UNIVERSITY OF OSTRAVA

Institute for Research and Applications of Fuzzy Modeling

Petri Nets as Fuzzy Modeling Tool

Viktor Pavliska

Research report No. 112

2006

Submitted/to appear:

Internal publication

Supported by:

MSM 6198898701 and project 1M0572 of the MŠMT ČR

University of Ostrava
Institute for Research and Applications of Fuzzy Modeling
30. dubna 22, 701 03 Ostrava 1, Czech Republic

tel.: +420-59-6160234 fax: +420-59-6120 478
e-mail: viktor.pavliska@osu.cz

Abstract

This article shows usage of modified Petri Nets as a tool for fuzzy modeling. Basic concepts and relations between Fuzzy Petri Nets and Fuzzy IF–THEN rules will be described and algorithm for decomposition of fuzzy Petri net into set of linguistic descriptions will be presented and its implementation mentioned.

1 Definitions

1.1 Fuzzy Petri Nets

Classical Petri Nets are defined as a structure $N = \langle S, T, F\rangle$, where S means set of places, T is set of transitions and F is $F \subseteq (S \times T) \cup (T \times S)$, where $(\forall\ t \in T)(\exists\ p, q \in S)(p, t), (t, q) \in F$. Graphical representation is set up by following symbols:

- Places – rings
- Transitions – rectangle
- Relations – pointers between transitions and places or places and transitions

We won't describe any more into details the idea and properties of basic Petri Nets and for deeper understanding of this problem we recommend literature. [1].

During process simulation by Petri Nets we have to time to time illustrate the status which we are not sure whether will happen or not. In classical Petri Nets there is token placed if the expression is true (1) or not if it is false (0). Let's now try too create such a Petri Net which will work with vague values ("small, "big", ...). We will use tools of fuzzy logic for work with such values especially fuzzy IF THEN rules [3].

Token will be bearer of fuzzy sets in our case, edges will be evaluated by linguistic expression from IF THEN rules and own transition represent basically fuzzy relation according to the IF THEN rule. Creation of that relation is depend on the chosen inferention method. For more implementation we refer to software LFLC 2000 developed UVAFM [6].

1.2 Fuzzy IF–THEN Rules

Fuzzy IF–THEN rule is a concept used for describing of logical dependence between variables of the following form:

$$\mathsf{IF}\ X_1\ \mathsf{is}\ \mathcal{A}_1\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_n\ \mathsf{is}\ \mathcal{A}_n\ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathcal{B}$$

where $\mathcal{A}_1, \ldots, \mathcal{A}_n$ and $\mathcal{B}$ are certain predicates characterizing the variables $X_1, \ldots, X_n$ and Y. They are often specified linguistically. We will work with specific kind of linguistic expressions as defined in [3]. For more information about logical structure of Fuzzy IF–THEN rules see [4].

The part before THEN is called the *antecedent* and the part after it the *succedent*. The variables $X_1, \ldots, X_n$ are called *input*, or *independent* variables. The variable Y (in general, there may be more of them, but we will limit oneself to only one) is called *output*, or *dependent* variable.

The fuzzy IF-THEN rules are usually put together to form the *linguistic description*

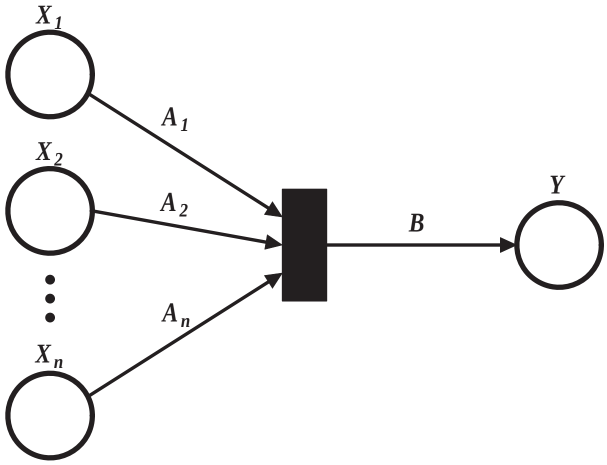
$$\begin{aligned}\mathcal{R}_1 :=&\ \mathsf{IF}\ X_1\ \mathsf{is}\ \mathcal{A}_{11}\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_n\ \mathsf{is}\ \mathcal{A}_{1n}\ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathcal{B}_1\\ &\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\ \mathcal{R}_m :=&\ \mathsf{IF}\ X_1\ \mathsf{is}\ \mathcal{A}_{m1}\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_n\ \mathsf{is}\ \mathcal{A}_{mn}\ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathcal{B}_m\end{aligned}$$

2 Modeling IF–THEN rules by Fuzzy Petri Net

Mapping between IF–THEN rules and fuzzy Petri nets is obvious. Any IF–THEN rule of the previous defined form:

$$\mathsf{IF}\ X_1\ \mathsf{is}\ \mathcal{A}_1\ \mathsf{AND}\ \cdots\ \mathsf{AND}\ X_n\ \mathsf{is}\ \mathcal{A}_n\ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathcal{B}$$

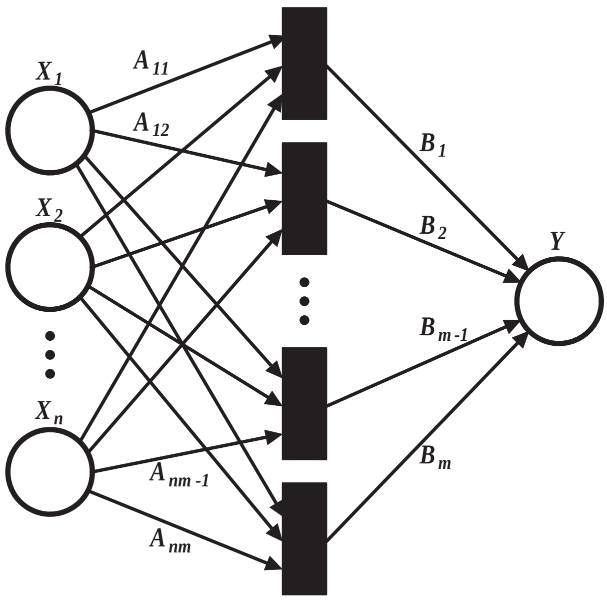
can be expressed by the following petri net:



The set of IF–THEN rules, which forms linguistic description:

$$\mathcal{R}_1 := \text{IF } X_1 \text{ is } \mathcal{A}_{11} \text{ AND } \cdots \text{ AND } X_n \text{ is } \mathcal{A}_{1n} \text{ THEN } Y \text{ is } \mathcal{B}_1$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$\mathcal{R}_m := \text{IF } X_1 \text{ is } \mathcal{A}_{m1} \text{ AND } \cdots \text{ AND } X_n \text{ is } \mathcal{A}_{mn} \text{ THEN } Y \text{ is } \mathcal{B}_m$$

can be modeled by the following way:

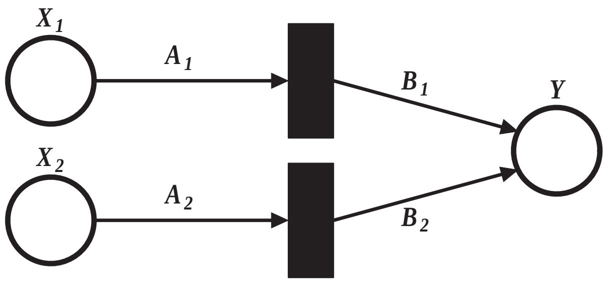


where each transition of the result fuzzy Petri net corresponds to one rule of linguistic description.

3 Construction of Linguistic Descriptions

Now lets focus on the opposite direction, so how to construct linguistic description from fuzzy Petri net. When the fuzzy Petri net is one of the previous forms, the corresponding linguistic description is obvious. But fuzzy Petri net may look something differently and we have to deal with this situations. Lets start with simple examples:

1. What will happen when some edges are missing:



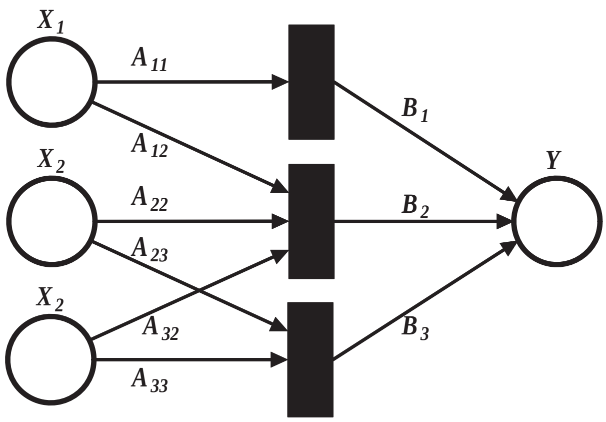
Two corresponding IF–THEN rules are:

$$\mathcal{R}_1 := \ \mathsf{IF}\ X_1\ \mathsf{is}\ \mathcal{A}_1\ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathcal{B}_1$$
$$\mathcal{R}_2 := \ \mathsf{IF}\ X_2\ \mathsf{is}\ \mathcal{A}_2\ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathcal{B}_2$$

To create linguistic description for one output variable Y and two input variables X_1, X_2, we may extend the set of predicates characterizing input variables by "UNDEF" linguistic expression which means that value of the variable may be undefined. So the linguistic description in this case will be following:

$$\mathcal{R}_1 := \ \mathsf{IF}\ X_1\ \mathsf{is}\ \mathcal{A}_1\ \mathsf{AND}\ X_2\ \mathsf{is\ UNDEF\ THEN}\ Y\ \mathsf{is}\ \mathcal{B}_1$$
$$\mathcal{R}_2 := \ \mathsf{IF}\ X_1\ \mathsf{is\ UNDEF\ AND}\ X_2\ \mathsf{is}\ \mathcal{A}_2\ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathcal{B}_2$$

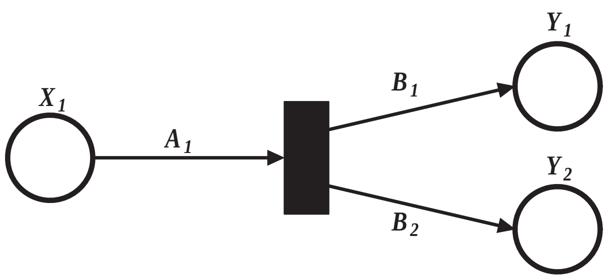
Generalization for more than two input variables is obvious. For example following fuzzy Petri net:



will lead to the following linguistic description:

$$\mathcal{R}_1 := \ \mathsf{IF}\ X_1\ \mathsf{is}\ \mathcal{A}_{11}\ \mathsf{AND}\ X_2\ \mathsf{is\ UNDEF\ AND}\ X_3\ \mathsf{is\ UNDEF\ THEN}\ Y\ \mathsf{is}\ \mathcal{B}_1$$
$$\mathcal{R}_2 := \ \mathsf{IF}\ X_1\ \mathsf{is}\ \mathcal{A}_{12}\ \mathsf{AND}\ X_2\ \mathsf{is}\ \mathcal{A}_{22}\ \mathsf{AND}\ X_3\ \mathsf{is}\ \mathcal{A}_{32}\ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathcal{B}_2$$
$$\mathcal{R}_3 := \ \mathsf{IF}\ X_1\ \mathsf{is\ UNDEF\ AND}\ X_2\ \mathsf{is}\ \mathcal{A}_{23}\ \mathsf{AND}\ X_3\ \mathsf{is}\ \mathcal{A}_{33}\ \mathsf{THEN}\ Y\ \mathsf{is}\ \mathcal{B}_3$$

2. Next thing, which we need to deal with, is presence of more than one output variable:



The corresponding linguistic description could look like this:

$$\mathcal{R} := \ \mathsf{IF}\ X_1\ \mathsf{is}\ \mathcal{A}_1\ \mathsf{THEN}\ Y_1\ \mathsf{is}\ \mathcal{B}_1\ \mathsf{AND}\ Y_2\ \mathsf{is}\ \mathcal{B}_2$$

but as we said above, we will limit oneself to only one output variable, so in this cases we will create several linguistic descriptions, one for each output variable. In this instance

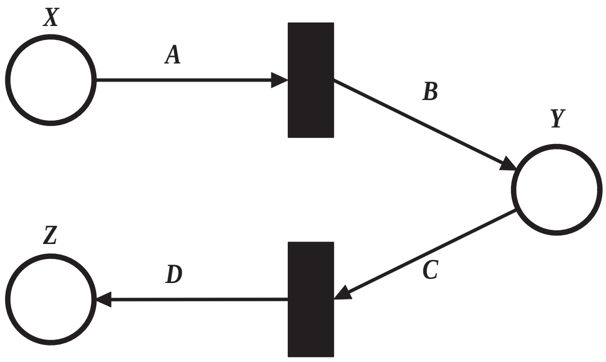
$$\mathcal{R} := \textsf{ IF } X_1 \textsf{ is } \mathcal{A}_1 \textsf{ THEN } Y_1 \textsf{ is } \mathcal{B}_1$$

for the first output variable and

$$\mathcal{R} := \textsf{ IF } X_1 \textsf{ is } \mathcal{A}_1 \textsf{ THEN } Y_2 \textsf{ is } \mathcal{B}_2$$

for the second output variable. In general for k output variables there will be constructed k linguistic descriptions with only one output variable in each of them.

3. The last situation which may arise is the case when an output place of a transition is simultaneously input place for another (exceptionally the same) transition. In this situation nothing bad happens and all up to now demonstrated methods for creating linguistic descriptions may be applied. For instance this fuzzy Petri net:



leads to two linguistic descriptions:

$$\mathcal{R} := \textsf{ IF } X \textsf{ is } \mathcal{A} \textsf{ THEN } Y \textsf{ is } \mathcal{B}$$

and

$$\mathcal{R} := \textsf{ IF } Y \textsf{ is } \mathcal{C} \textsf{ THEN } Z \textsf{ is } \mathcal{D}$$

When taking all around it together we are able to construct set of linguistic descriptions (number of linguistic descriptions is equal to number of output places) for any fuzzy Petri net as will be shown in the next section.

4 Algorithm for decomposition of fuzzy Petri net

Now the exact algorithm for decomposing fuzzy Petri net into set of linguistic descriptions which form Linguistic Fuzzy Logic Net (LFLN) will be shown. This algorithm gets fuzzy Petri Net as an input and creates set of linguistic descriptions corresponding to each output place of fuzzy Petri Net:

```
input  : fuzzy Petri net: fpn
output: set of linguistic descriptions: lfln
lfln = ∅;
foreach output place op of fpn do // create linguistic description
    // create set of input variables (places) on whose op depends
    inputs = ∅;
    foreach input transition it of op do
        // add all inputs of transition it to inputs set
        inputs = inputs ∪ it.inputs;
    end
    // construct linguistic description (set of rules)
    rb = ∅;
    foreach input transition it of op do
        // construct rule corresponding to transition it
        rule = ∅;
        foreach element in from inputs do
            if rule ≠ ∅ then rule = rule + AND;
            if in ∈ it.inputs then
                rule = rule + in.name is edge(in, it).value;
            else
                rule = rule + in.name is UNDEF;
            end
        end
        rule = rule + THEN op.name is edge(it, op).value;
        rb = rb ∪ rule ; // add rule to rule base
    end
    lfln = lfln ∪ rb ; // add rule base to set of linguistic descriptions
end
```

5 Implementation details

Algorithm presented above was implemented into a tool `fpn2lfln` which takes a file generated by The Petri Net Kernel (PNK 2.0) in PNML (XML) format as an input. PNML is general format for storing Petri net definition and information which is special for fuzzy Petri net was added naturally into PNML specification as it will be shown later. For more details about PNK and PNML see:

`http://www2.informatik.hu-berlin.de/top/pnk/index.html` or
`http://www2.informatik.hu-berlin.de/top/pnml/about.html`

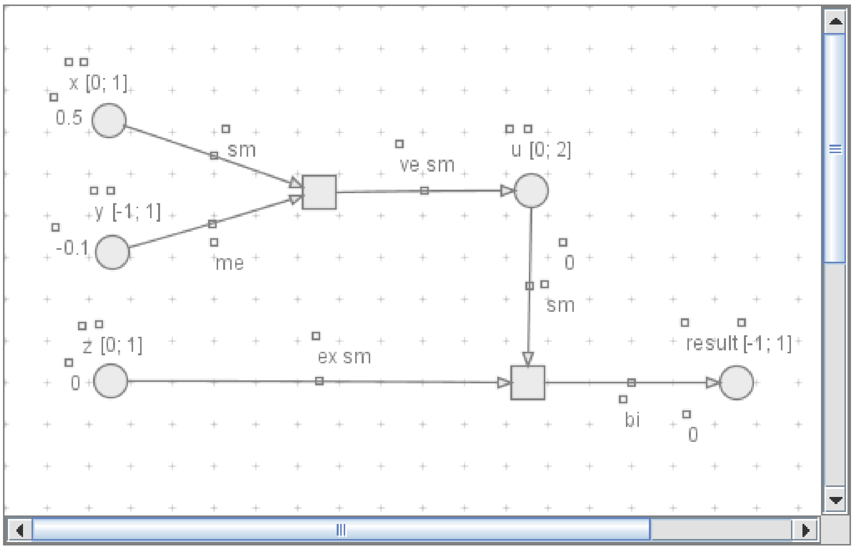
Developed tool `fpn2lfln` internally converts given *acyclic* fuzzy Petri Net into Linguistic Fuzzy Logic Net (LFLN). Then all output variables of LFLN are evaluated and results given to user. This computation is provided by calling inference method of linguistic descriptions corresponding to each output variable. Implementation of inference methods used for computation was taken from LFLC software package.

5.1 Meaning of PNML elements

PNML format enables to enter some kinds of additional information to Petri net elements. They were used by the following way:

- *Name* of Petri net place is used as name of corresponding fuzzy variable.
- *Marking* of a place is used for information about context of corresponding variable. It means the interval of min and max value of the variable.
- *Initial marking* of an input place holds information about initial value of corresponding input variable.
- *Arc value* carries linguistic value of IF–THEN rule.

Here is one example of fuzzy petri net defined in the PNK application and prepared for `fpn2lfln` conversion tool:



References

[1] Češka, M.: *Petriho sítě.* Akademické nakladatelství CERM, Brno, 1994

[2] Novák, V.: *Fuzzy Sets and Their Applications.* Adam Hilger, Bristol 1989.

[3] Novák, V.: *Základy fuzzy modelování.* BEN, Praha 2000. (Foundations of fuzzy modeling; in Czech)

[4] Novák, V., Lehmke, S.: *Logical Structure of Fuzzy IF–THEN Rules.* Research report No. 69, IRAFM, University of Ostrava.

[5] Knybel, J., Pavliska, V. *Representation of Fuzzy IF-THEN rules by Petri Nets.* In ASIS 2005. 6.9.2005-8.9.2005 Přerov. Ostrava : MARQ, 2005. s. 121-125. ISBN 80-86840-16-6.

[6] Dvořák, A., Habiballa, H., Novák, V., Pavliska, V.: *The software package LFLC 2000 – its specificity, recent and perspective applications.* Computers in Industry. 03. rel. 2003, vol. 51, p. 269 – 280