Modeling of, and Reasoning with Recurrent Events with Imprecise Durations

Stanislav Kurkovsky[1] and Rasiah Loganantharaj[2]

Department of Computer Science[1], Columbus State University,
kurkovsky_stan@colstate.edu
Automated Reasoning Laboratory, Center for Advanced Computer Studies[2]
University of Louisiana at Lafayette, logan@cacs.usl.edu

Abstract. In this paper we study how the framework of Petri nets can be extended and applied to study recurrent events. We use possibility theory to realistically model temporal properties of the recurrent processes being modeled by an extended Petri net. Such temporal properties include time-stamps stored in tokens and durations of firing the transitions. We apply our method to model the recurrent behavior of an automated manufacturing cell.

1. Introduction

Many events occurring in the real world are of repetitive in nature, that is, an event occurs more than once over a span of time. There are numerous attempts to study recurrent events in terms of qualitative relations among events and tasks. It is possible to have infinite relations among tasks in repetitive events [21]. In this paper, we focus on a repetitive event, in which the task sequence satisfies the given quantitative constraints of the event. While the sequence of tasks in the repetitive event remains the same, the duration of each repetition may differ since each task in each cycle may take different time to complete due to availability of different equipment, environmental factors, etc. The impreciseness of the task duration must be modeled. Further, we need a formal mechanism to model sequence of tasks or to maintain the partial order constraints imposed by the event. Once each task is appropriately represented with imprecise duration and we have a formal mechanism to maintain the task precedence relation in an event, we need to reason with the duration for a given specified cycle of the repetitive events. In this paper, we use fuzzy logic to represent imprecise duration and extend Petri net framework to maintain precedence requirements among the tasks and as well as to reason with distance among repetitions of events.

The paper is organized as follows. We provide the background information on Petri nets and representing recurrent events in section 2, which is followed by a review on timed Petri nets. In section 4, we describe possibilistic representation of time, which is followed by presenting algorithms for timed Petri nets with possibilistic time. Section 6 presents the results of experiments on modeling a system exhibiting a recurrent behavior. In section 7 we summarize the conclusion of this paper.

2. Background Information of Petri Net Theory

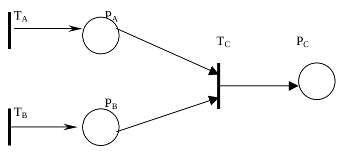

Fig. 1. Petri net graph

Ever since the introduction of the Petri net theory by Petri in his Ph.D. dissertation, it was widely used for modeling dynamic systems in the most diverse domains. Petri net is a graph-based structure consisting of places and transitions. Each transition is connected by, and connected to at least one place. When modeling tasks using Petri network, each place that connects to a transition corresponds to a pre-condition and the transition corresponds to a task. When a token is in a place, then the precondition denoted by the place is said to be *satisfied*. Consider a partial order of tasks A < C, B < C (the tasks A and B must be completed before C). When transition T_A fires (task A completes), it generates a token and it is placed in P_A. Similarly, when transition T_B fires (task B completes), it generates a token and it is placed in P_B. When there is at least one token in both P_A and P_B, the transition T_C is *enabled*. Thus the following Petri network models the partial order of tasks A<C, B<C (Figure 1).

Modeling Non-deterministic Behavior

Let us use an automated manufacturing cell shown in Figure 2 to illustrate how to model non-deterministic behavior of a dynamic system. The cell is composed of three machines, a robot and, input (*In*) and output (*Out*) buffers. Parts are coming into the cell and are stored in buffer *In* of unlimited capacity. Parts can be processed by any one of the three machines (determined non deterministically), namely, *Machine 1 (M_1)*, *Machine 2 (M_2)*, or *Machine 3 (M_3)*. Each processed part is removed from the buffer of the respective machine and is placed in buffer *Out* of unlimited capacity. *Robot* transfers parts between the buffers and machines. *Robot* is non-preemptive and at each moment it can be used to either load a part from buffer *In* to any of the machines, or to unload a processed part from any of the machines to buffer *Out*.

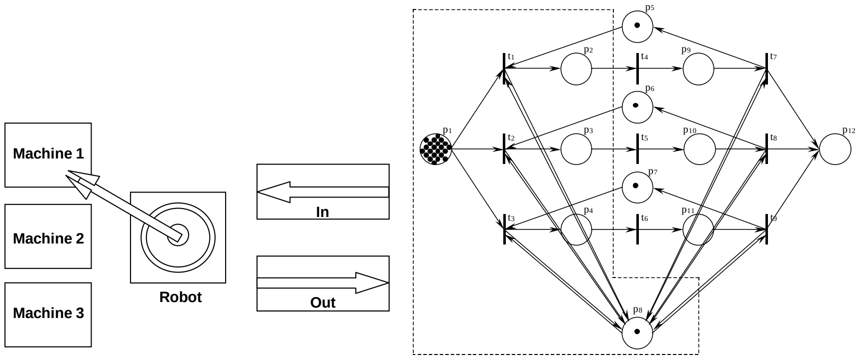

Fig. 2. An automated manufacturing cell and the corresponding Petri net graph.

A cyclic event has the following tasks: the robot arm picks a part from the In buffer and then transfers it to the buffer of one of the three machines, M_1, M_2, and M_3. Once the part is processed by the selected machine, the robot picks the part and leaves it in the output buffer. Let us consider a portion of the Petri net that models the robot picking a part from the In buffer and leaving it in the buffer of any one of the three machines. Once the part is placed in the buffer of a machine, the robot becomes free for other activities. The availability of a part in the *In* buffer is modeled as a token placed in p_1. Similarly, the availability of the robot is modeled by placing a token in place p_8. The transitions t_1, t_2 and t_3 respectively represent activity of placing a part in the buffer of machines 1, 2 and 3. Initially the three transitions, t_1, t_2 and t_3 are enabled, but only one can be fired at a time. When more than one transition is enabled, a transition from the enabled ones is selected either randomly or using some heuristic bias. This is how one models non-deterministic process in Petri network model. Once a transition is fired, a token from each incoming place is removed and a token is placed to each outgoing place. When t_2 fires, a pair of tokens, one from p_1 and another from p_8 are removed and a token is placed in each outgoing places (p_3 and p_8). The problem is somewhat complicated if we impose a constraint that the buffer size of machines 1, 2 and 3 is exactly one.

Modeling Recurrent Events

We have described the tasks that occur in an instance of the event. It an event repeats continually, the robot is free to take parts and place it in an empty buffer of a machine. The robot then performs either one of the following activities: (1) removes the processed part from the buffer of a machine and then places it in the *Out* buffer, or (2) if there is at least one machine ready for processing a part, the robot removes a part from *In* buffer and place it in the buffer of a machine that is ready. To model this continuous repeated activity, the *In* buffer is initialized with unlimited number of tokens. The Petri model shown inside of the contoured area in Figure 2 correctly models the repeated event of processing parts by the robot.

3. Extensions to Petri Nets to Incorporate Time

Qualitative notion of time is implicitly embedded into the Petri net framework. Only one firing of a transition can occur at a time and it is associated with one clock cycle or tick of a certain internal clock. 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 every component of a Petri net [3, 19].

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, 27, 28]. 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, 27]. Possible interpretations of time θ is associated with transition t_j include: t_j may (or must) fire only after θ time units pass after t_j becomes enabled (θ is relative); t_j may fire only during time interval θ (θ is absolute); the duration of firing of t_j is θ (θ is assumed to be relative). In this case t_j may (or must) start firing as soon as it becomes enabled.

Waiting time can be associated with places [8, 28]. Once a 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. Time can also be associated with arcs [18]. 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.

4. 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 a temporal primitive. Imprecisely known dates are represented as fuzzy sets 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. A typical possibilistic temporal interval representing a fuzzy duration may be approximated by a trapezoidal shape.

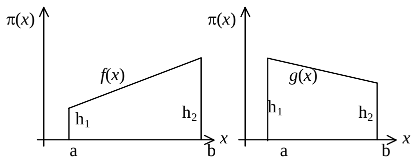

Fig. 3. L-trapezoidal and R-trapezoidal possibilistic distributions

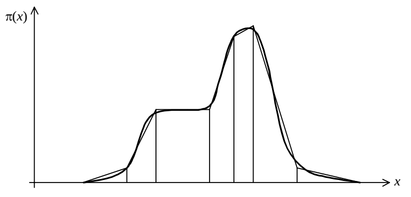

Fig. 4. Approximation of a function using L- and R-trapezoidal shapes

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, however, 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. To approximate functions and to minimize the number of elementary intervals participating in the approximation, it is proposed to represent possibilistic distributions using *alternative trapezoidal shapes* (Figure 3), which 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}. 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 $=0$) and R-triangular ($h_2=0$) shapes.

It is possible to approximate any arbitrary 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 4). 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 5. It is easy to show that a simple formula exists and it can be derived from the basic principle of combining two possibilistic distributions

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

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.

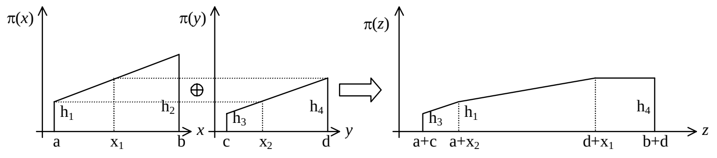

Fig. 5. Example of combination of two trapezoidal distributions.

For example, Fig. 5. 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 [20].

5. Algorithms for Timed Petri nets with Possibilistic Time

```
procedure PetriNet(PN, TC)
   input: Petri network PN;
   input: term. cond TC;
begin
   while not TC do
      FireTransition(PN);
   end
end
```

Fig. 7. Algorithm for execution of a Petri net

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 that executes a Petri net. It iterates 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. At each iteration, this algorithm calls procedure FireTransition, which finds an enabled transition and, if such a transition exists, fires it.

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

```
      token MaxToken ← -1
      for all input places P of transition T 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
         Firing ← T
         MinToken ← max(MaxToken, enabling time of T)
      end if
   end
   if Firing = ∅ then return NoTransitionToFire
   Token NewToken ← (MinToken + duration of Firing)
   update enabling time of Firing using NewToken
   for all input places P of Firing
      remove minimal token from P
   for all output places P of Firing
      add token NewToken into P
   return Success
end
```

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

A transition may fire only if it is enabled – when each of its input places has at least as many tokens in it as arcs from the place to the transition. Figure 8 presents an algorithm 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 each input places of all transitions we must find token *MinToken* with the earliest timestamp. This tells when the first available token has arrived to a given place. The timestamps of these tokens will be used for synchronizing 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. Thus, the firing transition is determined as an enabled transition whose synchronized token is minimal. If there are no enabled transitions the algorithm 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 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.

6. Experimental Modeling of Recurrent Events with Petri Nets

The following sections describe different simulation experiments performed on the automated manufacturing cell described earlier with different temporal characteristics. The simulation is performed as an iterated execution of the algorithm that fires a transition in a possibilistic timed Petri net (Figure 8). *Simulated time* is reflected in the timestamps of tokens populating the network and enabling times of the transitions. Each experiment is conducted until certain simulated *target time* is reached (1000 minutes in all experiments). Simulation continues for a pre-specified significantly large number of firings that achieves exceeding the target time. Temporal information about firing of each transition is recorded in *log files*, which are then processed and analyzed to build the throughput and utilization graphs of each entity of the Petri net.

The goal of these experiments is to study how temporal characteristics of different entities change the degree of non-determinism and how that affects the behavior of the cell. We will analyze the utilization and throughput of each individual entity of the automated manufacturing cell and the cell as a unit during the simulation of its operation over a significantly long period of the simulation time (1000 minutes). In Experiment 1 processing times of each machine are identical, as well as the robot's serving times. In Experiment 2 machine processing times vary insignificantly, while the robot's serving times are identical for each machine. In Experiment 3 machine processing times vary significantly and are comparable to the robot's serving times, which are the same as in the previous experiment. Based on the results of each of these experiments, graphs of utilization and throughput of each entity of the cell will be built. The throughput graphs are expected to grow linearly for the cell as a unit and for each individual entity. After some initial period of the simulated time, the utilization is expected to tend to certain constant value for each entity of the cell.

Table 1. Numeric data for Experiment 1.

Transition	**Possibilistic temporal distribution**
t_1	[30sec(1)]
t_2	[30sec(1)]
t_3	[30sec(1)]
t_4	[4min(0), 5min(1), 6min(0)]
t_5	[4min(0), 5min(1), 6min(0)]
t_6	[4min(0), 5min(1), 6min(0)]
t_7	[30sec(1)]
t_8	[30sec(1)]
t_9	[30sec(1)]

Experiment 1.

Table 1 presents the numeric data for experiment 1 (durations of operations of the automated manufacturing cell in the form of possibilistic temporal distributions). Durations of all robot's operations are fixed at 30 seconds. Durations of operations of all three machines are also identical (between 4 and 6 minutes, but most likely 5 minutes). Figure 9 shows that the throughput of all machines, robot, and the entire cell linearly increases over time, as expected. Also, throughput graphs of all three machines coincide due to their identical temporal characteristics. Figure 10shows the utilization of the machines and robot. After some introductory period (about 300 minutes) the utilization graphs of all three machines stabilize at the same value due to the identical temporal characteristics of all machines.

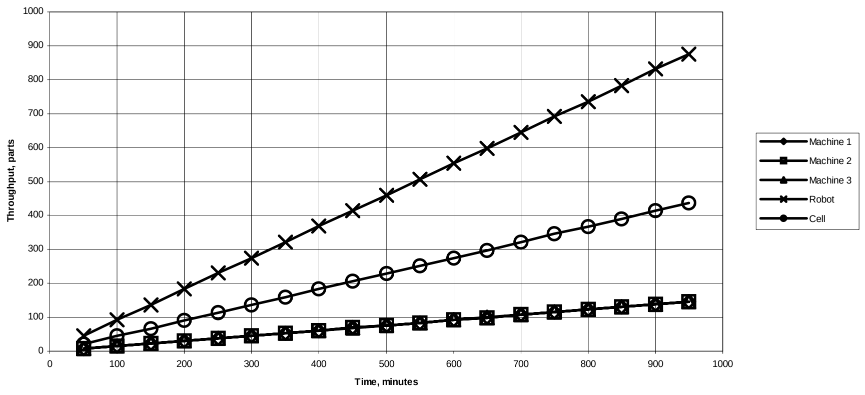

Fig. 9. Throughput of the machines, robot, and the entire cell in Experiment 1.

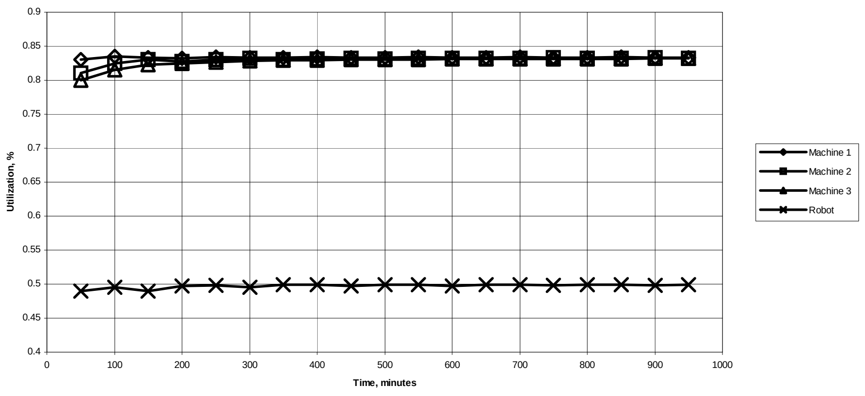

Fig. 10. Utilization of the machines and robot in Experiment 1.

Table 2. Numeric data for Experiment 2.

Transition	**Possibilistic temporal distribution**
t_1	[20sec(1)]
t_2	[30sec(1)]
t_3	[40sec(1)]
t_4	[3min(0), 4min(1), 5min(0)]
t_5	[4min(0), 5min(1), 6min(0)]
t_6	[5min(0), 6min(1), 7min(0)]
t_7	[40sec(1)]
t_8	[30sec(1)]
t_9	[20sec(1)]

Experiment 2.

Table 2. presents the numeric data for experiment 2. Durations of robot's operations vary from 20 to 40 seconds. Durations of operations of the machines vary, but they are significantly larger than the durations of operations of the robot. Figure 11 shows that the throughput of the machines, robot, and the entire cell linearly increases over the time, as expected. Unlike the previous experiment, the throughput graphs of the machines do not coincide due to the different temporal characteristics of the machines. Figure 12 shows the utilization of the machines and robot. After some introductory period (approximately 300 minutes) the utilization graphs of all three machines and the robot stabilize at the some fixed values.

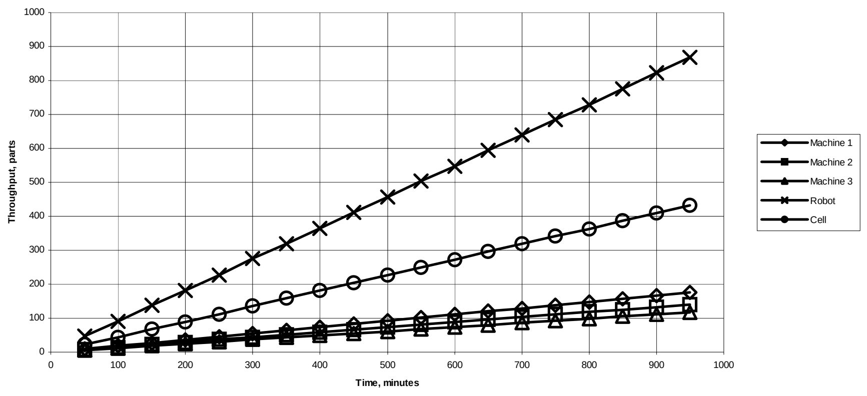

Fig. 11. Throughput of the machines, robot and the entire cell in Experiment 2.

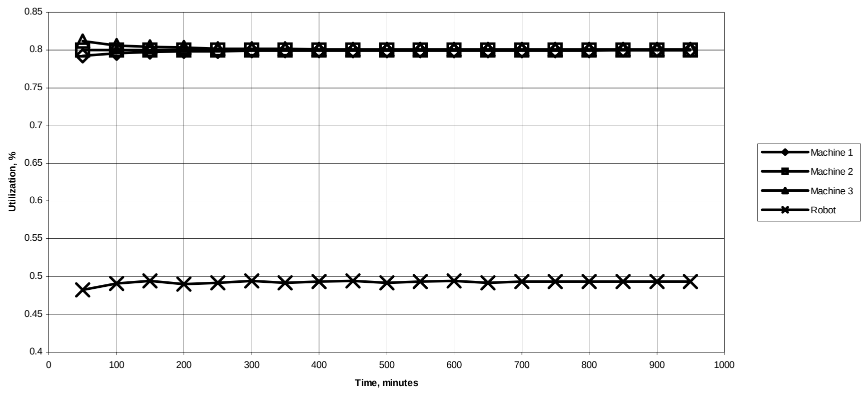

Fig. 12. Utilization of the machines and robot in Experiment 2.

Table 3. Numeric data for Experiment 3.

Transition	Possibilistic temporal distribution
t_1	[1min(1)]
t_2	[1min(1)]
t_3	[1min(1)]
t_4	[2min(0), 3min(1), 4min(0)]
t_5	[4min(0), 5min(1), 6min(0)]
t_6	[6min(0), 7min(1), 8min(0)]
t_7	[30sec(1)]
t_8	[30sec(1)]
t_9	[30sec(1)]

Experiment 3.

Table 3 presents the numeric data of experiment 3 for durations of operations of the automated manufacturing cell in the form of possibilistic temporal distributions. Durations of the robot's transferring parts to the machines are fixed at 1 minute; durations of the robot's transferring parts from the machines are fixed at 30 seconds. Durations of the operations of the machines differ and they are comparable with the durations of the operations of robot. Figure 13 shows that the throughput of the machines, robot, and the entire cell linearly increases over the time, as expected. Similarly to the previous experiment, the throughput graphs of the machines do not coincide due to the different temporal characteristics of the machines. Figure 14 shows the utilization of the machines and robot. After some introductory period (approximately 300 minutes) the utilization graphs of all three machines and the robot stabilize at the some fixed values.

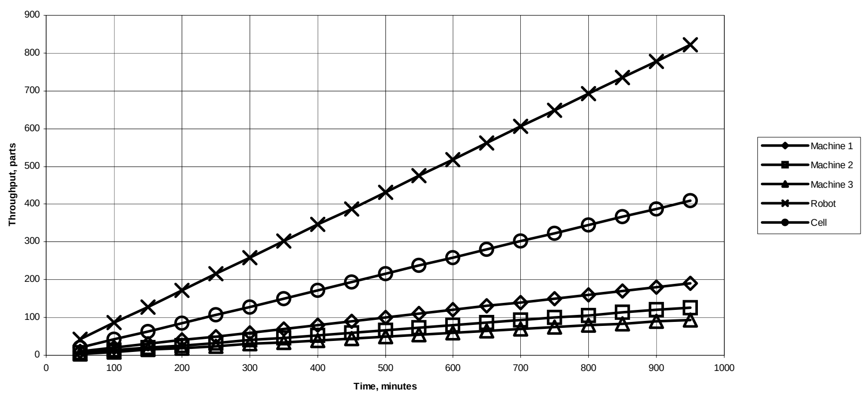

Fig. 13. Throughput of the machines, robot, and the entire cell in Experiment 3.

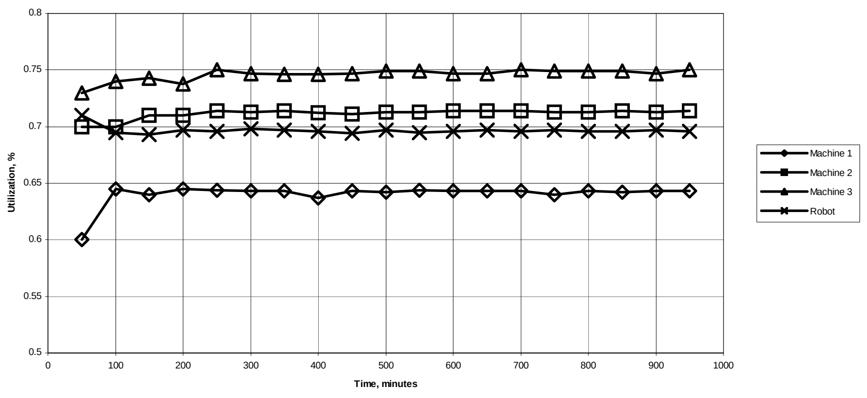

Fig. 14. Utilization of the machines and robot in Experiment 3.

The results of all three experiments coincide with the expected outcomes: utilization of each entity of the cell grows linearly and throughput stabilizes at a certain value after some initial time.

7. Conclusions

In this paper, we have shown how to model and reason with repetitive events using Petri nets. In general, the task sequence of an event may not be preserved in the repeated occurrence of the same event. Worse yet, the set of tasks occurring in an event may not be the same in the repeated occurrence of the same event. Petri network seems to be an ideal mechanism to model such complicated repetitive events; non-deterministic choices can be modeled elegantly in Petri network. In spite of all these advantages, Petri network does not have a well-defined notion of time. There are, however, number of ad-hoc works done to incorporate time with Petri network, which range from associating time with token to the extend of associating time with arcs. Nevertheless, none of these approaches uses time to represent imprecise durations of transitions or represents the marking of the network as a set of tokens timestamped with imprecise times of their creation. This approach adopted in our framework presents a way for a more realistic modeling of the real world problems.

We use an automated manufacturing cell as a running example throughout the paper. The duration of each 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 determining of 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, its advantage over other approaches and its applicability to model recurrent systems. Results of the experiments conducted with the created model coincide with the expected outcomes: throughput is a linear function of time and utilization tends to a certain constant. These facts allow us to conclude that our approach is valid for modeling real world problem where temporal properties, such as durations of tasks, can be modeled with a degree of imprecision.

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

[5] 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.

[6] 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

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

[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] 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.

[18] 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.

[19] 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.

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

[21] P. Ladkin. Time Representation: A Taxonomy of Interval Relations. *Proceedings of fifth national conference on Artificial Intelligence*, pp. 360-366. American Association for Artificial Intelligence, 1996.

[22] R. Loganantharaj, S Giambrone. Probabilistic Approach for Representing and Reasoning with Repetitive Events. In J. Stewman, editor, *Proceedings of FLAIRS-95*, pp. 26-30, Melbourne, FL, 27-29 April 1995.

[23] R. Loganantharaj, S. Giambrone. Representation of, and Reasoning with, Near-Periodic Recurrent Events. In F. Anger, H. Guesgen, J. van Benthem, editors, *Proceedings of IJCAI-95 Workshop on Spatial and Temporal Reasoning*, pp. 51-56, Montreal, Canada, 20-25 August 1995, Morgan Kaufmann, San Mateo, CA, 1995.

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

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

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

[27] 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.

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

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