**Fuzzy Petri Nets:**

**A Fuzzy Reasoning Algorithm for System Simulation and Modeling**

Yilin Yang (yy450@scarletmail.rutgers.edu)

Department of Electrical and Computer Engineering, Rutgers University

**Abstract**

This paper discusses the application of fuzzy logic in Petri Net modeling. Conjoining these two separate analysis techniques yields Fuzzy Petri Nets (FPNs), a versatile method for describing and simulating processes that cannot strictly be represented using Boolean logic. FPNs have been utilized in multiple fields, from chemical science to artificial intelligence. Their ability to represent multiple states of a system sharing resources makes them desirable tools in limitless applications. This paper will describe FPN mechanisms before discussing usage of FPNs and their impact on multiple industries.

**1. Introduction**

Petri Net (PN) modeling is a favored technique for the analysis and simulation of systems states possessing static or quantifiable parameters. For rule-based systems that operate in principle using logical connectives, such as propositional calculus, it can be difficult describe succinctly the relationship between the constituent elements that make up the system. Techniques to model such systems, such as finite state machines (FSM) or binary search trees (BST), do exist but either lack inherent analysis tools for examining mathematical properties (in the case of the former) or struggle to adapt systems that cannot easily be described numerically (in the case of the latter).

As a graphical and mathematical modeling algorithm, Petri Nets can overcome these shortcomings. Using a bipartite graph and system state nodes, it is possible to accurately and intuitively depict the relationships and prerequisites of different stages of a given model system, so long as it is well defined by a consistent set of rules. This stipulation is crucial, however, and poses a challenge not only to Petri Net modeling, but algorithms in general.

For example, how can real-world knowledge be precisely defined and classified for computational purposes if the information is qualitative or ambiguous? The saturation point of a chemical varies depending on multiple factors such as concentration, temperature, and pressure. Attempting to describe these factors using Boolean logic is a challenge already, let alone saturation. It may be possible to describe this information using two states, such as light vs. dense, hot vs. cold, and strong vs. weak. With this, it can be possible to describe if a certain chemical has met thresholds for concentration, heat, and pressure.

However, saturation cannot be described using such definitions. If a low temperature is permissible given a high enough concentration and pressure, the threshold for hot vs. cold will vary. How can Boolean logic classify these inputs if their definitions will vary relative to other unknown inputs? The short answer is: it cannot.

The solution is to apply fuzzy logic instead, which can represent an infinite degree of truth. If saturation has a threshold of *x* and concentration, temperature, and pressure have thresholds of *a*, *b*, and *c*, it is possible to relate the input factors to the output factors using relative confidence ratings rather than absolute mutual exclusion. The application of fuzzy logic to Petri Net modeling allows for the realization of fuzzy reasoning algorithms using Fuzzy Petri Nets (FPN).

**2. Definitions**

A brief background is provided to explain the critical mechanisms of fuzzy logic, Petri Nets, and Fuzzy Petri Nets.

2.1 Petri Nets

A Petri Net can be depicted as a bipartite directed graph consisting of two types of nodes. Nodes may either be referred to as “places” identifying system states, graphically represented by circles, and “transitions” identifying system actions that may cause changes in the current state, graphically represented by quadrilaterals. These nodes are connected via directed arcs, fulfilling the same role as edges, which are either from a transition to a place or from a place to a transition. At any given time, a place may contain zero or more tokens to describe the status of the state, such as number of resources or occupancy of a location. Tokens are graphically represented by black dots. Every arc is designated a weight to describe the number of tokens it may consume from or deposit to the place. If not specified, weight is assumed to have a value of 1.

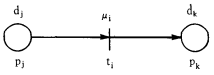


Figure 1: Example of a Petri Net

The formal definition is a 4-tuple described as the following:

PN = (P, T, F, W)

* P is a finite set of places such that: P = {p1, p2, …, pm}
* T is a finite set of transitions such that: T = {t1, t2, …, tn}
* F is a set of arcs where F ⊆ (P × T ) ⋃ (T × P)
* W is a multiset of weight associations of F, ex: W = (F, k)

A place p is called an input place of a transition t if and only if there exists a directed arc from p to t. A subset of T labelled •t denotes the set of input places for a transition t. Similarly, a place p is called an output place of transition t if and only if there exists a directed arc from t to p. Another subset known as t• denotes the set of output places for a transition t. The same conventions are used for notating •p and p•, which list the sets of transition sharing p as input places and output places, respectively.

The basic behavior of a Petri Net is the transfer of tokens from place to place. This process is referred to as “firing” tokens through the use of arc weights and transitions. Transition t is in an enabled state when the number of tokens of each member of •t is greater or equal to the weight of each corresponding arc F ⊆ (•t × T). When enabled, a transition may fire the tokens of •t into t•. Tokens are consumed from •t equal to the weight W = (F, •t) for each individual arc and deposits a number of tokens into t• equal to W = (F, t•).

2.2 Fuzzy Logic

In contrast to Boolean logic, which evaluates truth as one of two states, 0 or 1, fuzzy logic uses an infinite number of states consisting of decimal values between 0 and 1 inclusively. Decimal values serve to indicate a scalar representation of truthfulness based on their distance from the two absolute cases, 0 and 1. Values less than 0.5 share more similarities with case 0 while values greater than 0.5 share more similarities with case 1. Values exactly equal to 0.5 do not strongly lean toward either cases. Some implementations of fuzzy logic may choose to treat 0.5 as a special third case akin to three-valued logic.

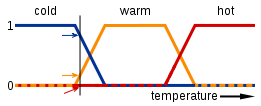


Figure 2: Graphical illustration of fuzzy logic to provide a generalized description of temperature

Fuzzy logic can also be used to describe element membership in a set. A fuzzy set is denoted as:

(U, m(x))

Here, U is a set and m is truth function. Let x represent a single member of set U. In such case, m(x) yields the degree of membership x has in set U. For elements where m(x) = 0, x is said to not be included in the fuzzy set (U, m(x)). For elements where m(x) is greater than 0 but less than 1, x is said to be a fuzzy member of the fuzzy set (U, m(x)). Finally, for elements where m(x) = 1, x is said to be a fully included member of the fuzzy set (U, m(x)).

This is in contrast to a non-fuzzy set where all members of the set are full members and no elements have partial membership. In a fuzzy set, elements that have no relation can technically be included, though it is common to only include elements with membership m(x) greater than 0 for simplicity.

2.3 Fuzzy Petri Nets

A Fuzzy Petri Net definition is an extension of a standard Petri Net given by the following 6-tuple:

PN = (P, T, F, W, λ, γ)

In addition to P, T, F, and W which were defined previously in Section 2.1, a FPN consists of λ, certainty factor associated with P, and γ, a threshold factor associated with T. FPNs have an additional requirement in the firing process for a transition to be enabled.

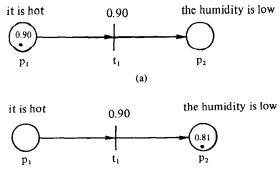


Figure 3: Example of a Fuzzy Petri Net firing process

In addition to the rules described in Section 2.1, for a transition t to be enabled, the associated truth value λ of p must be equal to or greater than the truth value γ of t. Values of γ are static whereas λ is calculated as λold = λnew \* γ. Figure 3 depicts an illustration of the firing process and calculation of the updated truth value for λ. The FPN algorithm can summarized as follows:

**FPN Algorithm**

1. for transition tn

2. iff tokens of •tn > weight of F ⊆ (•tn × T)

3. and λ of •tn > γ of t

4. fire tokens of •t into t•

5. assign weight of F ⊆ (•tn × T) to tokens of t•

6. assign tokens of •t – weight of F ⊆ (•tn × T) to tokens of •t

5. assign λ of •tn \* γ to λ of t•

**3. Analysis**

Because of the introduction of truth values associated to place and transition elements of a FPN, the firing process favors transitions t with the highest threshold γ that it can legally fire to, i.e. choose actions that the system is most certain it can accomplish. This allows for a sequential fuzzy action sequence where stages are executed in order of truth value. This is implemented entirely using only fuzzy transitions, which cannot be accomplished for a standalone PN. To mimic the same behavior for a regular PN would require the addition of a memory place for every counterpart transition such that in a FPN with m elements in •p and n elements in •t, a counterpart PN would require 2m elements in •p and n2 elements in •t. In other words, this mandates n2 more firings, each which must compute twice as many places compared to a FPN.

**4. Application**

The granular nature of evaluating information makes FPN analysis applicable in many different fields. One study found success in leveraging FPNs for the interpretation of sensory data in an autonomous vehicle. The kinematics of wheel adjustments was determined by analyzing the proximity of nearby objects using IR sensors. FPN calculations were performed to judge the distance and possible danger of objects through weighted scaling. Objects that were very far away posed little concern so minor adjustments were necessary to keep the vehicle safe from collision whereas objects within the immediate vicinity of the car were of the utmost concern and warranted major corrections to position. The degree of action is based on the degree of urgency, which is determined through the use of fuzzy logic.

**5. Related Work**

Many solutions have been proposed to further improve the base PN, FPN being one of the more prominent ones in recent decades. Another similar implementation to enhance the capabilities of the PN incorporated time based decision making to yield Time Petri Nets or TPNs. By controlling the interval at which a transition may be eligible to be enabled, it is possible to force certain action sequences based on time sensitive contexts. For example, if transitions t1 and t2 are both members of the set p• and have different time intervals such as 5 seconds for t1 and 10 seconds t2, the two transitions have equal weighting for 5 seconds and weighting in favor of t2 for 5 seconds, creating a 1:4 scale in favor of t2. This offers distinction from FPNs which are not aware of time-based contexts or limits.

**6. Conclusion**

In this paper the mechanisms, properties, and applications of Fuzzy Petri Nets (FPNs) were explored. Standard Petri Nets (PNs) provided a means of expressing knowledge and system processes that could not be replicated with other methods such as finite state machines. However, they too faced limitations in handling dynamic or amorphous information. This is remedied through the introduction of fuzzy logic into the algorithm. This inclusion greatly expands the degree of expressions that can be made to mathematically transcribe knowledge people would very easily form into words, such as “It is hot” or “It is cold”. The resulting FPN algorithm provides many improvements over the base PN algorithm, allowing for the modeling and simulation of systems in a much more compact and less clustered manner, greatly reducing task complexity. These advantages have been leveraged in industry, such as through the analysis of distance in an autonomous vehicle and position correction. Other solutions exist as well, such as the TPN, which seeks to address other concerns of the base PN algorithm. In short, the FPN model has been demonstrated to be a versatile tool for modeling and analysis of many different kinds of systems with applications in broad fields.

**7. References**

1. T. Murata, “Petri Nets: Properties, Analysis and Applications,” *Proc. of the IEEE,* Vol. 77, pp. 541-574, 1989
2. J.F. Chang, S.M. Chen, J.S. Ke, “Knowledge Representation Using Fuzzy Petri Nets,” *IEEE Transactions on Knowledge and Data Engineering,* Vol. 2, No. 3, pp. 311-319, 1990
3. E.C.C. Tsang, D.S. Yeung, “Fuzzy Knowledge Representation and Reasoning Using Petri Nets,” *Expert Systems with Applications,* Vol. 7, pp. 281-289, 1994
4. H. E Virtanen, “A study in fuzzy Petri nets and the relationship to fuzzy logic programming,” *Reports on Computer Science and Mathematics*, Ser. A., Åbo Akademi, No. 162. 1995
5. P.D. Baldoni, Y. Yang, S.Y. Kim, “Development of Efficient Obstacle Avoidance for a Mobile Robot using Fuzzy Petri nets”, *Proceedings of the IEEE 17th International Conference on Information Reuse and Integration*, pp. 265-269, 2016
6. M. Markiewicz, L. Gniewek, “A Program Model of Fuzzy Interpreted Petri Net to Control Discrete Event Systems, *Journal of Applied Sciences*, Vol. 7, 2017
7. P. Franco, P. Hansen, S.Y. Kim, “**Using Timed Petri nets to Regulate Optimal Game Strategy,”** *Proceedings of the ISCA International Conference on Computers and Their Applications*, pp. 207-213, 2018