Extension of Petri Nets and its Applications to Model Systems with Imprecise Task Duration

Stanislav Kurkovsky [1] and Rasiah Loganantharaj [2]
Department of Computer Science[1], Columbus State University, e-mail: kurkovsky_stan@colstate.edu
Automated Reasoning Laboratory, Center for Advanced Computer Studies[2]
University of Louisiana at Lafayette, e-mail: logan@cacs.usl.edu

***Abstract* – The framework of Petri nets is extended with imprecise temporal properties. Possibility theory is applied to model imprecise time by time-stamping tokens and assigning durations to firing of the transitions.**

I. INTRODUCTION

Careful examination of many real world applications reveals that they involve synchronization of several tasks and the durations of activities must be appropriately modeled as opposed to represent them by fixed quantities. Suppose, a machine C requires two units A and a unit of B to produce an assembled product ABA. The operation of machine C is said to be synchronized with the availability of two units of A and one unit of B. In real world applications, the duration of a task may vary and cannot be realistically modeled by a fixed number.

A Petri network is an ideal representation to model synchronization of tasks and to study the system's dynamic behavior and to prove some interesting properties such as deadlock and starvation. The framework of Petri nets does not explicitly provide the notion of time. Ordinary Petri nets may be viewed as operating according to some internal (system) clock. In this view each firing of a transition is associated with one clock cycle or tick. The length of the interval between ticks is not important, only the relations "before" and "after" among ticks are important. Occurrence of an event (firing of a transition) is not dependent on the number of internal ticks since the start of the clock. It depends only on the configuration of the net and its marking at the given clock cycle.

In this paper we will examine the previous work done on timed Petri network and extend it to handle tokens and transitions when they are representing imprecise time. We also show that how this extended Petri network will be used quite elegantly to model and reason with complex process application involving imprecise time. In many real world applications, the duration of a task is varying and it can be represented by a possibilistic distribution instead of using a fixed number as has been done in some crude models. When an interval is presented with possibilistic distribution, it is important to develop a temporal composition operation for propagating tokens across transitions in an extended Petri network. In this paper we describe the algorithms of the extended Petri network and provide a brief outline of temporal composition operation. Details of temporal composition operation can be found in our previous work [22].

The paper is organized as following. We provide the background information of Petri network in section2 and is followed by a review on timed Petri network. In section 4, we describe possibilistic representation of time and is followed by an algorithm for timed Petri network with possibilistic time. In section 6 we summarize the conclusion of this paper.

II. BACKGROUND INFORMATION OF PETRI NET THEORY

Ever since the introduction of the Petri net theory by Petri in his Ph.D. dissertation, it was widely used for modeling the dynamics of systems of the most diverse domains. Petri net is a graph-based structure consisting of places and transitions. The net simulates the dynamic behavior of a system by continuously firing enabled transitions when tokens are removed and inserted into places.

Let us illustrate the application of Petri Nets using the following example of an automated manufacturing system shown in Figure 1. The system is composed of four machines (Machine 1, Machine 2, Machine 3, and Machine 4) and two robots (Robot 1 and Robot 2). The system can process two types of parts, namely A and B. Part A needs sequential processing on Machine 1 and then on Machine 2. Part B needs processing on Machine 3. Machine 4 takes two parts A and one part B and assembles them into the final product. Robot 1 and Robot 2 handle loading/unloading operations between Machine 1 and Machine 2. The graph of the corresponding Petri net is shown on Figure 2. Availability of part A and part B is modeled by tokens in places p_1 and p_5 respectively. Processing of Machine 1 is

modeled by transition t_1, and its product after processing is modeled as a token in place p_2. When one token is available in place p_2 transitions t_2 and t_3 are enabled, but only one of them can fire. There is a non-determinism in selection which of these transitions should fire, which is usually resolved by randomly breaking a tie.

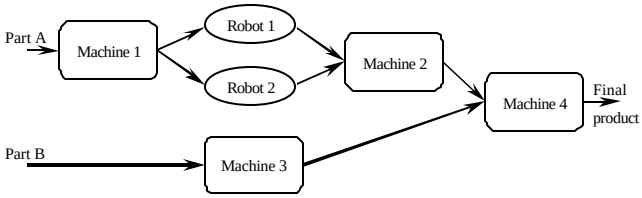

Figure 1. An automated manufacturing system

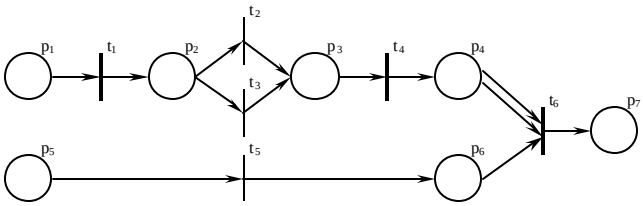

Figure 2. Petri net graph for the automated manufacturing system

In this representation, no information is given about the duration of operation of robots and machines. Alternatively, operations are assumed instantaneous, which is quite not realistic. Also, the capacity of the buffers (places p_1 through p_7) is assumed to be infinite.

III. Review of Timed and Fuzzy Petri Networks

Qualitative notion of time is implicitly represented in Petri in the sense that each firing of a transition is associated with one clock cycle or tick. The length of the interval between ticks is not important, only the relations "before" and "after" among ticks are important. Occurrence of an event (firing of a transition) is not dependent on the number of internal ticks since the start of the clock. It depends only on the configuration of the net and its marking at the given clock cycle.

To capture the quantitative notion of time, "external time" (or "external clock") was introduced in [2]. In such representation, the occurrence of the events depends not only on the marking, but also on the elapsed time since the occurrence of some other events. Having a single time line subsumes both internal and external times. There are several proposals incorporating the notion of time to virtually every component of the Petri nets framework, namely tokens, transitions, places, and arcs [3, 17, 20].

Because of the fact that any Petri net evolves by moving tokens over the places and the enabling of the transitions is dependent on the availability of tokens, it seems quite natural to associate time with tokens [15, 26, 27]. In most cases, the enabling of a transition depends on the timestamps of the tokens. This timestamp may be interpreted as the age of a particular token (how much time elapsed since it has been produced), or as the age of a sequence of tokens that are generated after a series of firing transitions.

Time can be associated with transitions too [4, 10, 26]. The possible interpretations of time q is associated with transition t_j include:

- t_j may (or must) fire only after q time units pass after t_j becomes enabled (q is assumed to be relative);
- t_j may fire only during time interval q (q is assumed to be absolute);
- the duration of firing of t_j is q (q is assumed to be relative). In this case t_j may (or must) start firing as soon as it becomes enabled.

First two approaches of associating time with transitions seem to be unnatural, because it is difficult to imagine a description of a system where certain processes can occur only within a given time frame. It is more likely that the processes taking place in a modeled system would have some length in time, which will be represented as durations of firing transitions of a Petri net.

Waiting time can be associated with places [8, 27]. Once token has been added to a place, it will not contribute to enabling any transition before the waiting time associated with that place has elapsed. Also, a transition must fire as soon as it becomes enabled.

Time can also be associated with arcs [19]. In this case it is interpreted as a period of time that must elapse until a token will arrive from a place to a transition or vice versa. This representation is equivalent to representing time as the duration of firing a transition. Associating time with arcs and/or places instead of transitions simply changes the way in which a Petri net is interpreted. In any case (time associated with transitions, places or arcs) the semantics of a Petri net are defined in a similar way.

A wide range of extensions to the basic Petri net framework has been done to incorporate ill-known information about evolutions of the network [6, 7, 21, 27], most of which are outlined in [5]. These extensions include: fuzzy token locations, fuzzy firing rules (when transitions fire with a degree of possibility), fuzzy firing windows (during which transitions may fire), etc. In our approach, imprecise information is given about temporal properties of a dynamic system. In Petri nets these

properties are modeled by the timestamps associated with each token (absolute time of its generation) and durations of firing of the transitions.

IV. POSSIBILISTIC REPRESENTATION OF TIME

Dubois and Prade [13, 14] use possibility theory to model and manage temporal knowledge that involves imprecisely known information. This approach uses points as the temporal primitive. Imprecisely known dates are modeled by a fuzzy set with a unimodal possibility distribution over the temporal axis. Fuzzy temporal intervals are derived from fuzzy dates that are limiting the time span during which an event occurs. Fuzzy durations are treated in a similar way. A typical possibilistic temporal interval representing a fuzzy duration may be approximated by a trapezoidal shape (Figure 3). A temporal interval shown on Figure 3 has a duration certainly between a and b and most plausibly between a' and b'. Approach proposed by Dubois and Prade allows for

- representation of uncertain precedence relations (such as "much before" or "closely after") between events,
- comparison, ranking and ordering of fuzzy dates and durations,
- propagation of qualitative fuzzy temporal constraints.

The approach proposed by Dubois and Prade establishes a very solid foundation for applying possibility theory for temporal reasoning by providing a possibilistic representation of temporal primitives.

Trapezoidal approximations of possibilistic distributions does not provide enough flexibility for modeling of mathematical functions. They are overly restrictive to the resulting function and are hardly useful for piece-wise approximation. For the purposes of modeling functions by approximation and to minimize the number of elementary intervals participating in the approximation, it is proposed to represent possibilistic distributions using *alternative trapezoidal shapes* (Figure 4). It is different from the trapezoids studied earlier – alternative trapezoids have their vertical edges parallel to each other, lower edge coincides with the horizontal axis, and the last edge (slope) is arbitrary. Such an alternative trapezoidal shape T is determined by four parameters:

$$T = \{a, b, h_1, h_2\}.$$

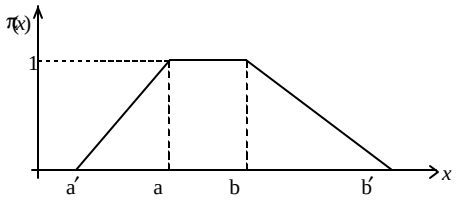

Figure 3. Trapezoidal approximation of a possibilistic interval

Depending on the inclination of the slope (i.e. the sign of $(h_1 - h_2)$) we will distinguish between L-trapezoidal ($h_1 < h_2$) and R-trapezoidal ($h_1 > h_2$) shapes. Instances of these trapezoidal shapes include rectangular shape ($h_1 = h_2$), L-triangular ($h_1 = 0$) and R-triangular ($h_2 = 0$) shapes.

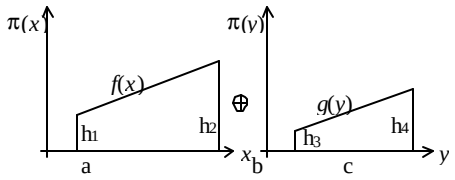

Figure 4. L-trapezoidal and R-trapezoidal possibilistic distributions

It is possible to approximate any function using only rectangular shapes, but better approximation will require a very fine quantification and therefore will result in many elementary rectangles used to approximate the function. Approximations using several trapezoidal shapes are closer to the real functions and require less elementary shapes participating in such approximations (Figure 5).

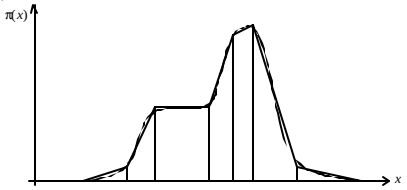

Figure 5. Approximation of a function using L- and R-trapezoidal shapes

An arbitrary function $f(x)$ representing a possibilistic distribution may be modeled as a disjunction of k elementary distributions $m_i(x)$.

$$f(x) = \bigcup_{i=1}^{k} m_i(x)$$

Consider a case of combining two trapezoids shown in Figure 4. It is easy to show that a simple formula exists and it can be derived from the basic principle of combining two possibilistic distributions

$$F(z) = f_i(x) \oplus g_j(y) = \sup(\min(f_i(x), g_j(y)) \mid x + y = z)$$

and given the analytical expressions for both functions $f_i(x)$ and $g_j(y)$. For each value of z there is an infinite

number of possible pairs of x and y, such that $x + y = z$. However, for each individual value of z we must select x and y such that $F(z)$ reaches it's maximal value.

Depending on the interrelationship among the height parameters h_1 through h_4, the resulting formula describes a piece-wise linear function resulting from a combination of $f_i(x)$ and $g_j(y)$. The specific parameters of such a formula depend only on the analytical expressions for $f_i(x)$ and $g_j(y)$ and on up to two additional parameters x_1 and x_2, which can be derived from the analytical expressions of $f_i(x)$ and $g_j(y)$.

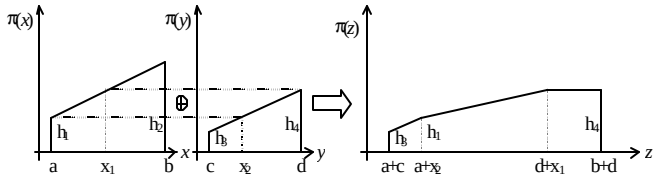

Figure 6. Example of combination of two trapezoidal distributions.

For example, Figure 6 shows two elementary trapezoidal distributions, interrelationships among height parameters h_1 through h_4 and parameters x_1 and x_2, which are the projection points of some height h_i onto the slope of the opposite trapezoid. Detailed studies and the techniques for deriving the analytical expressions for such combinations are presented in [22].

V. ALGORITHMS FOR TIMED PETRI NETS WITH POSSIBILISTIC TIME

Any Petri network executes by firing transitions. In the classical Petri network model, as well as in our approach, in each cycle not more than one transition fires. Figure 7 presents an algorithm of executing a Petri network. It runs in a loop until certain termination condition is met. It can be either a number of firings (cycles), a duration of the simulation, a condition when a token reaches certain place, etc. In each iteration of the loop, this algorithm calls procedure `FireTransition`, which finds an enabled transition and if such transition exists, fires it.

```
procedure PetriNet(PN, TC)
   input: Petri network PN;
   input: terminating condition TC;
   output: none;
begin
   while not TC do
      FireTransition(PN);
   end
end
```

Figure 7. Algorithm for execution of a Petri net

A transition may fire only if it is enabled, that is each of its input places has at least as many tokens in it as arcs from the place to the transition. Figure 8 presents algorithm `FireTransition` for firing a transition, which starts with finding all enabled transitions in the network. Enabled transitions are found according to the rule given above, which is taken directly from the applied Petri network theory.

When all enabled transitions are found, algorithm must determine which of them will fire. This is a source of non-determinism in the classical Petri net theory. In our approach we can use temporal information stored in the tokens for determining which enabled transition will fire. In order to do this, in each input places of all transitions we must find the token *MinToken* with the earliest timestamp. This gives the information when the first available token has arrived to a given place. The timestamps of these tokens will be used for the synchronization at the enabled transitions.

For each set of input places of each transition we find a maximum (a synchronized token) among previously found tokens *MinToken*s. This shows the earliest time when tokens are available in all input places of an enabled transition and therefore, the time when a transition becomes enabled to fire. By comparing these times for all enabled transitions, we can find what is the transition, which becomes the earliest enabled to fire. There are several ways how we can compare possibilistic distribution. On of them is comparing the leftmost boundary of each distribution. Another one involves calculating a median value of each distribution and comparing these medians

```
algorithm FireTransition(PN)
  input: Petri network PN
  output: Success or NoTransitionToFire
begin
  Token MinToken ← +Infinity
  Transition Firing ← ∅

  for all transitions T of PN do
    begin
      token MaxToken ← -1
      for all input places P of transition T do
        begin
          assume P is enabled
          if number of tokens in P < #(P, Input(T))then
            P is not enabled
          else if MaxToken < max(tokens in P) then
            MaxToken ← max(tokens in P)
        end
        if P is enabled and MinToken > MaxToken and
                    MinToken > enabling time of T then
          begin
            Firing ← T
            MinToken←max(MaxToken, enabling time of T)
          end
    end

  if Firing = ∅
    return NoTransitionToFire
  Token NewToken ← (MinToken + duration of Firing)
  update enabling time of Firing using NewToken
  for all input places P of Firing do
    remove minimal token from P
```

```
    for all output places P of Firing do
      add token NewToken into P
    return Success
  end
```

Figure 8. Algorithm of firing a transition in a possibilistic timed Petri net.

Thus, among all enabled transitions, the firing transition is determined as the one whose synchronized token is minimal. If there are no enabled transitions (and therefore no firing transition), algorithm `FireTransition` returns a failure. Otherwise, the selected transition fires. From all input places of the firing transition a minimal token is removed. Adding the duration of the firing transition to the synchronized token generates new token. This new token is added to all output places of the firing transition.

The algorithm for finding the firing a transition and firing it is extended as shown on Figure 8 to take into account that tokens are timestamped, transitions have durations, and they cannot fire again until a period of time passes that is equal to the duration of firing.

To illustrate the advantages of using possibility for representing temporal constrains, let us consider a solution of the above-described problem of modeling the automated manufacturing system. Assume that possibilistic representation of the operations' durations is such as shown in Table 1.

Duration of operation	Possibilistic distribution
Machine 1	[0.5min(0), 1min(1), 2min(1)]
Machine 2	[2min(1), 4min(1), 7min(0)]
Machine 3	[2min(0), 3min(1), 5 min(1), 8min(0)]
Machine 4	[3min(0), 6min(1), 7min(1), 10min(0)]
Robot 1	[0.5min(0), 1min(1), 1.5min(0)]
Robot 2	[0.5min(0), 1min(1), 1.5min(0)]

Table 1. Numeric data for durations of operations of the automated manufacturing system problem

The graph of the corresponding Timed Petri net and the configuration of the initial marking is presented in Figure 9. Assume the numeric values of the initial marking as shown in Table 2 (Figure 9 shows only locations of the tokens, but not the possibilistic distributions associated with them).

Token	Located in place	Possibilistic distribution of the token
t_1	p_1	[0min(0), 1min(1), 2min(1), 3min(0)]
t_2	p_1	[0min(0), 1min(1), 2min(1), 3min(0)]
t_3	p_5	[0min(0), 2min(1), 4min(1), 6min(0)]

Table 2. Numeric data for the initial marking of the automated manufacturing system problem

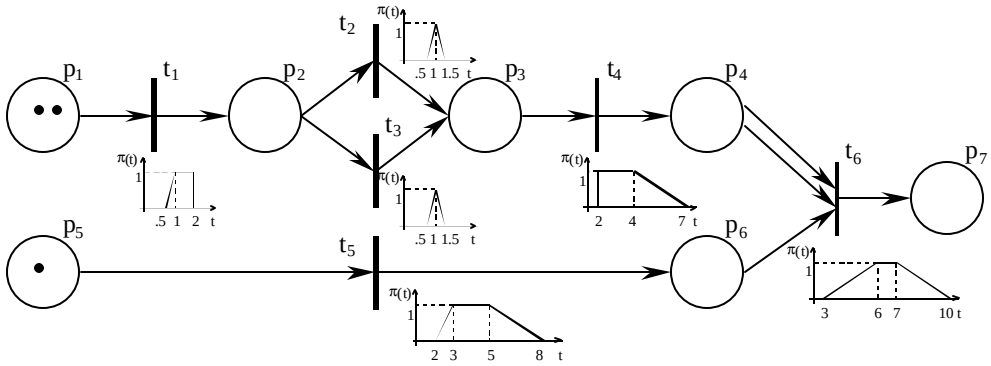

Figure 9. Timed Petri net graph for the automated manufacturing system

Tokens in place p_1 indicates that two pieces of part A will be available for machine 1 at any time between 0 and 3 minutes, but most plausibly between 1 and 2 minutes. Token in place p_5 indicates that machine 3 will receive part B at any time between 0 and 6 minutes, but most plausibly between 2 and 4 minutes.

Comparing the results with interval and possibilistic representation of time, it is obvious that the second result with possibilistic temporal intervals expresses much more information providing the most plausible interval for the final product to be assembled.

VI. CONCLUSIONS

In this paper, we have considered how time can be incorporated into the framework of Petri nets. Throughout the paper, we used an example of an automated manufacturing system that was used to show advantages and disadvantages of different approaches to representing time in Petri nets. The duration of a task, which is represented by a possibilistic distribution, is modeled as a transition in the extended Petri network. The tokens represent the completion time of the tasks and hence they are associated with possibilistic

distributions. In a typical Petri network, synchronization or to determining the enabling time of a transition is quite easy. We provide an elegant solution to enabling a transition with incoming tokens with possibilistic distributions. Using an example we have clearly shown the expressiveness of this approach and its advantage over other approaches.

REFERENCES

[1] J. Allen. Maintaining Knowledge about Temporal Intervals. *Communications of the ACM*, 26:832-843, 1983.

[2] I. Bestuzheva, V. Rudnev. Timed Petri Nets: Classification and Comparative Analysis. *Automation and Remote Control*, 51(10):1308-1318, Consultants Bureau, New York, 1990.

[3] C. Brown, D. Gurr. Temporal Logic and Categories of Petri Nets. In A. Lingass, R. Karlsson, editors, *Automata, Languages and Programming*, pp. 570-581, Springer-Verlag, New York, 1993.

[4] J. Carlier, P. Chretienne. Timed Petri Net Schedules. In G. Rozenberg, editor, *Advances in Petri Nets*, pp. 642-666, Springer-Verlag, New York, 1988.

[5] J. Cardoso, H. Camargo, editors. *Fuzziness in Petri Nets*, Physica Verlag, New York, NY, 1999.

[6] J. Cardoso, R. Valette, D. Dubois. Fuzzy Petri Nets: An Overview. In G. Rosenberg, editor, Proceedings of the 13th IFAC World Congress, pp. 443-448, San Francisco CA, 30 June - 5 July 1996.

[7] J. Cardoso, R. Valette, D. Dubois. Possibilistic Petri nets. *IEEE transactions on Systems, Man and Cybernetics*, part B: Cybernetics, October 1999, Vol. 29, N 5, p. 573-582

[8] J. Coolahan. N. Roussopoulos. Timing Requirements for Time-driven Systems Using Augmented Petri Nnets. *IEEE Transactions on Software Engineering*, 9(5):603-616, 1983.

[9] R. Dechter, I. Meiri, J. Pearl. Temporal Constraint Networks. *Artificial Intelligence*, 49:61-95, 1991.

[10] M. Diaz, P. Senac. Time Stream Petri Nets: a Model for Timed Multimedia Information. In R. Valette, editor, *Application and Theory of Petri Nets-94*, pp. 219-238, Springer-Verlag, New York, 1994.

[11] D. Dubois, H. Prade. *Possibility Theory*. Plenum Press, New York, 1988.

[12] D. Dubois, H. Prade. Processing Fuzzy Temporal Knowledge. *IEEE Transactions on Systems, Man and Cybernetics*, 19(4), July/August 1989.

[13] D. Dubois, J. Lang, H. Prade. Timed Possibilistic Logic. *Fundamenta Informaticae*. 15(3,4):211-234, 1991.

[14] D. Dubois, H. Prade. Processing Fuzzy Temporal Knowledge. *IEEE Transactions on Systems, Man and Cybernetics*, 19(4):729-744, 1989.

[15] M. Felder, A. Morzenti. A Temporal Logic Approach to Implementation and Refinement of Timed Petri Nets. In D. Gabbay, editor, Proceedings of *1st international conference on Temporal Logic ICTL-94*, Bonn, Germany, July 11-14, pp. 365-381, Springer-Verlag, New York, 1994.

[16] P. Fortemps. Jobshop Scheduling with Imprecise Durations: A Fuzzy Approach. *IEEE Transactions on Fuzzy Systems*, 5(4):557-569, 1997.

[17] C. Ghezzi, D. Mandrioli, S. Morasca, P. Mauro. A General Way to Put Time into Petri Nets. *ACM SIGSOFT Engineering Notes*, 14(3)-60-67, Pittsburgh, Pennsylvania, 1989.

[18] L. Godo, L. Vila. Possibilistic Temporal Reasoning Based on Fuzzy Temporal Constraints. In C. Mellish, editor, *Proceedings of IJCAI-95*, pp. 1916-1922, Montreal, Canada, 20-25 August, Morgan Kaufmann, San Francisco, CA, 1995.

[19] H. Hanisch. Analysis of Place/Transition Nets with Timed Arcs and Its Application to Batch Process Control. In M. Marsan, editor, *Application and Theory of Petri Nets-93*, pp. 282-299, Springer-Verlag, 1993.

[20] E. Kindler, T. Vesper. ESTL: A Temporal Logic for Events and States. In J. Desel, M. Silva, editors, *Application and Theory of Petri Nets-98*, pp. 365-384, Springer-Verlag, New York, 1998.

[21] L. Kunzle, R. Valette, B. Pradin-Chezalviel. Temporal Reasoning in Fuzzy Time Petri Nets. In J. Cardoso, H. Camargo, editors, Fuzziness in Petri Nets, pp. 146-173, Physica Verlag, New York, NY, 1999.

[22] S. Kurkovsky. Possibilistic Temporal Propagation. Ph.D. dissertation. University of Southwestern Louisiana, 1999.

[23] P. Merlin, D. Farber. Recoverability of Communication Protocols. *IEEE Transactions on Communications*, 24(9):541-580, 1989.

[24] J. Peterson. *Petri Net Theory and The Modeling of Systems*. Prentice Hall, 1981.

[25] C. Ramamoorthy, G. Ho, Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets. *IEEE Transactions on software Engineering*, 6(5):440-449, 1980.

[26] M. Tanabe. Timed Petri Nets and Temporal Linear Logic. In P. Azema, G. Balbo, editors, *Application and Theory of Petri Nets-97*, pp. 156-174, Springer-Verlag, New York, 1997.

[27] M. Woo, N. Qazi, A. Ghafoor. A Synchronization Framework for Communication of Pre-orchestrated Multimedia Information. *IEEE Networks*, 8(1)52-61, 1994.

[28] Y. Yao. A Petri Net Model for Temporal Knowledge Representation and Reasoning. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(9):1374-1382, 1994.