Improved Algorithm on Rule-Based Reasoning Systems Modeled by Fuzzy Petri Nets

Rong Yang, Wing Shan Leung, Pheng Ann Heng, Kwong Sak Leung
Dept. of Computer Science & Engineering, The Chinese University of Hong Kong
Shatin, N.T., Hong Kong

Abstract: In this paper, we propose a complete and efficient algorithm to perform the fuzzy inference of a rule-based system modeled by Fuzzy Petri Nets (FPN). The algorithm is developed to simulate the inference process from the starting propositions to the goal propositions. The structures of the FPN are in more generic forms, compared to those applied in previous papers. The formal description of models and the fuzzy reasoning algorithm are described in detail, with several illustration examples. The differences between other algorithms are also presented.

I. INTRODUCTION

Since Zadeh first proposed the concepts of fuzzy logic in 60s, the word “fuzzy” shows itself frequently all around the world, from the top papers in science to the common daily necessaries. Among various applications of fuzzy logic, fuzzy rule-based reasoning has attracted many researchers’ attention. Through years’ hard work, there sprout out a number of efficient representation schemes for fuzzy rules.

Matrix representation of the implications between statements was first proposed in [1]. The flexibility of this knowledge representation and inference process was achieved greatly in the following researches [2][3][4]. However, although matrix representation possesses its advantage of computational cost, the matrices structure limits its expressive abundance of the fuzzy rules. For this reason, Petri nets (PN) appeared on the scene [5].

The technique of fuzzy reasoning via transformation of fuzzy truth state vectors by fuzzy rule matrices is extended to Fuzzy Petri nets (FPN) in [6]. An inference algorithm, which is suitable to simple PN structure, is introduced and implemented in this approach. In [7], Chen gave a more precise and complete definition on FPN, and the complexity of the system was increased by this approach. However, we find the fuzzy reasoning algorithm proposed in [7] not to be working with all types of situations, and the PN structures was still limited into simple ones.

Knowledge in rule-based system is updated and modified frequently, so the models must have the ability to learn by themselves according to the systems’ changes. To achieve this purpose, a generalized FPN model, which can be transformed easily into neural networks, is proposed in [8]. Yeung [9] introduced a multilevel weighted fuzzy reasoning algorithm. After that, adaptive fuzzy Petri nets was proposed in [10], where the model not only can be implemented to do knowledge inference, but also has a learning ability like a neural network.

We find either in the fuzzy reasoning algorithm years ago [6][7], or those appear recently [10], there exist the following limitations and incorrect respondence:

• Some paths from the starting antecedent proposition to end consequence propositions in the FPN are not considered, such as in [7].
• Not suitable to parallel reasoning, such as in [6][7].
• If the adjacent places of current place is not inferred from user-input or source places, but other places in the FPN, then the algorithm responses in a wrong way, such as in [7].
• After one transition fired, all its input places will be deleted in [10]. If there exist other transitions whose set of input places includes the deleted one, the inference process cannot continue further.

In this paper, aim at those limitations, we present a more complete and efficient algorithm for rule-based reasoning system modeled as FPN. The algorithm is developed to simulate the inference process from the starting propositions to the goal propositions. The structures of the FPN are in more generic forms, compared to those proposed in previous papers. It has the ability of parallel reasoning, and can be used to solve more complicated problems efficiently. Based on this algorithm, more advanced reasoning scheme such as backward reasoning, or self-learning reasoning can be developed.

The organization of the paper is as follows. In Section II, we briefly review the concepts of fuzzy Petri nets. The definitions and notations needed in following illustration are also introduced in this section. In Section III, we present the fuzzy forward reasoning algorithm using fuzzy Petri nets. Some examples are provided in Section IV to illustrate the reasoning process via this algorithm. The differences between our algorithm with others are presented in section V. Finally, the conclusion and further development are given in section VI.

II. FUZZY REASONING VIA PETRI NETS

A fuzzy Petri net (FPN) is a directed, weighted, bipartite graph consisting of two kinds of nodes, called places and transitions, where arcs are either from a place to a transition or from a transition to a place. In graphical representation, places are drawn as circles, transitions as bars. Applying FPN into rule-based system, each rule is presented as transitions, while proposition in those rules are presented as places.
place may or may not contain a token associated with the degree of truth between zero and one to denote the degree of truth of the corresponding proposition. Each transition is associated with a certainty factor (CF) value between zero and one to express the degree of certainty of the corresponding rules.

**EXAMPLE 1**: the following fuzzy production rule
R1: IF have a long nose THEN it’s an elephant (CF=0.75). can be modeled as shown in Fig. 1.

![Fig. 1. An example of fuzzy Petri net.](image)

where,
- \( p_1 \rightarrow \text{have a long nose}, \ p_2 \rightarrow \text{it’s an elephant} \)
- If we put a token valued 0.90 into \( p_1 \), after \( t_1 \) fires, the degree of truth of proposition corresponding to \( p_2 \) will be given as \( 0.9 \times \text{CF} = 0.9 \times 0.75 = 0.675 \).

According to the definition in [7], a FPN structure for rule reasoning can be defined as a 6-tuples:

\[
\text{FPN} = (P, T, I, O, F, W)
\]

where,
- \( P = \{ p_1, p_2, \ldots, p_n \} \) is a finite set of places, corresponding to propositions;
- \( T = \{ t_1, t_2, \ldots, t_n \} \) is a finite set of transitions, corresponding to rules;
- \( I : T \rightarrow P \) is an input function, maps transition to bags of its input places;
- \( O : T \rightarrow P \) is an output function, maps transition to bags of its output places;
- \( F : T \rightarrow [0,1] \) is a function, maps transition to a real value between zero and one;
- \( W : P \rightarrow [0,1] \) is a function, maps places to a real value between zero and one.

A transition \( t_i \) may be fired if for all \( p_j \in I(t_i) \), \( W(p_j) \geq \lambda \), where \( \lambda \) is a threshold value between zero and one. In Petri net structures describing fuzzy reasoning, tokens in places represent truth implication of corresponding proposition. For this reason, when transition \( t_i \) is fired, only the copies of tokens in its input places \( p_j \in I(t_i) \) are passed to its output places \( p_k \in O(t_i) \).

According to these definitions, the fuzzy production rule in Example 1 can be defined as:

\[
\text{FPN} = (P, T, I, O, F, W)
\]

- \( P = \{ p_1, p_2 \} \)
- \( p_1 \rightarrow \text{have a long nose}, \ p_2 \rightarrow \text{it’s an elephant} \)
- \( T = \{ t_1 \} \)
- \( I(t_1) = \{ p_1 \} \)
- \( O(t_1) = \{ p_2 \} \)
- \( F(t_1) = 0.75; \ W(p_1) = 0.90; \ W(p_2) = 0 \)

Let \( \lambda = 0.5 \), \( t_1 \) can be fired for \( W(p_1) = 0.90 > \lambda \). The fuzzy Petri net after firing is shown in Fig. 2.

![Fig. 2. Fuzzy Petri net after firing.](image)

There exist some cases in fuzzy rules reasoning when the antecedent part or consequence part of a fuzzy rule contains “AND” or “OR” connectors. According to [7] and [11], such composite fuzzy production rules can be distinguished into the following types and their corresponding PN structures are depicted along:

**Type 1**: If \( p_1 \ AND \ p_2 \ AND \ldots \ AND \ p_m \) THEN \( p_z \) (CF = \( \mu \)). The fuzzy reasoning process of this type of rule can be modeled in Fig. 3.

**Type 2**: If \( p_1 \) THEN \( p_a \ AND \ p_b \ AND \ldots \ AND \ p_z \) (CF = \( \mu \)). The fuzzy reasoning process of this type of rule can be modeled in Fig. 4.

**Type 3**: If \( p_1 \ OR \ p_2 \ OR \ldots \ OR \ p_m \) THEN \( p_z \) (CF = \( \mu \)). The fuzzy reasoning process of this type of rule can be modeled in Fig. 5.

Let \( p_i \) be a place, and \( t_a \) be transition in a fuzzy Petri net \( P \). If \( p_i \in I(t_a) \), then \( p_i \) is called the **Nearest Backward Place** of \( t_a \). All the nearest backward places of \( t_a \) constitute the Set of Nearest Backward Places of \( t_a \), and denoted as \( \text{SNBP}(t_a) \).

![Fig. 3. Fuzzy reasoning process of Type 1](image)

![Fig. 4. Fuzzy reasoning process of Type 2](image)
If \( p_i \in O(t_a) \), then \( p_i \) is called the **Nearest Forward Place** of \( t_a \). All the nearest forward places of \( t_a \) constitute the Set of Nearest Forward Places of \( t_a \), denoted as \( \text{SNFP}(t_a) \). If there exists a sequence of arcs (through places and transitions in \( P \)) connecting from \( t_k \) to \( p_i \), then \( p_i \) is called the **Forward Place** of \( t_k \). All forward places of \( t_a \) constitute the Set of Forward Places of \( t_a \), denoted as \( \text{SFP}(t_a) \). Obviously, as to the definition of Petri nets, \( \text{SNBP}(t_a) \) contains all input places of transition \( t_a \), and \( \text{SNFP}(t_a) \) contains all output places of transition \( t_a \).

\[
\text{SNFP}(t_a) \subseteq \text{SFP}(t_a).
\]

**EXAMPLE 2**: Fig. 6 shows a graph of a FPN. The set of the nearest backward places \( \text{SNBP}(t_i) \), the set of the nearest forward places \( \text{SFP}(t_i) \), and the set of the Forward Places \( \text{FP}(t_i) \) of each transition in this net are enumerated in Table I. Please note, to simplify the figures, the degree of truth of corresponding places in the following examples are omitted.

III. FUZZY FORWARD REASONING ALGORITHM

In fuzzy reasoning system, we often want to know whether there exists an antecedent-consequence relationship from proposition \( p_i \) to proposition \( p_j \). If such a relationship exists, given the degree of truth of \( p_i \), what the degree of truth of \( p_j \) might be. In this section, we present an innovate algorithm performing efficient and effective fuzzy forward reasoning.

Assume \( p_i \) is a place in net \( P \), if there does not exist such a transition \( t_i \), so that \( p_i \in \text{SNFP}(t_i) \), we define \( p_i \) as a **seed place**. The degree of truth of seed place is either already known before reasoning starts or provided by user during reasoning process. Generally, the reasoning process gets started from their seed places in fuzzy Petri net. Therefore, the seed place whose degree of truth has already be given before process starts are called **starting place**. The place, whose degree of truth we are interested in, is called the **goal place**. For an instance, let \( p_a, p_b, \ldots, p_z \) be places in a fuzzy Petri net \( P \), given the degree of truth of \( p_a \) and \( p_b \), we desire to know whether there exists an antecedent-consequence relationship between \( p_a \) and \( p_b \), if such a relationship exists, what’s the degree of truth of \( p_b \) should be. Here, \( p_a \) and \( p_b \) are starting places, and \( p_z \) is our goal place.

We examine in turn all transitions whose \( \text{SFP} \) contain the goal place. Before illustration of our algorithm, some definitions are given as followings:

- **Node**: node \( n_i \) is in form of \( (p_i, \text{W}(p_i)) \), where \( p_i \) is a place in net, and \( \text{W}(p_i) \) is the degree of truth of \( p_i \).
- **Known Nodes Set (KNS)**: if the degree of truth \( \text{W}(p_i) \) of place \( p_i \) in node \( n_i \) is already known by user input or by inference during reasoning process, then node \( n_i \) becomes a member of the KNS. A transition would fire under the conditions that all places in its \( \text{SNBP} \) can find their corresponding nodes in KNS.
- **WTS (Waiting Transition Set)**: is a set of transitions which are waiting to fire.
- **UFTS (Un-Fired Transition Set)**: is a set of transitions which haven’t fired.

Let \( t_i \) be a transition in net, \( \text{F}(t_i) \) represent the certainty value of \( t_i \). \( p_j \) be a place in net, \( \text{W}(p_j) \) denote the degree of truth of \( p_j \). The algorithm is now presented as follows:

**Fuzzy Forward Reasoning Algorithm**

**INPUT**: starting place \( p_i \) and its degree of truth \( \text{W}(p_i) \), where \( \text{W}(p_i) \in [0,1] \).

**OUTPUT**: \( \text{W}(p_g) \), the degree of truth of goal place \( p_g \).

**STEP 1**: create node \( n_i(\text{W}(p_i)) \), put \( n_i \) into KNS.

**STEP 2**: for all transitions \( t_i \in P \)

- if \( p_i \in \text{SFP}(t_i) \) then put \( t_i \) into WTS.

**STEP 3**: copy WTS \( \rightarrow \) UFTS

start do-while loop if UFTS \( \neq \) \( \emptyset \)

- for all \( t_i \in \) UFTS

\[
\begin{array}{|c|c|c|c|}
\hline
\text{Transition} t_i & \text{SNBP}(t_i) & \text{SNFP}(t_i) & \text{SFP}(t_i) \\
\hline
\text{t}_1 & \{p_0\} & \{p_1\} & \{p_1,p_3,p_5,p_6\} \\
\text{t}_2 & \{p_1\} & \{p_2\} & \{p_2,p_3,p_4,p_6\} \\
\text{t}_3 & \{p_2\} & \{p_1\} & \{p_2,p_4,p_5,p_6\} \\
\text{t}_4 & \{p_3,p_4\} & \{p_5\} & \{p_5,p_6\} \\
\text{t}_5 & \{p_3\} & \{p_4\} & \{p_4,p_5,p_6\} \\
\hline
\end{array}
\]
\{ \text{if } \text{SNBP}(t_i) \subseteq \text{KNS} \text{ then} \\
\quad \{ t_i \text{ fires} \\
\quad \quad \text{create node } n(p, W(p)), \text{ where } p \in \text{SNFP}(t_i), \\
\quad \quad \quad W(p) = \text{MIN}(W(\text{SNBP}(t_i))) \ast F(t_i) \\
\quad \quad \text{put node } n(p, w(p)) \text{ into KNS (if there} \text{ already exist } n_0(p_0, w(p_0)) \text{ where} \\
\quad \quad \quad p_0 = p, \text{ then replace } n_0 \text{ by } n \text{ if } w(p_0) \leq w(p), \text{ else keep } n_0 \} \\
\quad \text{remove } t_i \text{ from UFTS} \\
\} \\
\text{else if } \text{SNBP}(t_i) \not\subseteq \text{KNS} \text{ (let } p_j, \ldots, p_z \text{ be places} \text{ belong to SNBP}(t_i), \text{ but not} \text{ available in KNS) then} \\
\quad \{ \text{if } p_j, \ldots, p_z \text{ are all seed places then} \\
\quad \quad \{ \text{ask user input } W(p_j), \ldots, W(p_z) \}
\quad \quad t_i \text{ fires}, \\
\quad \quad \text{create node } n(p, w(p)), \text{ where } p \in \text{SNFP}(t_i), \\
\quad \quad \quad W(p) = \text{MIN}(W(\text{SNBP}(t_i))) \ast F(t_i) \\
\quad \quad \text{put node } n(p, w(p)) \text{ into KNS (if there} \text{ already exist } n_0(p_0, w(p_0)) \text{ where} \\
\quad \quad \quad p_0 = p, \text{ then replace } n_0 \text{ by } n \text{ if } w(p_0) \leq w(p), \text{ else keep } n_0 \} \\
\quad \text{remove } t_i \text{ from UFTS} \\
\quad \} \\
\quad \text{else} \\
\quad \quad \text{add } t_i \text{ to WTS} \\
\} \\
\text{copy WTS} \rightarrow \text{UFTS} \\
\text{reset WTS} \\
\} \\
\text{STEP 4: if there exists node } n(p_\theta, W(p_\theta)) \in \text{KNS AND} \\
W(p_\theta) \geq \lambda \text{ then} \\
\text{conclusion: the degree of truth of } p_\theta \text{ is} \\
W(p_\theta). \\
\text{else} \\
\text{conclusion: there exists no antecedent-} \\
\text{consequence relationship} \\
\text{between starting place and} \\
\text{goal place} \\
\}

IV. ILLUSTRATION EXAMPLES

To illustrate the fuzzy forward reasoning process performed in our algorithm, several examples are analyzed in this section.

\textbf{EXAMPLE 3}: Let \( p_1, p_2, p_3, p_4, p_5, p_6, p_7, \text{ and } p_8 \) be eight propositions. The rule-based system composed by these propositions contains the following fuzzy rules:

\begin{align*}
R1: & \text{ IF } p_1 \text{ THEN } p_4 \quad (\text{CF}=0.90) \\
R2: & \text{ IF } p_2 \text{ THEN } p_3 \quad (\text{CF}=0.80) \\
R3: & \text{ IF } p_3 \text{ THEN } p_6 \quad (\text{CF}=0.85) \\
R4: & \text{ IF } p_4 \text{ and } p_5 \text{ THEN } p_7 \quad (\text{CF}=0.95) \\
R5: & \text{ IF } p_5 \text{ and } p_6 \text{ THEN } p_8 \quad (\text{CF}=0.70)
\end{align*}

The antecedent-consequence relationship in rules can be modeled as a fuzzy Petri net \( P \) shown in Fig. 7.

For each transition \( t_i \in P \), the set of the nearest backward places \( \text{SNBP}(t_i) \), the set of the nearest forward places \( \text{SFBP}(t_i) \), and the set of the forward places \( \text{SFP}(t_i) \), are shown in Table II.

**TABLE II**: The set of the nearest backward place \( \text{SNBP}(t_i) \), the set of the nearest forward place \( \text{SNFP}(t_i) \), and the set of the forward place \( \text{SFP}(t_i) \) of each transition \( t_i \) in example 3

<table>
<thead>
<tr>
<th>Transition</th>
<th>( \text{SNBP}(t_i) )</th>
<th>( \text{SNFP}(t_i) )</th>
<th>( \text{SFP}(t_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>{ ( p_1 ) }</td>
<td>{ ( p_4 ) }</td>
<td>{ ( p_4, p_7 ) }</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>{ ( p_2 ) }</td>
<td>{ ( p_5 ) }</td>
<td>{ ( p_5, p_7, p_8 ) }</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>{ ( p_3 ) }</td>
<td>{ ( p_6 ) }</td>
<td>{ ( p_6, p_8 ) }</td>
</tr>
<tr>
<td>( t_4 )</td>
<td>{ ( p_4, p_5 ) }</td>
<td>{ ( p_7 ) }</td>
<td></td>
</tr>
<tr>
<td>( t_5 )</td>
<td>{ ( p_5, p_6 ) }</td>
<td>{ ( p_8 ) }</td>
<td></td>
</tr>
</tbody>
</table>

Given the degree of truth of proposition \( p_1 \) as 0.95, we want to know the degree of truth of proposition \( p_7 \) and \( p_8 \). Here, \( p_1, p_2, p_3 \) are seed places. Since \( p_7 \in \text{SFP}(t_i) \), where \( t_i \) might be \( t_1, t_2, t_3, t_4, t_5 \), so before do-while loop starts, \( \text{WTS} = \{ t_1, t_2, t_3, t_4, t_5 \} \), and \( \text{KNS} = \{ \{ p_1, 0.95 \} \} \). Now we check each transition \( t_i \) in \( \text{UFTS} \):

1) \( t_1 \), since \( \text{SNBP}(t_1) = \{ p_1 \} \subseteq \text{KNS} \), then \( t_1 \) fires, create node \( n(p_4, W(p_4)) \), where \( p_4 \in \text{SNFP}(t_1) \), and \( W(p_4) = W(p_1)^*F(t_1) = 0.95^*0.90 = 0.86 \). Add \( n \) into \( \text{KNS} \), and remove \( t_1 \) from \( \text{UFTS} \).

\( \text{UFTS} = \{ t_2, t_3, t_4, t_5 \} \), \( \text{KNS} = \{ \{ p_1, 0.95 \}, \{ p_4, 0.86 \} \} \)

2) \( t_2 \), since \( \text{SNBP}(t_2) = \{ p_2 \} \not\subseteq \text{KNS} \), but \( p_2 \) is a seed place, then ask user to input the degree of truth of \( p_2 \) (assume user input \( W(p_2) = 0.80 \)). Hence, \( t_2 \) fires, create node \( n(p_5, W(p_5)) \), where \( p_2 \in \text{SNFP}(t_2) \), and \( W(p_5) = W(p_2)^*F(t_2) = 0.80^*0.80 = 0.64 \). Add \( n \) into \( \text{KNS} \), and remove \( t_2 \) from \( \text{UFTS} \).

\( \text{UFTS} = \{ t_3, t_4, t_5 \} \), \( \text{KNS} = \{ \{ p_1, 0.95 \}, \{ p_4, 0.86 \}, \{ p_5, 0.86 \} \} \)

3) \( t_3 \), since \( \text{SNBP}(t_3) = \{ p_3 \} \not\subseteq \text{KNS} \), but \( p_3 \) is a seed place, then ask user to input the degree of truth of \( p_3 \) (assume user input \( W(p_3) = 0.90 \)). Hence, \( t_3 \) fires, create node \( n(p_6, W(p_6)) \), where \( p_3 \in \text{SNFP}(t_3) \), and \( W(p_6) = W(p_3)^*F(t_3) = 0.90^*0.85 = 0.77 \). Add \( n \) into \( \text{KNS} \), and remove \( t_3 \) from \( \text{UFTS} \).

\( \text{UFTS} = \{ t_4, t_5 \} \), \( \text{KNS} = \{ \{ p_1, 0.95 \}, \{ p_4, 0.86 \}, \{ p_5, 0.86 \}, \{ p_6, 0.77 \} \} \)

Fig. 7. Fuzzy Petri net in example 3
= W(p3)*F(t3) = 0.90*0.85 = 0.77. Add n into KNS, and remove t4 from UFTS.

UFTS = {t1, t5}

KNS = {{p1, 0.95},{p4, 0.86},{p6, 0.64},{p8, 0.77}}

4) t4, since SNBP(t4) = {p1, p4} ⊆ KNS, then t4 fires, create n(p4, W(p4)), where p4 ∈ SNFP (t4), and W(p4) = MIN(W(p1), W(p4))*F(t4) = MIN(0.90, 0.85)*0.85 = 0.70. Add n into KNS, and remove t4 from UFTS.

UFTS = {t1, t5}, KNS = {{p1, 0.95},{p4, 0.86},{p6, 0.64},{p8, 0.77}}

5) t5, since SNBP(t5) = {p5, p6} ⊆ KNS, then t5 fires, create n(p5, W(p5)), where p5 ∈ SNFP (t5), and W(p5) = MIN(W(p5)), W(p6))*F(t5) = MIN(0.95, 0.70) = 0.70. Add n into KNS, and remove t5 from UFTS.

UFTS = Φ, KNS = {{p1, 0.95},{p4, 0.86},{p6, 0.64},{p8, 0.77}}

Until now, all transitions has fired, and UFTS=Φ, thus, skip out of do-while loop. We find nodes corresponding to the goal places p7 and p8 are available in KNS. Assume the threshold value in this example is λ=0.30, both of the degree of truth of p7 and p8 are greater than λ, so we can draw the conclusion as:

The degree of truth of p7 and p8 are 0.61 and 0.50, respectively.

EXAMPLE 4: In this example, we consider a rules-based system modeled by Petri nets in a more complex structure, the one we show in Example 2. From the Petri net depicted in Fig. 6, p0, p1, p2, p3, p4, p5 and p6 are places, t1, t2, t3, t4, t5, t6, and t7 are transitions. Let F(t1) = 0.90, F(t2) = 0.85, F(t3) = 0.95, F(t4) = 0.80, F(t5) = 0.90, F(t6) = 0.75, F(t7) = 0.90. Assume p0 is the starting place and its degree of truth is given as 0.90, the starting proposition and the goal proposition. The algorithm for this process is based on whether the corresponding node of a place is a seed place, add to UFTS, or not. The algorithm is an improvement over existing ones, and suitable for more generic FPN forms which model complex rule-based systems. In this section, the improvements of our algorithm over Chen’s [7] and Li’s [10] approaches are presented as illustrations.

A. Improvement over Chen’s algorithm

Actually, Chen’s algorithm is not an inference algorithm, it is checking program to check whether there exists a antecedent-consequence relationship between the given starting proposition and the goal proposition. The algorithm is based on whether the corresponding node of a place is a terminal node, such a scheme limits the application of it only in simple FPN structures, and has the following disadvantages:

• Some paths from the starting antecedent propositions to end consequence propositions in the FPN are not...
considered. For example, let \( p_s \) be current node, if the goal place \( p_g \) belong to \( IRS(p_s) \), then Chen’s algorithm will ignore the path such as \( p_s \rightarrow p_1 \rightarrow p_2 \).

- When the adjacent place(s) \( p_2 \) of current place \( p_s \) is not a seed place whose degree of truth is inputted by user, but an arbitrary place whose truth of value is inferred from other places in FPN, Chen’s algorithm cannot continue the inference process further. It may ask the user to input the truth of value of place \( p_2 \), while the truth of value of \( p_2 \) probably has not been inferred through \( p_1 \).

The limitations in Chen’s algorithm shown above have been solved and improved by our algorithm. In our scheme, every path directed to the goal places is considered, and the one(s) to get the highest truth of value is chosen as the final certainty value of the goal place(s). Furthermore, our algorithm is suitable for more complex FPN structure. This can be shown in EXAMPLE II, while the inference of such a structure is impossible to be processed by Chen’s algorithm.

**B. Improvement over Li’s algorithm**

Li considered the firing of transitions in her algorithm, but once a transition has been fired, all its input places will be deleted from the set of places in FPN. Consider this case: If a place \( p_i \) belongs to the set of input places of two transitions \( t_1 \) and \( t_2 \), assume at this time, \( t_1 \) is enabled transitions while \( t_2 \) is not since some other input places of \( t_2 \) are still not available, then \( t_1 \) fires. After \( t_1 \) fires, all it input places will be deleted from set of places in FPN, including \( p_i \). Thus, transition \( t_2 \) will never fire since \( p_i \) doesn’t exist in FPN after the firing of \( t_1 \).

In our algorithm, when the truth of value of a place has been known, the corresponding node is put into a set named KNS. Whether a transition is an enabled transition depends on whether all its input places have their corresponding nodes in KNS. After firing, only the transition is deleted from UFTS, the new nodes due to the firing of this transition are put into KNS.

**VI. CONCLUSION AND FURTHER DEVELOPMENTS**

In this paper, a rule-based reasoning system has been modeled as a fuzzy Petri net. The forward reasoning process is simulated as tokens transferring from the starting places to the goal places in the net. Based on this structure, we derive an innovative and efficient algorithm. The algorithm concentrates on the firing of transitions relevant to the goal places until all such transitions have fired. Compared to the algorithm proposed in previous strategies [1][4][7][10], the structure of the FPN is in more generic and complex forms which are required by actual expert systems.

Based on this algorithm, more advanced reasoning schemes such as backward reasoning, or self-learning reasoning can be developed. Further developments focus on applying this algorithm onto Intelligent Chinese Acupuncture System (ICAS) [12].

**VII. ACKNOWLEDGEMENTS**

The work described in this paper was fully supported by the Research Grants Council of the Hong Kong Special Administrative Region (Project No. CUHK 4185/00E).

**REFERENCE**