Dynamic Knowledge Inference and Learning under Adaptive Fuzzy Petri Net Framework

Xiaoou Li, Wen Yu, *Member, IEEE,* and Felipe Lara-Rosano, *Associate Member, IEEE*

***Abstract*—Since knowledge in expert system is vague and modified frequently, expert systems are fuzzy and dynamic systems. It is very important to design a dynamic knowledge inference framework which is adjustable according to knowledge variation as human cognition and thinking. Aiming at this object, a generalized fuzzy Petri net model is proposed in this paper, it is called adaptive fuzzy Petri net (AFPN). AFPN not only takes the descriptive advantages of fuzzy Petri net, but also has learning ability like neural network. Just as other fuzzy Petri net (FPN) models, AFPN can be used for knowledge representation and reasoning, but AFPN has one important advantage: it is suitable for dynamic knowledge, i.e., the weights of AFPN are ajustable. Based on AFPN transition firing rule, a modified back propagation learning algorithm is developed to assure the convergence of the weights.**

***Index Terms*—Expert system, fuzzy reasoning, knowledge learning, neural network, Petri net.**

I. Introduction

PETRI NETS (PNs) have ability to represent and analyze in an easy way concurrency and synchronization phenomena, like concurrent evolutions, where various processes that evolve simultaneously are partially independent. Furthermore, PN approach can be easily combined with other techniques and theories such as object-oriented 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.

PNs have an inherent quality in representing logic in intuitive and visual way, and FPNs take all the advantages of PNs. So, the reasoning path of expert systems can be reduced to simple sprouting trees if FPN-based reasoning algorithms are applied as an inference engine. FPN are also used for fuzzy knowledge representation and reasoning, many results prove that FPN is suitable to represent and reason misty logic implication relations [2], [3], [1], [12], [4], [8].

Knowledge in expert systems is updated or modified frequently, expert systems may be regarded as dynamic systems. Suitable models for them should be adaptable. In other words, the models must have ability to adjust themselves according to the systems' changes. However, the lack of adjustment (learning) mechanism in FPNs can not cope with potential changes of actual systems [5].

Manuscript received June 19, 1999; revised September 1, 2000.

X. Li and W. Yu are with the Sección de Computación, Departamento de Ingeniería Eléctrica, CINVESTAV-IPN, México City, D.F. 07360, México (e-mail: lixo@cs.cinvestav.mx).

F. Lara-Rosano is with the Centro de Instrumentos, Universidad Nacional Autonoma de México (UNAM), A.P. 70-186, Cd. Universitaria, México City, D.F. 04510, México.

Publisher Item Identifier S 1094-6977(00)11205-2.

Recently, some adjustable FPNs were proposed. [3] gave an algorithm to adjust thresholds of FPN, but weights' adjustments were realized by test. [6] proposed a generalized FPN model (GFPN) which can be transformed into neural networks with OR/AND logic neurons [5], thus, parameters of the corresponding neural networks can be learned (trained). In fact, the knowledge learning in [6] was under the framework of neural networks. Adaptive Fuzzy Petri Net (AFPN) [13] has also the learning ability of a neural network, but it does not need to be transformed into neural networks. However the learning algorithm in [13] is based on a special transition firing rule, it is necessary to know certainty factors of each consequence proposition in the system. Obviously, this restriction is too strict for an expert system.

In this paper, we propose a more generalized reasoning rule for AFPN. Back propagation algorithm is developed for the knowledge learning under generalized conditions. The structure of the paper is organized as follows: after the introduction of the FPN and AFPN models, the reasoning algorithm and the weight learning algorithm are developed, examples are included as an illustration.

II. Knowledge Representation and Fuzzy Petri Net

In this section, we will review weighted fuzzy production rules and FPN.

A. Weighted Fuzzy Production Rules

In many situations, it may be difficult to capture data in a precise form. In order to properly represent real world knowledge, fuzzy production rules have been used for knowledge representation [2]. A fuzzy production rule (FPR) is a rule which describes the fuzzy relation between two propositions. If the antecedent portion of a fuzzy production rule contains "AND" or "OR" connectors, then it is called a composite fuzzy production rule. If the relative degree of importance of each proposition in the antecedent contributing to the consequent is considered, Weighted Fuzzy Production Rule (WFPR) has to be introduced [7].

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

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

where

$a = \langle a_1, a_2, \ldots, a_n \rangle$ antecedent portion which comprises of one or more propositions connected by either "AND" or "OR";

c	consequent proposition;
μ	certainty factor of the rule;
Th	threshold;
w	weight.

In general, WFPRs are categorized into three types which are defined as follows.

Type 1: A Simple Fuzzy Production Rule

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

For this type of rule, since there is only one proposition a in the antecedent, the weight w is meaningless.

Type 2: A Composite Conjunctive Rule

$$R: \text{IF } a_1 \text{ AND } a_2 \text{ AND } \cdots \text{ AND } a_n \text{ THEN } c\,(CF=\mu), \lambda_1, \lambda_2, \ldots, \lambda_n, w_1, w_2, \ldots, w_n$$

Type 3: A Composite Disjunctive Rule

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

For Type 2 and Type 3, a_i is the ith antecedent proposition of rule R, and c the consequent one. Each proposition a_i can have the format "x is f_a", where f_a is an element of a set of fuzzy sets F. μ are the threshold and certainty factor of a simple or composite rule; λ_i, w_i are the threshold and weight of the ith antecedent of a composite conjuctive or disjunctive rule. In above definition, thresholds are assigned to antecedent propositions. For composite conjuctive rules, thresholds are assigned to the weighted sum of all antecedent propositions.

In this paper, in order to cope with Adaptive Fuzzy Petri Net (AFPN), we define WFPRs as following new forms:

Type 1: A Simple Fuzzy Production Rule

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

Type 2: A Composite Conjunctive Rule

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

Type 3: A Composite Disjunctive Rule

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

B. Definition of Fuzzy Petri Net

FPN is a promising modeling methodology for expert system [2], [6], [12]. A GFPN structure is defined as a 8-tuple [2]

$$\text{FPN} = (P, T, D, I, O, f, \alpha, \beta) \tag{1}$$

where

$P=\{p_1, p_2, \ldots, p_n\}$	set of places;
$T=\{t_1, t_2, \ldots, t_m\}$	set of transitions;
$D=\{d_1, d_2, \ldots, d_n\}$	set of propositions;
$I(O): T \to P^\infty$	input (output) function which defines a mapping from transitions to bags of places;

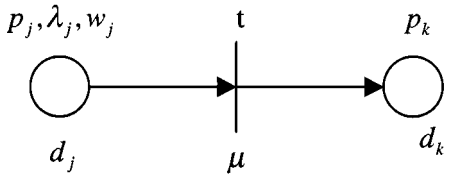

Fig. 1. FPN of Type 1 WFPR in [7].

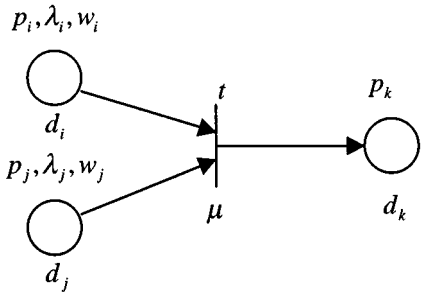

Fig. 2. FPN of Type 2 WFPR in [7].

$f: T \to [0,1]$	association function which assigns a certainty value to each transition;
$\alpha: P \to [0,1]$	association function which assigns a real value between zero to one to each place;
$\beta: P \to D$	bijective mapping between the proposition and place label for each node.

$P \cap T \cap D = \phi, |P| = |D|$.

In order to capture more information of the WFPRs, the FPN model has been enhanced to include a set of threshold values and weights, it consists of a 13-tuple [7]

$$\text{FPN} = (P, T, D, Th, I, O, F, W, f, \alpha, \beta, \gamma, \theta) \tag{2}$$

where

$Th=\{\lambda_1, \lambda_2, \ldots, \lambda_n\}$	set of threshold values;
$F=\{f_1, f_2, \ldots f_s\}$	set of fuzzy sets;
$W=\{w_1, w_2, \ldots, w_r\}$	set of weights of WFPRs;
$\alpha: P \to F$	association function which assigns a fuzzy set to each place;
$\gamma: P \to \text{Th}$	association function which defines a mapping from places to threshold values.

The definitions of P, T, D, I, O, f and β are the same as above. Each proposition in the antecedent is assigned a threshold value, and $\theta: P \to W$ is an association function which assigns a weight to each place.

C. Mapping WFPRs into FPN

The mapping of the three types of weighted fuzzy production rules into the FPNs in [7] are shown in Figs. 1, 2, and 3, respectively. For example, a rule of Type 2 may be represented as

R : IF d_1 AND d_2 AND $\cdots$ AND d_n THEN $c\,(CF=\mu), \gamma(p_j)=\lambda_j, \theta(p_j)=w_j, \alpha(p_j)=f_j, j=1,2,\ldots,n$ (tokens representing fuzzy sets of given facts).

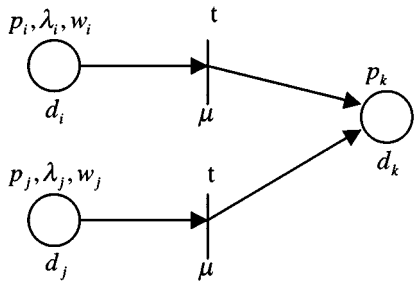

Fig. 3. FPN of Type 3 WFPR in [7].

III. Adaptive Fuzzy Petri Net

FPN [7] can represent WFPRs perfectly. But it can not adjust itself according to the knowledge updating. In another word, it has not learning ability. In this paper, we introduce the conception "adaptive" into FPN, the proposed model is called AFPN.

A. Definition of AFPN

Definition 1: An AFPN is a 9-tuple

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

where $P, T, D, I, O, \alpha, \beta$ are defined the same as [2]. $Th : T \to [0, 1]$ is the function which assigns a threshold value λ_i from zero to one to transition t_i. $Th = \{\lambda_1, \lambda_2, \ldots \lambda_m\}$. $W = W_I \cup W_O$. $W_I : I \to [0, 1]$ and $W_O : O \to [0, 1]$, are sets of input weights and output weights which assign weights to all the arcs of a net.

B. Mapping WFPR into AFPN

The mappings of the three types of WFPR into the AFPNs are shown as Figs. 4, Section III-B, and 5 respectively. The three types of WFPR may be represented as follows.

Type 1: A Simple Fuzzy Production Rule

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

Type 2: A Composite Conjunctive Rule

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

Type 3: A Composite Disjunctive Rule

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

The mapping between AFPN and WFPR 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).

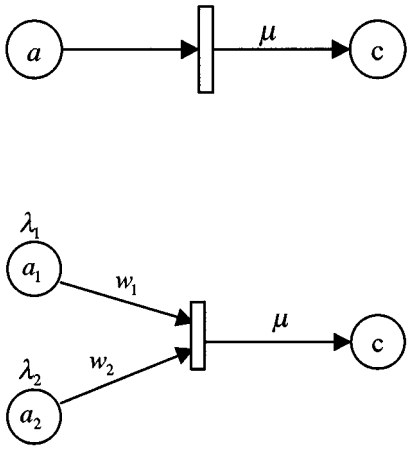

Fig. 4. AFPN of Type 1 WFPR.

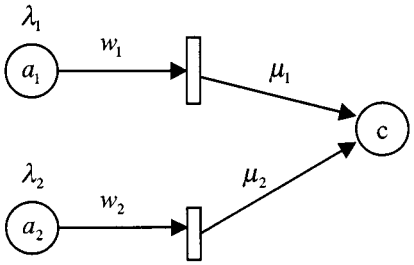

Fig. 5. AFPN of Type 3 WFPR.

C. Fuzzy Reasoning Using AFPN

Firstly, we give some basic definitions which are useful to explain the transition firing rule of AFPN.

Definition 2 (Source Places, Sink Places): A place p is called a source place if it has no input transitions. It is called a sink place if it has no output transitions.

A source place corresponds to a precondition proposition in WFPR, and a sink place corresponds to a consequent. For example, in Fig. 6, P_1, P_2, P_3 are source places, P_6 is a sink place.

Definition 3 (Route): Given a place p, a transition string $t_1 t_2 \ldots t_n$ is called a route to p if p can get a token through firing this transition string in sequence from a group of source places. If a transition string fire in sequence, we call the corresponding route *active*.

For a place p, it is possible that there are more than one route to it. For example, in Fig. 6, $t_1 t_3 t_4$ is a route to P_6, t_2 is another route to it. Let $I(t) = \{p_{i1}, p_{i2}, \ldots, p_{in}\}, w_{I1}, w_{I2}, \ldots, w_{In}$ the corresponding input weights to these places, $\lambda_1, \lambda_2, \ldots, \lambda_n$ thresholds. Let $O(t) = \{p_{o1}, p_{o2}, \ldots, p_{om}\}$, and $w_{o1}, w_{o2}, \ldots, w_{om}$ the corresponding output weights to these places.

We divide the set of places P into three parts $P = P_{\text{UI}} \cup P_{\text{int}} \cup P_O$, where P is the set of places of AFPN; $P_{\text{UI}} = \{p \in P \mid {}^{\cdot}p = \emptyset\}, p \in P_{\text{UI}}$ is called a user input place; $P_{\text{int}} = \{p \in P \mid {}^{\cdot}p \neq \emptyset \text{ and } p^{\cdot} \neq \emptyset\}, p \in P_{\text{int}}$ is called an interior place; $P_O = \{p^{\cdot} = \emptyset\}, p \in P_O$ is called an output place. In this paper, $\emptyset$ is an empty set.

Definition 4: The marking of a place $m(p)$ is defined as the certainty factor of the token in it.

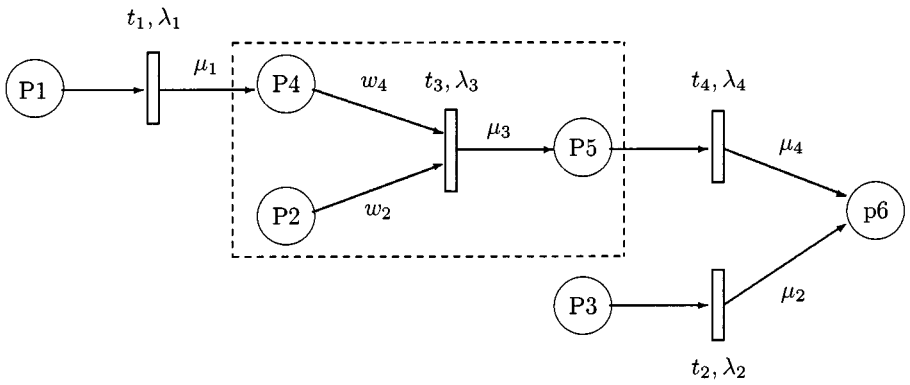

Fig. 6. AFPN of Example 1.

Definition 5: $\forall t \in T, t$ is enabled if $\forall p_{Ij} \in I(t), m(p_{Ij}) > 0, j = 1, 2, \ldots, n$.

Definition 6: When t is enabled, it produces a new certainty factor $CF(t)$

$$CF(t) = \begin{cases} \sum_j m(p_{Ij}) \cdot w_{Ij}, & \sum_j m(p_{Ij}) \cdot w_{Ij} > Th(t) \\ 0, & \sum_j m(p_{Ij}) \cdot w_{Ij} < Th(t). \end{cases}$$

We may use a continuous function $CF(t)(x)$ to approximate $CF(t)$

$$CF(t)(x) = x \cdot F(x)$$

where

$$x = \sum_j m(p_{Ij}) \cdot w_{Ij}$$

where $F(x)$ is a sigmoid function which approximates the threshold of t

$$F(x) = 1 \Big/ \left(1 + e^{-b(x - \mathrm{Th}(t))}\right)$$

where b is an instant. If b is big enough, when $x > Th(t), e^{-b(x-\mathrm{Th}(t))} \approx 0$, then $F(x) \approx 1$, and when $x < Th(t), e^{-b(x-\mathrm{Th}(t))} \to \infty$, then $F(x) \approx 0$.

Definition 7: If $x > Th(t)$, transition t fires, at the same time, token transmission takes place.

1) If a place p_{ok} only has one input transition t, a new token with certainty factor $w_{ok} \cdot CF(t)$ is put into each output place $p_{ok}, k = 1, 2, \ldots, m$, and all tokens in $p_{Ij}, j = 1, 2, \ldots, n$ are removed.
2) If a place p_{ok} has more than one input transitions (as Fig. 5), and more than one of them fire, i.e. more than one routes are active at the same time, then the new certainty factor of p_{ok} is decided by the center of gravity of the fired transitions

$$m(p_{ok}) = \frac{\sum_j [w_{oj} \cdot CF(t_j)]}{\sum_j w_{oj}}$$

where t_j fires, $t_j \in {}^{\cdot}p_{ok}$.

According to above definitions, a transition t is enabled if all its input places have tokens, if the certainty factor produced by it is greater than its threshold, then t fires, so an AFPN can be implemented. Thus, through firing transitions, certainty factors can be reasoned from a set of known antecedent propositions to a set of consequent propositions step by step.

Let $T_{\mathrm{initial}} = \{t \in T \,|\, {}^{\cdot}t \cap P_{\mathrm{UI}} \neq \emptyset \text{ and } {}^{\cdot}t \cap P_{\mathrm{int}} = \emptyset\}, t \in T_{\mathrm{initial}}$ is called an initially enabled transition.

Let $T_{\mathrm{current}} = \{t \in T_{\mathrm{initial}} \,|\, \forall p_i \in {}^{\cdot}t, m(p_i) > 0, \text{ and } CF(t) > Th(t)\}, t \in T_{\mathrm{current}}$ is called a current enabled transition.

Fuzzy Reasoning Algorithm

INPUT: the certainty factors of a set of antecedent propositions (correspond to P_{UI} in AFPN)

OUTPUT: the certainty factors of a set of consequence propositions (correspond to $P_{\mathrm{int}} \cup P_O$ in AFPN)

Step 1) Build the set of user input places P_{UI}.
Step 2) Build the set of initially enabled transitions T_{initial}.
Step 3) Find current enabled transitions T_{current} according to *Definition 5*.
Step 4) Calculate new certainty factors produced by fired transitions according to *Definition 6*.
Step 5) Make token transmission according to *Definition 7*.
Step 6) Let $T = T - T_{\mathrm{current}}, P = P - {}^{\cdot}T_{\mathrm{current}}$.
Step 7) Go to *Step 3* and repeat, until $T_{\mathrm{current}} = \emptyset$.

IV. Knowledge Learning and AFPN Training

In [13], we developed a weights learning algorithm under following conditions.

1) It is necessary to know the certainty factors of all output places (i.e. the right hand of all rules).
2) Only one layer of weights can be learned.
3) For rules of Type 3, if there are more than one transition fire, we must know which input transition is the token contributor to the output place.
4) In case 2 of the Definition 7, error distribution.

These conditions are very strict, because these information in real expert systems may be not available. In this paper we will relax these conditions to more general cases. The main idea is that all layer weights can be updated through the back-propagation algorithm if certainty factors of all sink places are given.

Back propagation algorithm

We assume that

- AFPN model of an expert system has been developed;

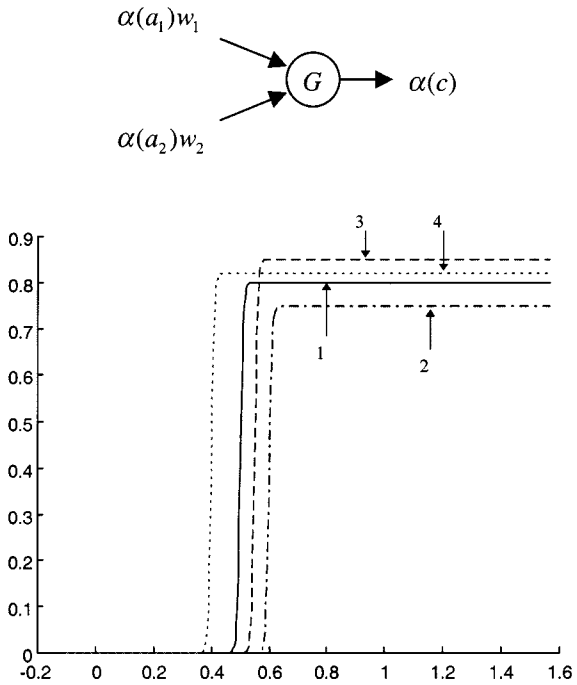

Fig. 7. Sigmiod functions in Example 1.

- in AFPN model, *Th* and W_O are known;
- set of certainty factor values of $P_{\rm UI}$ and P_O is given.

Here we take Type 2 as an illustration to show knowledge learning procedure using AFPN. Type 2 can be translated into an AFPN like Section III-B, this AFPN structure can be translated further into a neural networks-like structure (see Section IV), where G is

$$G(x) := f(x) \cdot x \tag{3}$$

where

$x = W^T\Lambda$	$= \sum_{i=1}^{n} \alpha_i w_i, f(x) = \mu/1 + e^{-b(x-\lambda)}$ sigmoid function;
b	constant which adjust the steepness of $f(x)$;
W	weight vector $W := [w_1, w_2 \dots w_n]^T$;
α	output vector of previous layer, $\Lambda := [\alpha_1, \alpha_2 \dots \alpha_n]^T$.

This continuous function may approximate a logic factor if μ, b and λ are selected suitable values. For example, no. 1 in Fig. 7 has the values as $\mu = 0.8, b = 200$ and $\lambda = 0.5$.

For a place p, there are some learning routes which are from a set of source places to it. The weights in these routes can be trained according the back propagation algorithm developed in this section. Along the selected route Ω, the feedforward propagation process (one hidden layer) is that given any input data U and the fixed weights w_i, the output $O(\Omega)$ can be expressed

$$O(\Omega) = G^{(n)}\left\{W^{(n)}G^{(n-1)}\left[W^{(n-1)}G^{(n-2)}\dots W^{(2)}G^{(1)} \times \left(W^{(1)}U\right)\right]\right\}$$

TABLE I
RESULTS OF AFPN

Group No.	α(P1)	α(P2)	α(P3)	α(P4)	α(P6)	α(P6)
1	0.2190	0.0470	0.6789	0	0	0.3243
2	0.6793	0.9347	0.3835	0.5434	0.5850	0.3055
3	0.5194	0.8310	0.0346	0.4072	0.4517	0.2359
4	0.0535	0.5297	0.6711	0	0	0.3206
5	0.0077	0.3834	0.0668	0	0	0
6	0.4175	0.6868	0.5890	0	0	0.0279
7	0.9304	0.8462	0.5269	0.7443	0.6647	0.3472
8	0.0920	0.6539	0.4160	0	0	0
9	0.7012	0.9103	0.7622	0.5610	0.5867	0.6705
10	0.2625	0.0475	0.7361	0	0	0.3516

where $G^{(k)}\ (k = 1 \dots n)$ is the active function of the k_th layer, $W^{(k)}\ (k = 1 \dots n)$ is the weight of the k_th layer. If the real data is O^*, the output error vector is

$$e_n = O(\Omega) - O^*.$$

Since we do not process the tokens in the output layer, the output layer may be selected as the rule of the center of gravity (see Definition 7), i.e.,

$$G^{(n)}(x) = \frac{x}{\sum_j w_{oj}} \tag{4}$$

The learning algorithm is the same as the backpropagation of multilayer neural networks:

- The weights in output layer is updated as

$$W^{(n)}(k+1) = W^{(n)}(k) - \gamma_i e_n \Lambda^{(n)}$$

where

$\Lambda^{(n)}$	input of the n_th layer;
$\gamma_i > 0$,	adaptive gain;
$W(k)$	weight at the time of k.

$$e_i = e_{i+1}\dot{G}^{(i+1)}(x)W^{(i)} \tag{5}$$

the weights are updated as

$$\begin{aligned} W^{(n-1)}(k+1) &= W^{(n-1)}(k) - \gamma_{n-1}\dot{G}^{(n-1)}e_{n-1}\Lambda^{(n-1)} \\ &\vdots \\ W^{(2)}(k+1) &= W^{(2)}(k) - \gamma_2\dot{G}^{(2)}e_2\Lambda^{(2)} \\ W^{(1)}(k+1) &= W^{(1)}(k) - \gamma_1\dot{G}^{(1)}e_1\Lambda^{(1)} \end{aligned} \tag{6}$$

where $\dot{G}$ is the derivative of the nonlinear function G.

$$\begin{aligned} \dot{G} &:= \frac{d}{dx}\left[\frac{\mu \cdot x}{1 + e^{-b(x-\lambda)}}\right] \\ &= \frac{\mu x b e^{-b(x-\lambda)}}{\left(1 + e^{-b(x-\lambda)}\right)^2} + \frac{\mu}{1 + e^{-b(x-\lambda)}}. \end{aligned} \tag{7}$$

The justification of backpropagation can be found in [11]. Finally, we summarize the learning algorithm of AFPN as

Step 1) Select a set of initial weight values.
Step 2) For each set of input data, find all active routes, and mark them $\Omega_1, \Omega_2, \ldots, \Omega_k$.
Step 3) Following each active route, according to the reasoning algorithm, calculate the corresponding output.
Step 4) Set the difference between the idea output and the calculated output as the error e, select γ_n use (6) to adjust the weights on these routes.

V. Simulation

In this section, two typical examples are selected to show the results in the prior sections.

Example 1: $P1, P2, P3, P4, P5$ and $P6$ are related propositions of an expert system Γ_1. Between them there exist the following weighted fuzzy production rules

$R1$: IF $P1$ THEN $P4$ (λ_1, μ_1)
$R2$: IF $P2$ AND $P4$ THEN $P5$ $(w_2, w_4, \lambda_3, \mu_3)$
$R3$: IF $P3$ OR $P5$ THEN $P6$ $(\lambda_2, \lambda_4, \mu_2, \mu_4)$.

This example includes all the three types of rules, in which $R1$ is a simple WFPR, $R2$ is a composite conjunctive one, and $R3$ is a composite disjunctive one. We want to show the fuzzy reasoning and the weights learning algorithm.

First, based on the translation principle, we map Γ_1 into an AFPN as follows (shown as Fig. 6)

$$\text{AFPN}_1 = \{P, T, D, I, O, \alpha, \beta, Th, W\}$$

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

We have three input propositions ($P1, P2$ and $P3$) and three consequence propositions ($P4, P5$ and $P6$). The data are given as

$$\begin{aligned} &\mu_1 = 0.80, \quad \mu_2 = 0.75, \quad \mu_3 = 0.85, \quad \mu_4 = 0.82 \\ &\lambda_1 = 0.50, \quad \lambda_2 = 0.60, \quad \lambda_3 = 0.55, \quad \lambda_4 = 0.40 \\ &w_4 = 0.63, \quad w_2 = 0.37 \end{aligned}$$

We use four sigmoid functions as

$$F_i(x) := \frac{\mu_i}{1 + e^{-b_i(x - \lambda_i)}}, \quad i = 1, 2, 3, 4$$

to approximate the four thresholds $\lambda_1, \lambda_2, \lambda_3, \lambda_4$, the steepness b_i are selected as 200 (see Fig. 7). Especially, for the transition t_3, the argument of function $G(x)$ is

$$x = \alpha(P4)w_4 + \alpha(P2)w_2.$$

Using fuzzy reasoning algorithm, a set of output data (certainty factors of consequence propositions) can be calculated according to the input data (certainty factors of antecedent propositions). Table I gives the results of AFPN.

One can see that some data are 0. This means that the corresponding thresholds were not passed. For example, in Group 1,

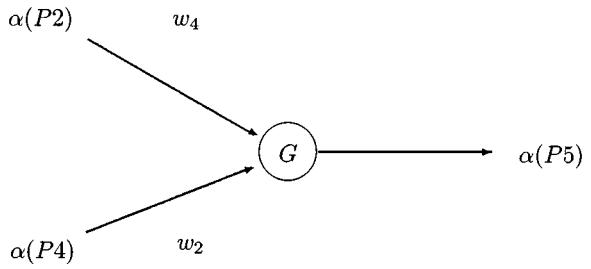

Fig. 8. The neural network translation of the learning part in Example 1.

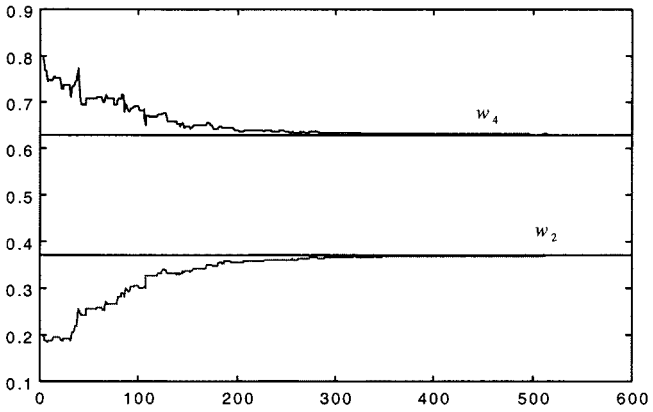

Fig. 9. Single layer learning results of Example 1.

$\alpha(P1) = 0.2190$, the threshold λ_1 is 0.50. Since $\alpha(P1) < \lambda_1$, transition t_1 cannot fire, so the output certainty factor is $\alpha(P4)$ is 0. The use a sigmoid function to approximate a threshold means that exact zero is impossible to get (for example, 0.0001). But if the steepness coefficient is small enough, the sigmoid function can approximate the threshold with good accuracy.

If the weights are unknown, neural networks technique may be used to estimate the weights. The learning part of the AFPN (see the part in the dashed box in Fig. 6) may be formed as a standard single layer neural networks (see Fig. 8). Assume the ideal weights are

$$W_4 = 0.63, \quad W_2 = 0.37.$$

The sigmoid function is

$$F_3(x) := \frac{0.85}{1 + e^{-200(x - 0.55)}}. \tag{8}$$

If the inputs $\alpha(P1)$ and $\alpha(P2)$ are given random data from 1 to 0, we can get the real output $\alpha(P5)$ according to the expert system Γ_1. Given any initial condition for w_4 and w_2, put the same inputs to the neural network. The error between the output of neural network $(\alpha'(P5))$ and that of the expert system $\Gamma_1(\alpha(P5))$ can be used to modified the weights, we may use the following learning law

$$\begin{aligned} W(k+1) &= W(k) + \dot{G}\delta e(k)\Lambda(P(k)) \quad \delta > 0 \\ e(k) &:= \alpha(P5)(k) - \alpha'(P5)(k) \end{aligned} \tag{9}$$

where δ is learning rate, a small δ may assure the learning process is stable. Here, we select $\delta = 0.07$.

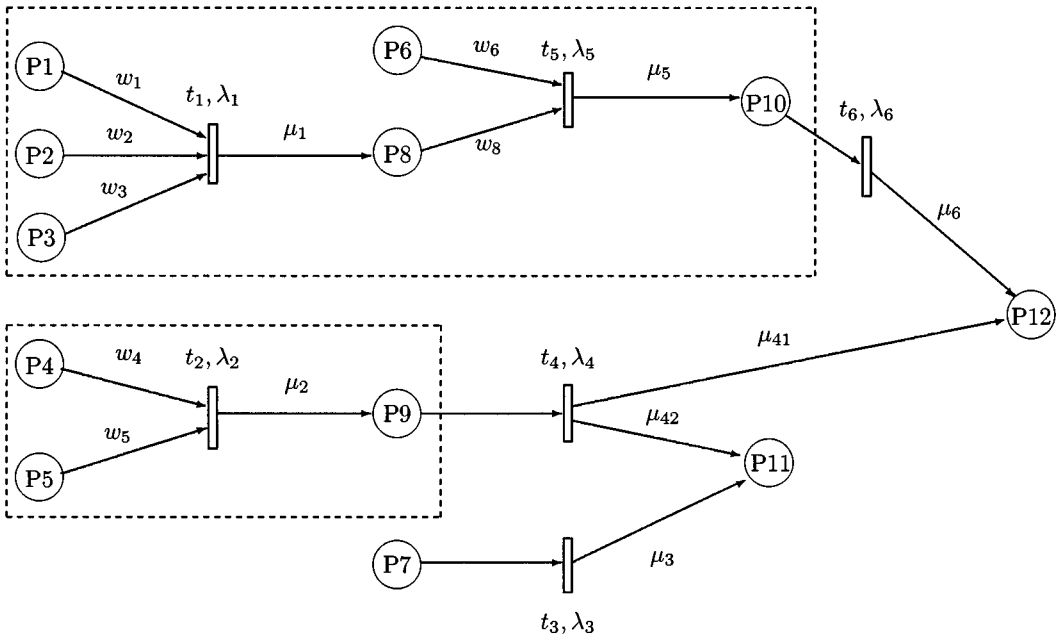

Fig. 10. AFPN of Example 2.

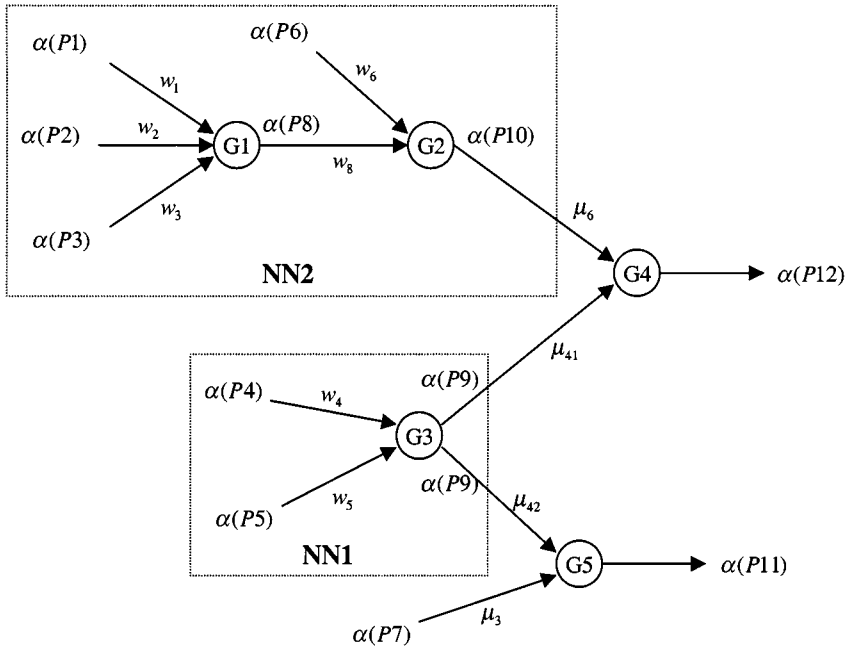

Fig. 11. The neural networks translation of the AFPN in Fig. 12.

$W(k) = [w_2(k), w_4(k)]$, $\Lambda(P(k)) = [\alpha(P2(k)), \alpha(P4(k))]$, and

$$G(x) = \frac{0.85x}{1 + e^{-200(x-0.55)}}.$$

After a training process ($k > 400$), the weights convergence to real values. Fig. 9 shows simulation results.

In this example there is only one learning layer. Example 2 will show a more complicated case where two learning layers (multilayer perceptrons) is used.

Example 2: $P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11$ and $P12$ are related propositions of an expert system Γ_2. There exist the following weighted fuzzy production rules

$R1$: IF $P1$ AND $P2$ AND $P3$ THEN $P8$ $(w_1, w_2, w_3, \lambda_1, \mu_1)$
$R2$: IF $P4$ AND $P5$ THEN $P9$ $(w_4, w_5, \lambda_2, \mu_2)$
$R3$: IF $P6$ AND $P8$ THEN $P10$ $(w_6, w_8, \lambda_5, \mu_5)$
$R4$: IF $P7$ OR $P9$ THEN $P11$ $(\lambda_3, \lambda_4, \mu_3, \mu_{42})$
$R5$: IF $P9$ OR $P10$ THEN $P12$ $(\lambda_4, \lambda_6, \mu_{42}, \mu_6)$

Based on the translation principle, we map Γ_2 into an AFPN (see Fig. 10).

$$\text{AFPN}_2 = \{P, T, D, I, O, \alpha, \beta, Th, W\}$$

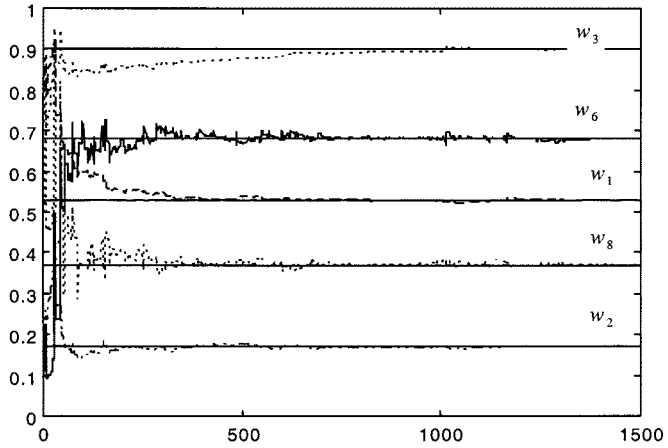

Fig. 12. MLP learning results of Example 2.

where

$$P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}, p_{12}\}$$
$$T = \{t_1, t_2, t_3, t_4, t_5, t_6\}$$
$$D = \{P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12\}$$
$$Th = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6\}$$
$$W_I = \{w_1, w_2, w_3, w_4, w_5, w_6, w_8\}$$
$$W_O = \{\mu_1, \mu_2, \mu_3, \mu_{41}, \mu_{42}, \mu_5, \mu_6\}$$

So AFPN model for this expert system may be repressed as in Fig. 10, the two dashed-boxes are the learning parts. This AFPN model may be transferred into a normal neural networks as Fig. 11.

Since the weights of $\mu_{41}, \mu_{42}, \mu_3$ and μ_6 are known, we may simplify this complex neural networks as two sub neural networks: NN1 and NN2. Here sub-networks NN1 is single layer and sub-networks NN2 is multilayer. The neural networks corresponding to G_5 are fixed.

We can train the two networks independently. The original learning error is e_{12}. Because the output function is select as (4)

$$G^{(4)}(x) = \frac{x}{\mu_{41} + \mu_6}, \quad x = \mu_{41}\alpha(P9) + \mu_6\alpha(P10)$$

- In case 1 of Definition 7, if only t_4 fires, then:

$$e_9 = \mu_{41}e_{12}, \quad e_{10} = 0$$

  if only t_6 fires, then:

$$e_{10} = \mu_6 e_{12}, \quad e_9 = 0$$

- In case 2 of Definition 7, when t_4 and t_6 fire at the same time, according to error backpropagation rule (5)

$$e_9 = e_{12} \times \frac{1}{\mu_{41} + \mu_6} \times \mu_{41} = \frac{\mu_{41}}{\mu_{41} + \mu_6} e_{12}$$
$$e_{10} = e_{12} \times \frac{1}{\mu_{41} + \mu_6} \times \mu_6 = \frac{\mu_6}{\mu_{41} + \mu_6} e_{12}$$

The learning algorithms for single layer neural network NN1 is the same as that in Example 1. The adaptive law for multilayer perceptrons NN2 is as in (6). We assume the ideal weights are

$$W_1 = 0.53, \quad W_2 = 0.17, \quad W_3 = 0.9, \quad W_8 = 0.37,$$
$$W_6 = 0.68$$

a set of data about the learning part of the AFPN

$$\lambda_1 = 0.5, \quad \lambda_5 = 0.4, \quad \mu_1 = 0.9, \quad \mu_5 = 0.8,$$
$$b_1 = b_2 = 20$$

Give a set of initial value of the weights

$$w_1(1) = 0.8, \quad w_2(1) = 0.2, \quad w_3(1) = 0.5,$$
$$w_8(1) = 0.7, \quad w_6(1) = 0.1$$

and the learning rate $\delta = 0.5$. The on-line MLP learning results are shown in Fig. 12.

From these two examples, we can see that the fuzzy reasoning algorithm and the back propagation algorithm are very effectively if we do not know the weights of AFPN. After a training process, we can get an excellent input–output mapping of the knowledge system.

VI. Conclusion

This paper introduce a new modified fuzzy Petri net: Adaptive Fuzzy Petri Net (AFPN). It has learning ability as neural networks. So fuzzy knowledge in expert systems can be learned through an AFPN model. The idea proposed in this paper is a new formal way to solve the knowledge learning problem in expert systems. Our ongoing research is to predict expert systems behavior using AFPN framework.

References

[1] H. Scarpelli, F. Gomide, and R. R. Yager, "A reasoning algorithm for high-level fuzzy Petri nets," *IEEE Trans. Fuzzy Syst.*, vol. 4, no. 3, pp. 282–293, 1996.

[2] S. Chen, J. Ke, and J. Chang, "Knowledge representation using fuzzy Petri nets," *IEEE Trans. Knowl. Data Eng.*, vol. 2, no. 3, pp. 311–319, 1990.

[3] C. G. Looney, "Fuzzy Petri nets and applications," in *Fuzzy Reasoning in Information, Decision and Control Systems*, S. G. Tzafestas and A. N. Venetsanopoulos, Eds. Norwell, MA: Kluwer, 1994, pp. 511–527.

[4] A. J. Bugarn and S. Barro, "Fuzzy reasoning supported by Petri nets," *IEEE Trans. Fuzzy Syst.*, vol. 2, no. 2, pp. 135–150, 1994.

[5] K. Hirota and W. Pedrycz, "OR/AND neuron in modeling fuzzy set connectives," *IEEE Trans. Fuzzy Syst.*, vol. 2, no. 2, pp. 151–161, 1994.

[6] W. Pedrycz and F. Gomide, "A generalized fuzzy Petri net model," *IEEE Trans. Fuzzy Syst.*, vol. 2, no. 4, pp. 295–301, 1994.

[7] D. S. Yeung and E. C. C. Tsang, "A multilevel weighted fuzzy reasoning algorithm for expert systems," *IEEE Trans. Syst., Man, Cybern. A*, vol. 28, no. 2, pp. 149–158, 1998.

[8] T. Cao and A. C. Sanderson, "Representation and analysis of uncertainty using fuzzy Petri nets," *J. Intell. Fuzzy Syst.*, vol. 3, pp. 3–19, 1995.

[9] S. M. Chen, "A fuzzy reasoning approach for rule-based systems based on fuzzy logics," *IEEE Trans. Syst., Man, Cybern. B*, vol. 26, no. 5, pp. 769–778, 1996.

[10] M. L. Garg, S. I. Ahson, and P. V. Gupta, "A fuzzy Petri net for knowledge representation and reasoning," *Inf. Process. Lett.*, vol. 39, pp. 165–171, 1991.

[11] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design*. Boston, MA: PWS, 1996, ch. 11.

[12] D. S. Yeung and E. C. C. Tsang, "Fuzzy knowledge representation and reasoning using Petri nets," *Expert Syst. Applicat.*, vol. 7, pp. 281–290, 1994.

[13] X. Li and F. L. Rosano, "Adaptive fuzzy Petri nets for dynamic knowledge representation and inference," *Expert Syst. Applicat.*, vol. 19, no. 3, 2000.

Xiaoou Li was born in China in 1969. She received the B.S. and the Ph.D. degrees in applied mathematics and electrical engineering from Northeastern University, Shenyang, China, in 1991 and 1995.

From 1995 to 1997, she was a Lecturer in electrical engineering with the Department of Automatic Control, Northeastern University. From 1998 to 1999, she was an Associate Professor of computer science with the Center for Instrumentation Research, National University of Mexico. Since 2000, she has been an Associate Professor of computer science with the Section of Computing, Department of Electrical Engineering, CINVESTAV-IPN, Mexico. Her research interests include Petri net theory and application, neural networks, artificial intelligence, computer integrated manufacturing, and discrete event systems.

Wen Yu (M'99) was born in Shenyang, China, in 1966. He received the B.S. degree from Tsinghua University, Beijing, China, in 1990 and the M.S. and Ph.D. degrees, both in electrical engineering, from Northeastern University, Shenyang, China, in 1992 and 1995, respectively.

From 1995 to 1996, he was a Lecturer with the Department of Automatic Control, Northeastern University. In 1996, he joined CINVESTAV-IPN, Mexico, where he is a Professor with the Departament of Automatic Control. His research interests include adaptive control, neural networks, and industrial automation.

Felipe Lara-Rosano (A'93) received the B.S. degree in civil engineering from the University of Puebla, Mexico, in 1962, the M.S. degree in mechanical and electrical engineering from the National University of Mexico in 1970 and the Ph.D. degree in engineering in the field of operations research from the National University of Mexico in 1973. Also he performed graduate studies at the University of Aachen, Aachen, Germany in the area of industrial engineering and instrumentation.

He joined the Systems Department, Institute of Engineering, National University of Mexico, in 1970 as Associate Researcher. In 1982, he was promoted to Senior Researcher. Additional posts at this University have included: Head of the Graduate Department for Engineering (1991–1993), Head of the Academic Senate for Mathematics, Physics and Engineering (1993–1997), Head of the Department of Computer Science at the Institute for Applied Mathematics and Systems (1997), and Director of the Centre for Instrumentation Research (1998 to present). His research interests include artificial intelligence, expert systems, theoretical and applied cybernetics, neural nets, fuzzy logic, complex systems analysis and modeling and Petri nets and applications. He has published more than 50 international journal papers, book chapters, and conference proceeding papers in his research areas and another 108 research articles in Mexican media. In addition, he has served as member of program committees of 31 scientific meetings.

Dr. Lara-Rosano was listed in the *2000 Marquis Who's Who in the World* and the *2000 Marquis Who's Who in Science and Engineering*. He was recipient of the Outstanding Scholarly Contribution Award from the Systems Research Foundation in 1995 and the Best Paper Award, for the Anticipatory, Fuzzy Semantic, and Linguistic Systems Symposium of the 3rd International Conference on Computing Anticipatory Systems, Liege, Belgium, in 1999. He is a member of the New York Academy of Sciences, the Mexican Academy of Sciences, the Mexican Academy of Engineering, and the Mexican Academy of Technology as well as an honorary doctorate from the International Institute for Advanced Systems Research and Cybernetics. He was elected to the office of president of the Mexican Society for Instrumentation in 1998 and Executive Secretary of the Mexican Academy of Technology in 2000. He is a fellow and board member of International Institute for Advanced Systems Research and Cybernetics.