

Before training an Adaptive Fuzzy Petri Net (AFPN), we should be clear what are the system output and input. Assume that a system is described by WFPRs', we define all the right hands of the rules as system output. If we have enough training data, the parameters can be adjusted well enough.

Given a WFPR R, we suppose that the thresholds of all its antecedent propositions $\lambda_1,\ \lambda_2, \cdots,\ \lambda_n$ and its certainty factor μ are known. However, we are not sure about the input weights $w_{I1},\ w_{I2},\ \cdots,\ w_{In}$. These weights can be learned in the situations which there are enabled transitions. For Type 1 WFPR's, the input weights are meaningless, only Type 2 and 3 WFPR's weights need to be studied.

We take Type 2 WFPR as an illustration. Type 2 FPR's can be translated into a standard neural network:

$$y(k) = W\ [(k)^T\ P(k) + b]$$

where k is time, input vector

$$P(k) = [\alpha(p_1)(k) - \lambda_1, \alpha(p_2)(k) - \lambda_2]^T$$

the weight vector $W(k) = [w_{I1}(k), w_{I2}(k)]^T$, bias $b = \min(\lambda_1, \lambda_2)$, the output $y(k)$ is the certainty factor of the consequence. So the Widrow-Hoff learning law (Least Mean Square) can be applied as

$$\begin{array}{c} W(k+1) = W(k) + 2\ \delta\ e(k)\ P(k) \\ e(k) = t(k) - y(k) \end{array} \quad (1)$$

where $t(k)$ is the goal output (teacher) and the weight vector $W(k)$ is calculated recursively. It is known that for a small enough positive constant δ, the updating law (1) converges to real values (Yu and Li, 2001).

For Type 3 WFPR, the learning law is similar. According to Definition 2, by comparing the weights of the input arcs of a place, it is not difficult to know which transition is the dominating one to it, so if the data are available, then training can be implemented.

3. MODELING CONTINUOUS DYNAMIC WITH NEURAL NETWORKS

Consider a dynamic process given by

$$\dot{x_t} = f(x_t, u_t) \quad (2)$$

where $x_t \in \Re^n$ is the state, $u_t \in \Re^m$ is the input vector . $f : \Re^n \times \Re^m \rightarrow \Re^n$ is locally Lipschitz. Let us consider the following dynamic neural network to identify the chemical process (2)

$$\dot{\widehat{x}_t} = Ax_t + W_{1,t}\sigma(\widehat{x}_t) + W_{2,t}\phi(\widehat{x}_t)u_t \quad (3)$$

here $\widehat{x}_t \in \Re^n$ is the state of the neural network, $A \in \Re^{n\times n}$ is a stable matrix. $W_{1,t} \in \Re^{n\times n}$ and $W_{2,t} \in \Re^{n\times n}$ are weight matrices. The vector field $\sigma(x_t) : \Re^n \rightarrow \Re^m$ is assumed to have the elements increasing monotonically. The function $\phi(\cdot)$ is the transformation from $\Re^n$ to $\Re^{m\times m}$. The typical presentation of the elements $\sigma_i(\cdot)$ and $\phi_{ij}(.)$ are as sigmoid functions

$$\sigma_i(\widehat{x}_{i,t}) = a_i / \left(1 + e^{-b_i\widehat{x}_{i,t}}\right) - c_i.$$

The structure of the dynamic neural networks (3) is shown in Fig.4. This kind of dynamic neural networks have been discussed by many authors, for example (Yu and Li, 2001), (Rovithakis and Christodoulou, 1994) and (Yu and Poznyak, 1999). Generally, dynamic neural network (3) cannot follow the nonlinear system (2) exactly, the nonlinear system may be written as

$$\dot{x_t} = Ax_t + W_1^*\sigma(x_t) + W_2^*\phi(x_t)\pi(u_t) - \widetilde{f}_t \quad (4)$$

where W_1^* and W_2^* are optimal matrix which may minimize the modelling error $\widetilde{f}_t$. They are bounded as

$$\begin{array}{c} W_1^*\Lambda_1^{-1}W_1^{*T} \leq \overline{W}_1, \\ W_2^*\Lambda_2^{-1}W_2^{*T} \leq \overline{W}_2 \end{array} \quad (5)$$

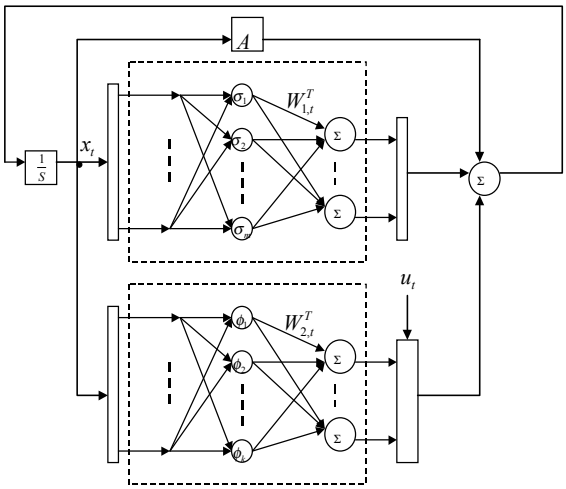

Fig. 4. The structure of the dynamic neural network.

Because $\sigma(\cdot)$ and $\phi(\cdot)$ are chosen as sigmoid functions, clearly they satisfy Lipschitz. condition,

$$\begin{array}{c}\widetilde{\sigma}_t^T\Lambda_1\widetilde{\sigma}_t \le \Delta_t^T\Lambda_\sigma\Delta_t, \\ \widetilde{\phi}_t{}^T\Lambda_2\widetilde{\phi}_t \le \overline{u}\Delta_t^T\Lambda_\phi\Delta_t\end{array} \tag{6}$$

where $\widetilde{\sigma}_t := \sigma(\widehat{x}_t) - \sigma(x_t)$, $\widetilde{\phi}_t := \phi(\widehat{x}_t)u_t - \phi(x_t)u_t$, Λ_1, Λ_2, Λ_σ and Λ_ϕ are positive define matrices. Let us define identification error as

$$\Delta_t := \widehat{x}_t - x_t$$

, its dynamic is obtained from (3) and (4)

$$\begin{array}{c}\dot{\Delta}_t = A\Delta_t + \widetilde{W}_{1,t}\sigma(\widehat{x}_t) + \\ \widetilde{W}_{2,t}\phi(\widehat{x}_t)u_t + W_1^*\widetilde{\sigma}_t + W_2^*\widetilde{\phi}_t + \widetilde{f}_t\end{array} \tag{7}$$

where $\widetilde{W}_{1,t} := W_{1,t} - W_1^*$, $\widetilde{W}_{2,t} := W_{2,t} - W_2^*$. Next Theorem states the learning procedure of neuro identifier.

Theorem 1. If the modelling error $\widetilde{f}_t$ is bounded ($\widetilde{f}_t^T\Lambda_f^{-1}\widetilde{f}_t \le \overline{\eta}$) and the weights $W_{1,t}$ and $W_{2,t}$ of the dynamic neural networks defined by (3) are updated as

$$\begin{array}{c}\dot{W}_{1,t} = -K_1\sigma(\widehat{x}_t)\Delta_t^T, \\ \dot{W}_{2,t} = -K_2\phi(\widehat{x}_t)u_t\Delta_t^T\end{array} \tag{8}$$

where K_1 and K_2 are positive matrices, then the identification error is bounded and satisfies

$$\limsup_{T\to\infty}\frac{1}{T}\int_0^T \|\Delta_t\|_{Q_0}^2\, dt \le \overline{\eta}, \quad Q_0 = Q_0^T > 0 \tag{9}$$

The proof may be found in (Yu and X. Li, 2001).

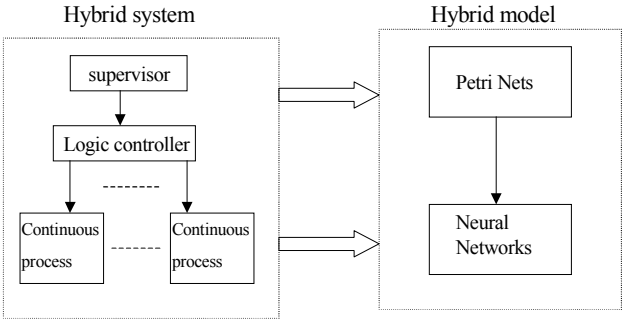

Fig. 5. Structure for hybrid modeling

4. MODELING SUPERVISORY HYBRID SYSTEM

The structure of the supervisory hybrid system and hybrid Petri net is showed in Fig.5.

We define a hybrid Petri net as

$$H = (AFPN, NN_\sigma)$$

where $\sigma \in M = \{1, 2, ..., m\}$ is the set of continuous processes, the multiple dynamic neural networks are

$$\dot{\widehat{x}}_t = A_\sigma\widehat{x}_t + W_{1,t}^\sigma\sigma_\sigma(\widehat{x}_t) + W_{2,t}^\sigma\phi_\sigma(\widehat{x}_t)u_t \tag{10}$$

(10) may be rewritten as following dynamic system

$$\dot{x} = f_\sigma(x, u), \quad x(0) = x_0 \tag{11}$$

Each proposition in AFPN may correspond to a neural network. So the three types of WFPR describing hybrid systems may be represented as follows (see Fig.6-Fig-8):

Type 1: A Simple Fuzzy Production Rule

R: IF a THEN process c_1 $Th(t) = \lambda$, $W_O(t, p_j) = \mu$, $W_I(p_i, t) = w$

Type 2: A Composite Conjunctive Rule

R: IF a_1 AND a_2 AND $\cdots$ AND a_n THEN process c_2, $Th(t) = \lambda$, $W_O(t, p_j) = \mu$, $W_I(p_i, t) = w_i$, $i = 1, \cdots, n$

Type 3: A Composite Disjunctive Rule

R: IF a_1 OR a_2 OR $\cdots$ OR a_n THEN process c_2, $Th(t_i) = \lambda_i$, $W_O(t_i, p_j) = \mu$, $W_I(p_j, t_i) = w_i$, $i = 1, \cdots, n$

5. CASES STUDY

Example 1. $\{A, B, C, D, E, F, G\}$ are related propositions of a knowledge based system Γ. Between them there exist the following rules

R1: If A and B Then E, $\lambda_A, \lambda_B, w_A, w_B$ (CF=μ_1)

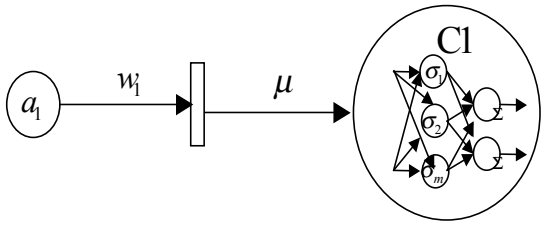

Fig. 6. Hybrid model 1

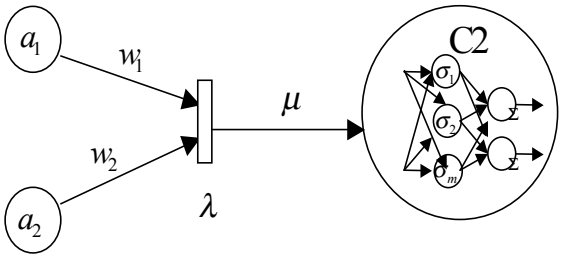

Fig. 7. Hybrid model 2

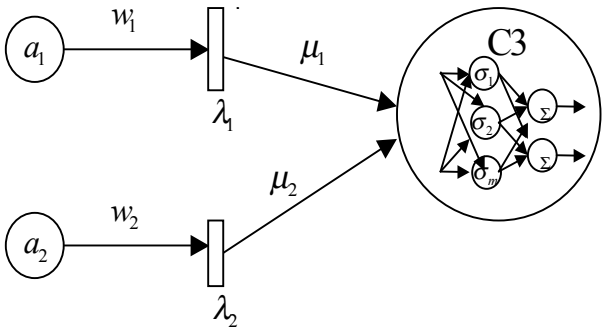

Fig. 8. Hybrid model 3

R2: If C Then F, λ_C (CF=μ_2)

R3: If F Then G, λ_F (CF=μ_3)

R4: If D and E Then G, $\lambda_D, \lambda_E, w_D, w_E$ (CF=μ_4).

Based on above translation principle, we map this system Γ into an AFPN which is shown as Fig. 9.

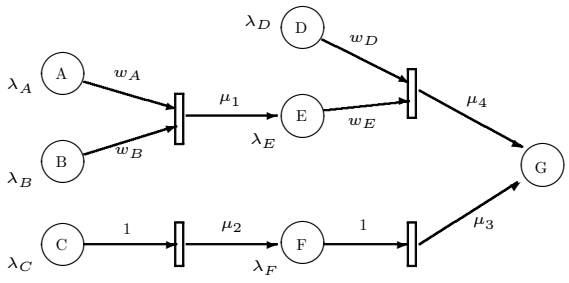

Fig. 9. AFPN representation of a knowledge-based system

Assume we know the following real data

$$\lambda_A = 0.5,\ \lambda_B = 0.8,\ \lambda_C = 0.3,$$
$$\lambda_D = 0.8,\ \lambda_E = 0.1,\ \lambda_F = 0.4,$$
$$\mu_1 = 0.8, \mu_2 = 0.9, \mu_3 = 0.6, \mu_4 = 0.7$$

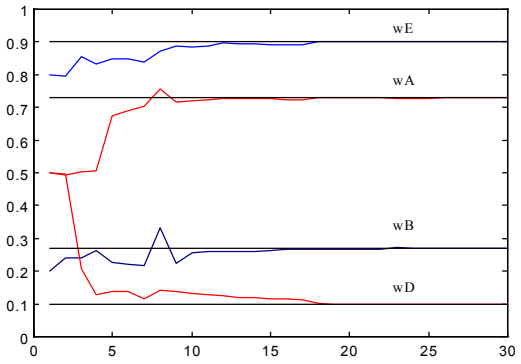

Fig. 10. Weights learning

As the desired weights

$$w_A^* = 0.73,\ w_B^* = 0.27,\ w_D^* = 0.1,\ w_E^* = 0.9 \quad (12)$$

are unknown, we will use neural networks technique to learn these weights from actual data. Given any input $\Upsilon = \{\alpha(p_A), \alpha(p_B), \alpha(p_C), \alpha(p_D)\}$, the human expert can give the corresponding output $\Psi = \{\alpha(p_E), \alpha(p_F), \alpha(p_G)\}$. For example suppose that for $\Upsilon_1 = \{0.7, 0.9, 0.5, 0.9\}$, the desired output is $\Psi = \{0.5148, 0.45, 0.3406\}$; and for $\Upsilon_2 = \{0.4, 0.2, 0.9, 0.7\}$, the desired output is $\Psi = \{0, 0.81, 0.648\}$. Let the initial a priori weights be:

$$w_A(0) = 0.5; w_B(0) = 0.5; w_D(0) = 0.2; w_E(0) = 0.8$$

To apply Widrow-Hoff learning law, we make $t(k)$ the desired outputs $t_{\alpha(p_E)}(k) = [0.5148, 0]$, $t_{\alpha(p_G)}(k) = [0.3406, 0.648]$, $y(k) = [\alpha(p_E), \alpha(p_G)]$, $P_1(k) = [\alpha(p_A), \alpha(p_B)]$, $P_2(k) = [\alpha(p_D), \alpha(p_E)]$, $\delta = 1.7$. The learning results are shown in Fig. 10. One can see that after 20 steps, the weights converge to their real values, i.e., AFPN mathematically models the knowledge based system Γ.

Example 2. A typical batch process cell shown schematically in Fig.11. The cell consists of two reactors and one mixing tank.

The reactor $P4$ is controlled by the command $P1$: IF $P1$ THEN $P4(\lambda_1,\mu_1)$.

The reactor $P5$ is controlled by the command $P2$ and also reactor $P4$ is finished

IF $P2$ AND $P4$ THEN $P5(w_2, w_4, \lambda_3, \mu_3)$

The mixing tank $P6$ is started if reactor $P5$ is finished, or there is input from another batch cell $P3$

IF $P3$ AND $P5$ THEN $P6(\lambda_2, \mu_2, \lambda_4, \mu_4)$*First*, based on the translation principle, we map the logic system into an AFPN as follows (shown as Fig.12).

$$FPN = \{P, T, D, I, O, \alpha, \beta, Th, W\}$$

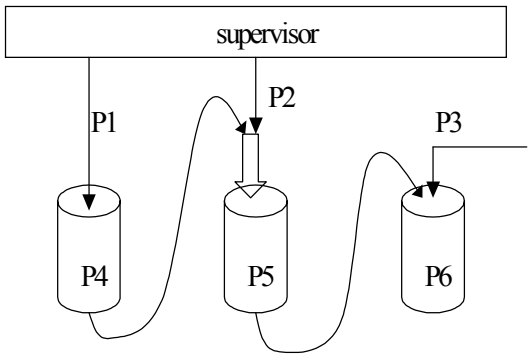

Fig. 11. Batch process cell scheme

Example 3.

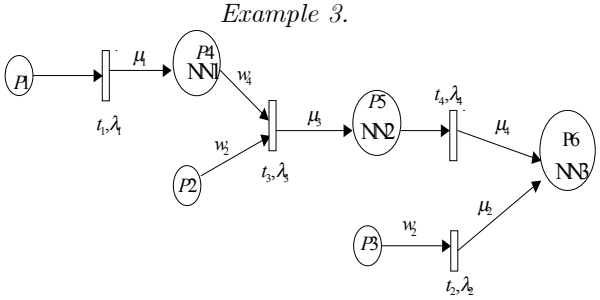

Fig. 12. Hybrid model of batch cell

where $P = \{p_1, p_2, p_3, p_4, p_5, p_6\}$, $T = \{t_1, t_2, t_3, t_4\}$, $D = \{P1, P2, P3, P4, P5, P6\}$, $Th = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$, $W_I = \{w_2, w_4\}$, $W_O = \{\mu_1, \mu_2, \mu_3, \mu_4\}$.

The two reactors and one mixing tank are modeled by three dynamic neural networks NN1-NN3. We may see that the combination of Petri net and neural networks can model logic-based hybrid systems easily.

6. CONCLUSION

This paper introduce a new method for hybrid system modeling. Since a lot of hybrid systems may be described by two aspects: knowledge parts which is easily represented by production rules, and continuous process which can be easily modeled by neural networks. We use fuzzy Petri net to model the knowledge part, and neural networks to model the detailed continuous processes. Two hybrid example show that the method proposed by us is very easy to be implemented.

7. REFERENCES

[1]
S. Chen, J. Ke, and J. Chang (1990), Knowledge representation using fuzzy Petri nets, *IEEE Trans. Knowledge and Data Engineering*, Vol.2, No.3, 311-319

X. Li, W. Yu and F. Lara(2000), Dynamic Knowledge Inference and Learning Under Adaptive Fuzzy Petri Net Framework, IEEE Trans. On System, Man, and Cybernetics, Part C, vol.30, no.4, 1-9.

R. David and H.Alla(1994), Petri Nets for Modeling of Dynamic Systems-A Survey, *Automatica*, Vol.30, No.2, 175-202.

K.J.Hunt, D.Sbarbaro(1992), R.Zbikowski and P.J.Gawthrop, Neural Networks for Control Systems- A Survey, *Automatica*, Vol. 28, 1083-1112.

K.H.Johansson, J.Lygeros, S.Sastry and M.Egersted(1999), Simulation of Zeno Hybrid Automata, 38th *IEEE Conf. on Decision and Control*, Phoenix, 3538-3543.

T.J.Koo and S.Sastry(1998), Output Tracking Control Design of a Helicopter Model Based on Approximate Linearization, *IEEE Conf. on Decision and Control*, Florida, 3635-3640.

J.Lygeros, C.Tomlin and S.Sastry(1999), Controllers for reachability Specifications for Hybrid Systems, *Automatica,* 3

G.A.Rovithakis and M.A.Christodoulou(1994), Adaptive Control of Unknown Plants Using Dynamical Neural Networks, *IEEE Trans. on Syst., Man and Cybern.*, Vol. 24, 400-412.

E.D.Sontag and Y.Wang(1995), On Characterization of the Input-to-State Stability Property, *System & Control Letters*, Vol.24, 351-359.

W.Yu and A.S.Poznyak(1999), Indirect Adaptive Control via Parallel Dynamic Neural Networks, *IEE Proceedings - Control Theory and Applications*, Vol.146, No.1, 25-30.

W.Yu and X. Li(2001), Some Stability Properties of Dynamic Neural Networks, *IEEE Trans. Circuits and Systems, Part I*, Vol.48, No.1, 256-259.

W.Yu and X. Li(2001), Some New Results on System Identification with Dynamic Neural Networks, *IEEE Trans. Neural Networks*, Vol.12, No.2, 412-417.

D.S. Yeung and E.C.C. Tsang(1998), A multilevel weighted fuzzy reasoning algorithm for expert systems, *IEEE Trans. SMC-Part A: Systems and Humans*, Vol.28, No.2, 149-158.