33-rd Midwest Symposium on Circuits and Systems (MSCAS'90). Calgary, Canada, 12-14 August 1990, pp.774-778.
Copyright © 1990 IEEE (DOI 10.1109/MWSCAS.1990.140836).

TRANSFORMATIONS OF TIMED PETRI NETS AND PERFORMANCE ANALYSIS

W.M. Zuberek and M.S. Zuberek

Department of Computer Science, Memorial University
St.John's, NL, Canada A1C–5S7

A b s t r a c t

There are two basic approaches to analysis of timed Petri net models, the so called reachability analysis and structural analysis. Reachability analysis is based of the space of reachable states while structural analysis derives properties of models from properties of model components and component interconnections. This paper discusses several simple transformations of timed nets that significantly simplify performance analysis preserving all important performance properties of the original model. In particular, they can convert a model that can be analyzed by the reachability approach only into an equivalent one (with respect to performance) that can be analyzed by structural methods.

1. INTRODUCTION

The increasing availability of inexpensive processors as well as computer networks results in a trend to distribute functions of information processing systems into different processes and processors. This, however, implies a significant growth in complexity of coordination in such systems, which is particularly acute for the interactions or protocols that specify how distributed processes or processors are synchronized and how they communicate with one another. Formal methods are gradually being developed to describe such interactions [Boch,Dan,ZWCB], and Petri nets [Pet,Reis] proved to be a convenient example of such formalisms [CAA,Dan].

Petri nets have been successfully used in modelling [Diaz,MeFa], validation [BeTe,Dan] and analysis [CAA,Diaz,Raz] of systems of events in which it is possible for some events to occur concurrently, but there are constraints on the occurrence, precedence, or frequency of these occurrences. Basic Petri nets, however, are not complete enough for the performance studies since no assumption is made on the duration of systems events. Timed Petri nets have been introduced by Ramchandani [Ram] by assigning firing times to the transitions of Petri nets (t–timed nets), and by Sifakis [Sif] by assigning time to places of a net (p–timed nets). Merlin and Farber [MeFa] discussed timed Petri nets where a time threshold and maximum delay were assigned to each transition of a net to allow the incorporation of timeouts into protocol models. Berthomieu and Menasche [BeMe] used "state classes" to obtain finite representation of behavior of nets defined by Merlin and Farber; their description is sufficient for validation and verification studies, but it requires further refinements for performance analysis since no distribution of firing times is assumed. Razouk [Raz] and Razouk and Phelps [RaPh] discussed an interesting class of timed Petri nets with enabling as well as firing times (p & t–timed nets), and derived performance expressions for simple communication protocols.

This paper describes timed Petri nets augmented by "interrupt" arcs which can "cancel" initialized firings of transitions, as required in strict modelling of timeouts. Similarly as in [Ram,Z85,Z87], constant (or deterministic) firing times are assigned to transitions of a Petri net, and then the "state space" of a net is a discrete–space discrete–time homogeneous semi–Markov process. Standard techniques derived for analysis of Markov chains can thus be used to derive many performance measures such as utilization of systems components, average waiting times and turnaround times or average throughput rates.

In many cases the original modelling nets can be simplified by net transformations which do not change the behavior, i.e., state space, of the original net. It is shown that even a small number of such transformations may convert a modelling net to a very simple net with well–known properties, for example to a net covered by invariants [Reis], and then many properties (such as liveness and boundedness) can be determined without studying the space of reachable states.

The paper is organized in 4 main sections. Sections 2 and 3 recall some definitions of basic concepts for (extended) free–choice ordinary and D–timed Petri nets, respectively (more details can be found in [Z87]). Equivalent net transformations are introduced in section 4, while section 5 shows application of net transformation to modelling and analysis of a simple communication protocol.

2. MARKED PETRI NETS

An inhibitor Petri net $\mathbf{N}$ is a quadruple $\mathbf{N}=(P,T,A,B)$ where:

P is a finite, nonempty set of places,

T is a finite, nonempty set of transitions,

A is a set of directed arcs which connect places with transitions and transitions with places such that for each transition there is at least one place connected with it,

B is a (possibly empty) set of inhibitor arcs which connect places with transitions, $B \subset P \times T$; A and B are disjoint sets.

A marked Petri net $\mathbf{M}$ is a pair $\mathbf{M}=(\mathbf{N}, m_0)$ where:

$\mathbf{N}$ is an inhibitor Petri net, $\mathbf{N}=(P,T,A,B)$,

m_0 is an initial marking function which assigns a nonnegative integer number of so called tokens to each place of the net, $m_0 : P \to \{0,1,...\}$.

Let any function $m : P \to \{0, 1, ...\}$ be called a marking in a net $\mathbf{N}=(P, T, A, B)$.

A transition t is enabled by a marking m iff every input place of this transition contains at least one token and every inhibitor place of t contains zero tokens. The set of all transitions enabled by a marking m is denoted by $En(m)$.

A place p is shared iff it is an input place for more than one transition. In inhibitor nets, a shared place p is guarded iff for each two different transitions t_i and t_j sharing p there exists another place p_k such that p_k is in the input set of one and in the inhibitor set of the other of these two transitions, i.e., no two transitions from the output set of p can be enabled by the same marking.

A shared place p is free–choice (or extended free–choice) iff the input sets and inhibitor sets of all transitions sharing p are identical. An inhibitor net is free–choice iff all its shared places are either free–choice or guarded. Only free–choice nets are considered in this paper since in most cases free–choice nets are sufficient for modelling of random events, e.g., random faults in communication networks or random services with discrete distributions (another class of timed Petri nets is used for random events with continuous distributions [AMCB,Z86]).

Since the relation of sharing a free–choice place is an equivalence relation in T, it determines a partition of T into a set of free–choice equivalence classes denoted by $Free(T) = \{T_1, T_2, ..., T_k\}$.

Every transition enabled by a marking m can fire. When a transition fires, a token is removed from each of its input places (but not inhibitor places) and a token is added to each of its output places. This determines a new marking in a net, a new set of enabled transitions, and so on.

A set $M(\mathbf{M})$ of reachable markings of a marked Petri net $\mathbf{M}=(\mathbf{N}, m_0)$ is the set of all markings which are reachable from the initial marking m_0.

In analysis of timed nets it appears very convenient to have a concise notation that indicates all possibilities of firing transitions for a given marking m. A selection function g of a marking m in a net $\mathbf{N}$ is any function $g : T \to \{0, 1, ...\}$ which indicates (by nonzero values) all those transitions which can simultaneously initiate their firings (and some transitions may initiate their firings "several times"). The set of all selection functions of a marking m is denoted by $Sel(m)$.

Example. The Petri net shown in Fig.1 (as usual, places are represented by circles, transitions by bars, inhibitor arcs by arcs with small circles instead of arrowheads, and the initial marking function is indicated by dots inside places) contains one inhibitor arc (p_5, t_5), one free–choice place p_3, and one guarded place p_4. It should be observed that the set $M(\mathbf{M})$ is infinite since the sequence of firing transitions $(t_1, t_5, t_1, t_5, ...)$ can be continued "for ever" creating consecutive markings $(m_1, m_3, m_7, ...)$ with increasing number of tokens in p_2. Similarly, the cyclic sequence of firing transitions $(t_1, t_2, t_4, t_1, ...)$ increases the number of tokens in p_5. Consequently, the net is unbounded. □

3. D–TIMED PETRI NETS

In timed Petri nets each transition takes a "real time" to fire, i.e., there is a "firing time" associated with each transition of a net. The firing times can be defined in several ways. In D–timed Petri nets [Z85,Z88] they are deterministic (or constant), i.e., there is a nonnegative number assigned to each transition of a net which determines the duration of transition's firings. In M–timed Petri nets [Z86] (or stochastic Petri nets [AMCB]), the firing times are exponentially distributed random variables, and the corresponding firing rates are assigned to transitions of a net.

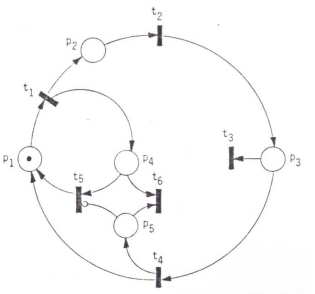

Fig.1. Inhibitor Petri net $\mathbf{M}_1$.

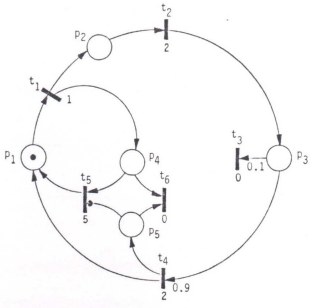

Fig.2. D-timed Petri net $\mathbf{T}_1$.

Since in timed Petri nets the firings of transitions are not instantaneous events, the concept of inhibitor arcs can be generalized to cover the timed behavior of transitions. The "proper" inhibitor arcs affect the transitions only at the beginning of their firings because they participate in enabling of transitions. The generalized inhibitor arcs, called interrupt arcs, affect a transition not only at the beginning of firing, but also during its firing; they can "interrupt" firing transitions and preempt the "resources" acquired at the beginning of firing. Interrupt arcs are necessary to model preempting scheduling disciplines, to represent properly timeout mechanisms, or to model unreliable processors which can "fail" during processing of user jobs. In some cases such interrupts and preemptions can be represented by inhibitor nets [Zb85], but usually such models and their behavior become unnecessarily complicated.

An extended D–timed free–choice Petri net $\mathbf{T}$ is a triple $\mathbf{T}=(\mathbf{M}, c, f)$ where:

- $\mathbf{M}$ is an extended free–choice marked Petri net, $\mathbf{M}=(\mathbf{N}, m_0)$, $\mathbf{N}=(P, T, A, B, C)$, and C is a set of interrupt arcs, $C \subset B$,
- c is a choice function which assigns a "free–choice" probability to each transition t of the net in such a way that for each free–choice equivalence class the sum of these probabilities is equal to 1,
- f is a firing time function which assigns a nonnegative real number $f(t)$ to each transition t of the net, $f : T \to \mathbf{R}^{\oplus}$ and $\mathbf{R}^{\oplus}$ denotes the set of nonnegative real numbers.

In ordinary nets (i.e., nets without time), interrupt arcs are equivalent to inhibitor arcs since the firings of enabled transitions are instantaneous events. In extended timed Petri nets, the firing of a transition may be "discontinued" by any one of interrupt arcs associated with this transition. If, during a firing period of a transition t, one of places connected with t by interrupt arcs becomes nonempty (i.e., it receives at least one token), the firing of t ceases and the tokens removed from t's input places at the beginning of firing are "returned" to their original places.

Moreover, an extended Petri net is simple if the input sets of transitions with nonempty interrupting sets are disjoint with interrupting sets of other transitions. Simple nets eliminate "propagation" of interrupts when one interrupted transition, through its input set, interrupts another transition. In order to simplify the description of net behavior, only simple nets are considered in this paper.

The behavior of an extended D–timed Petri net can be represented by a sequence of "states" where each "state" describes the current marking as well as the firing transitions of a net. Each termination of a transition firing changes the state of a net.

A state s of an extended D–timed Petri net $\mathbf{T}$ is a triple $s = (m, n, r)$ where:

m is a marking function, $m : P \to \{0, 1, ...\}$,

n is a firing–rank function which indicates the number of active irings (i.e., the number of firings which have been initiated but are not yet terminated) for each transition of the net, $n : T \leftarrow \{0, 1, ...\}$,

r is a remaining–firing–time function which assigns the remaining firing time to each independent firing (if any) of a transition, i.e., if the firing rank of a transition t is equal to k, $n(t) = k$, the remaining–firing–time function $r(t)$ is a vector of k nonnegative nondecreasing real numbers denoted by $r(t)[1]$, $r(t)[2]$, ..., $r(t)[k]$; r is a partial function and it is undefined for all those transitions t for which $n(t) = 0$.

An initial state s_i of a net $\mathbf{T}$ is a triple $s_i{=}(m_i, n_i, r_i)$ where n_i is a selection function from the set $Sel(m_0)$, $n_i \in Sel(m_0)$, the remaining–firing–time function is equal to the firing times $f(t)$ for all those transitions t for which $n_i > 0$, and the marking m_i describes only those tokens that do not participate in firings of transitions described by n_i. An extended free–choice D–timed net $\mathbf{T}$ may have several different initial states.

A direct reachability relation can be defined for states [Z85,Z87] similarly to the reachability relation for markings. A state $s_j = (m_j, n_j, r_j)$ which is directly reachable (or g_k–reachable) from a state $s_i = (m_i, n_i, r_i)$ is obtained by the termination of the "next" firings (i.e., those firings for which the remaining firing time is the smallest one; this time is denoted by $h(s_i)$ and is called the holding time or the sojourn time of the state s_i), updating the marking of a net, checking if updated interrupting sets discontinue any active firing and performing required modifications to create an intermediate marking m_{ij}, and then initiating new firings (if any) which correspond to the selection function g_k from the set $Sel(m_{ij})$.

Also, a state s_j is (generally) reachable from a state s_i if there is a sequence of directly reachable states from the state s_i to the state s_j. A set $S(\mathbf{T})$ of reachable states is defined as the set of all states of a net $\mathbf{T}$ which are reachable from the initial states of the net $\mathbf{T}$.

A state graph $\mathbf{G}$ of a D–timed Petri net $\mathbf{T}$ is a labeled directed graph $\mathbf{G}(\mathbf{T}) = (V, D, h, b)$ where:

V is a set of vertices which is equal to the set of reachable states of the net $\mathbf{T}$, $V = S(\mathbf{T})$,

D is a set of directed arcs, $D \subset V \times V$, such that (s_i, s_j) is in D iff s_j is directly reachable from s_i,

h is a vertex labeling function which assigns the holding time $h(s_i)$ of the state s_i to each state $s_i \in S(\mathbf{T})$, $h : V \to \mathbf{R}^{\oplus}$,

b is an arc labeling function which assigns the probability of transitions from s_i to s_j to each arc (s_i, s_j) in the set D, $b : D \to [0, 1]$ [Z87].

It should be observed that the state graph of a free–choice D–timed Petri net is a discrete–state discrete–time semi–Markov process [Klei]. Stationary probabilities $x(s)$ of the states $s \in S(\mathbf{T})$, can thus be obtained by standard methods from the embedded stationary probabilities.

Example. The D–timed Petri net shown in Fig.2 (the interrupt arcs have small dots instead of arrowheads and the firing time function as well as the choice function are given as additional descriptions of transitions) is a refinement of the net from Fig.1; it contains one interrupt arc (p_5, t_5), one free–choice place p_3, and one guarded place p_4. The derivation of the set $S(\mathbf{T})$ of reachable states is given in Tab.1, which also shows the stationary probabilities of the states $x(s_i)$. •

		m_i	n_i			
i	$x(s_i)$	1 2 3 4 5	1 2 3 4 5 6	h_i	j	$b(s_i, s_j)$
1	0.196	0 0 0 0 0	1 0 0 0 0 0	1.0	2	1.00
2	0.392	0 0 0 0 0	0 1 0 0 1 0	2.0	3	0.90
					4	0.10
3	0.353	0 0 0 0 0	0 0 0 1 1 0	2.0	5	1.00
4	0.000	0 0 0 0 0	0 0 1 0 1 0	0.0	6	1.00
5	0.000	0 0 0 0 0	1 0 0 0 0 1	0.0	1	1.00
6	0.059	0 0 0 0 0	0 0 0 0 1 0	3.0	1	1.00

Tab.1. The set of reachable states for $\mathbf{T}_1$.

4. NET TRANSFORMATIONS

It can easily be shown that different D–timed nets may have isomorphic state graphs, which means that the behavior of such nets is "equivalent" in the sense of stationary probabilities of states and performance properties that can be derived from these stationary probabilities. Fig.3 shows a D–timed net which is much simpler than the net from Fig.2, and which is equivalent to the net $\mathbf{T}_1$. Tab.2 shows the derivations of the state space $S(\mathbf{T}_2)$. $\mathbf{T}_2$ can be further simplified by aggregating the transitions which form simple paths in a net, e.g., t_1 and t_2, and removing transitions with firing times equal to zero (t_3 and t_6) since they are insignificant for "timed" behavior of the net (the stationary probabilities of states corresponding to firing such transitions are equal to zero, and they can be eliminated without any effect on the performance properties of the model).

		m_i	n_i			
i	$x(s_i)$	1 2 3 4 5	1 2 3 4 5 6	h_i	j	$b(s_i, s_j)$
1	0.196	0 0 0 0 0	1 0 0 0 0 0	1.0	2	1.00
2	0.392	0 0 0 0 0	0 1 0 0 0 0	2.0	3	0.90
					4	0.10
3	0.353	0 0 0 0 0	0 0 0 1 0 0	2.0	5	1.00
4	0.000	0 0 0 0 0	0 0 1 0 0 0	0.0	6	1.00
5	0.000	0 0 0 0 0	0 0 0 0 0 1	0.0	1	1.00
6	0.059	0 0 0 0 0	0 0 0 0 1 0	3.0	1	1.00

Tab.2. The set of reachable states for $\mathbf{T}_2$.

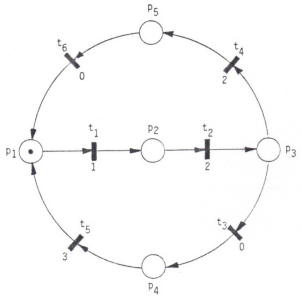

Fig.3. D-timed net $\mathbf{T}_2$.

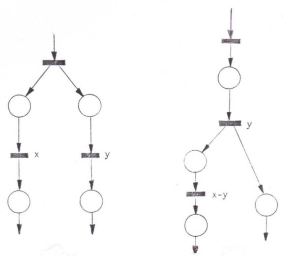

Fig.4. Delayed split transformation.

The net $\mathbf{T}_2$ is live and bounded (in fact, it is safe) since it is covered by live and bounded subnets (or S–invariants [Reis]); verification of protocol properties using $\mathbf{T}_2$ is thus a trivial task.

The net $\mathbf{T}_2$ can be derived from the net $\mathbf{T}_1$ by applying a number of simple net transformations.

(a) Delayed split: Since the transitions initiate their firings in the same time instances in which they become enabled, a pair of transitions that initiate their firings simultaneously can be replaced by a single transitions with the firing time equal to the smaller one of the original firing times, as shown in Fig.4. Delayed split applied to t_2 and t_5 in Fig.2 results in the net $\mathbf{T}_3$ shown in Fig.5 (it should be noticed that the "unmarked" place p_5 is critical for this application).

(b) Shifted choice: Actually this is a variation of the previous transformation when one of the output places of a split transition is a free–choice place p; in such situation the delayed split is "moved" to all free–choice transitions of p, as shown in Fig.6. This transformation applied to t_2, t_3 and t_4 in Fig.5 results in the net $\mathbf{T}_4$ shown in Fig.7. It can be observed that $\mathbf{T}_4$ contains places and transitions which can be removed without any significant effect on the behavior of the net.

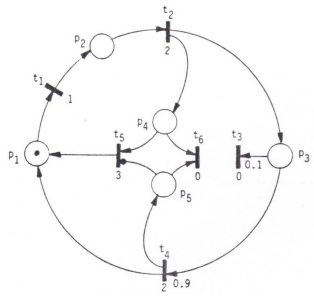

Fig.5. D-timed net $\mathbf{T}_3$.

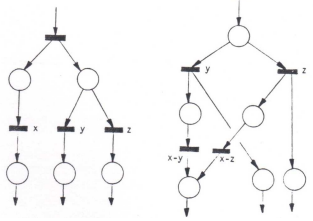

Fig.6. Shifted choice transformation.

(c) Removal of redundant arcs: It should be noticed that firing t_3 in $\mathbf{T}_4$ disables t_4, and therefore the subsequent firing of t_5 cannot be interrupted; consequently, the interrupt arc (p_5, t_5) is redundant and can be removed, and similarly the arc (p_4, t_6). Moreover, the firing of t_4 will deposit tokens in p_4 as well as in p_5, after which t_6 fires removing the deposited tokens from p_4 and p_5; the arcs (t_4, p_5) and (p_4, t_6) are clearly redundant and can be removed (however, the firing of t_6 should be maintained to preserve the state space). The net $\mathbf{T}_4$ after this modifications is shown in Fig.8.

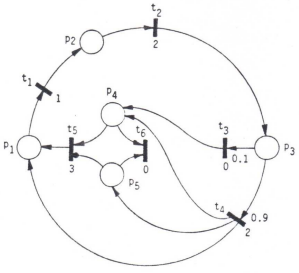

Fig.7. D-timed Petri net $\mathbf{T}_4$.

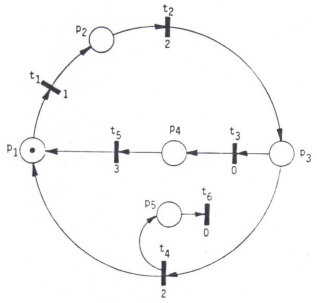

Fig.8. D-timed Petri net $\mathbf{T}_5$.

Since the firing time of t_6 is equal to zero, the delayed split transformation applied to it inserts t_6 in series with t_4 which results in $\mathbf{T}_2$ from Fig.3.

5. PROTOCOL MODELLING AND ANALYSIS

The D–timed net shown in Fig.2 is a model of a very simple protocol in which messages are exchanged between a sender (place p_1) and a receiver (place p_3), and each received message is confirmed by an acknowledgement sent back to the sender (in the loop $p_1, t_1, p_2, t_2, p_3, t_4, p_1$). There is a nonzero probability that the system can lose (or distort) a message or an acknowledgement; the place p_3 is a free–choice place, and the transition t_3 models a message/acknowledgement "sink"; the probability associated with t_3, $c(t_3)$, represents thus the probability of losing a message or an acknowledgement (or shortly a "token") in the system. A "timeout" is used to recover from lost "tokens". It works in the following way. An event of "sending a message" is modelled by the transition t_1. When it fires, single tokens are deposited in p_2 (a "message") and in p_4 (a "timeout"). A token in p_4 immediately starts a firing of the "timeout" transition t_5 (since p_5 is empty). The firing time associated with t_5 is large enough to allow the transfer of a message and an acknowledgement. If there is no loss of tokens, i.e., if t_4 is selected for firing (according to its probability), the transition t_4 will finish its firing before t_5, and then a token in the place p_5 interrupts and cancels the timeout (i.e., the firing of t_5), the "timeout" token is returned to p_4, and then t_6 fires and removes the tokens from p_4 and p_5 (t_6 is another token "sink"). If, however, a message or acknowledgement has been lost (i.e., if t_3 has been selected for firing instead of t_4), the timeout t_5 ends its firing without interruption, and regenerates the "lost" token in p_1, i.e., the message is retransmitted to the receiver.

The throughput of this protocol can be obtained from stationary probabilities of the states (Tab.1). Since each correct transfer corresponds to a single firing of t_4 (i.e., to receiving an acknowledgement), and t_4 fires in the state s_3 only (Tab.1, $n_3(t_4) > 0$) with firing time $f(t_4) = 2$, the throughput is equal to $x(s_3)/f(t_4)$ which is 0.353/2=0.176 messages per time unit.

This simple communication protocol (Fig.2) may be adequate for short distances, but is very inefficient for long distance communication since to send another message, the

sender must wait the whole delay of the message as well as acknowledgement. For long distance communication, a more efficient method is to allow the sender to transmit several consecutive messages without waiting for an acknowledgement; since the transmitted messages may be lost or damaged in transit, the sender must store them in a buffer for possible retransmission. The concept of a "window" is used to denote the messages stored in the buffer, and "window size" determines the maximum number of outstanding unacknowledged messages. When the sender eventually receives acknowledgement for a buffered message, it releases the corresponding space in the buffer and transmits another message storing it in the released buffer space (the protocol's window "slides" by one message). Such "sliding window" protocols can be modelled and analyzed in a way very similar to the presented example.

6. CONCLUDING REMARKS

Interrupt arcs (which are "special" inhibitor arcs) provide a simple mechanism to discontinue the firing of transitions, which can be used for strict modelling of timeouts in communication protocols. Consequently, D–timed protocol models are rather simple (in fact, they are simpler than in many other approaches), and their parameters correspond in a very natural way to components or activities of modelled systems (e.g., the numbers of messages, timeout signals, etc.).

It should be noticed that the state space can easily be generated from model specifications (for example by appropriate computer programs), and since the states of the modelling net directly correspond to the states of the modelled system, a verification step is provided which is not available in simulation or analytical modelling. Moreover, a number of simple net transformation have been presented which simplify the structure of nets but which preserve the behavior of the original modelling net. Systematic applications of such transformations can convert a modelling net with many extensions into an equivalent simple "standard" net with well–known or easily verifiable properties. The described transformations (as well as many other ones, some of which may be quite difficult to check) can be implemented within more general systems for computer–aided design and analysis of protocols; their application can be either interactive, or fully "automated", in which case a more general strategy for net conversions is needed.

The class of timed Petri nets discussed in the paper is restricted in several ways (simple free–choice nets), some of the restrictions, however, can be removed by rather straightforward extensions of the formalism.

Acknowledgement

The Natural Sciences and Engineering Research Council of Canada partially supported this research through Operating Grant A8222.

References

[AMCB] M. Ajmone Marsan, G. Conte, G. Balbo, "A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems"; ACM Trans. on Computer Systems, vol.2, no.2, pp.93–122, 1984.

[BeTe] G. Berthelot, R. Terrat, "Petri net theory for the correctness of protocols"; IEEE Trans. on Communications, vol.30, no.12, pp.2497–2505, 1982.

[BeMe] B. Berthomieu, M. Menasche, "An enumerative approach for analyzing time Petri nets"; Information Processing 83, R.E.A. Mason (ed.), pp.41–45, IFIP 1983.

[Boch] G.V. Bochmann, "A general transition model for protocols and communication services"; IEEE Trans. on Communications, vol.28, no.4, pp.643–650, 1980.

[CAA] J.P. Courtiat, J.M. Ayache, B. Algayres, "Petri nets are good for protocols"; Computer Communication Review, vol.14, no.2, pp.66–74, 1984.

[Dan] A. Danthine, "Protocol representation with finite–state models"; IEEE Trans. on Communications, vol.28, no.4, pp.632–643, 1980.

[Diaz] M. Diaz, "Modeling and analysis of communication and cooperation protocols using Petri net based models"; Computer Networks, vol.6, no.6, pp.419–441, 1982.

[Klei] L. Kleinrock, "Queueing systems", vol.1: "Theory", vol.2: "Computer applications"; J. Wiley & Sons 1975, 1976.

[MeFa] P.M. Merlin, D.J. Farber, "Recoverability of communication protocols – implications of a theoretical study"; IEEE Trans. on Communications, vol.24, no.9, pp.1036–1049, 1976.

[Pet] J.L. Peterson, "Petri net theory and the modeling of systems", Prentice–Hall 1981.

[Ram] C. Ramchandani, "Analysis of asynchronous concurrent systems by timed Petri nets"; Project MAC Technical Report MAC–TR–120, Massachusetts Institute of Technology, Cambridge MA, 1974.

[Raz] R.R. Razouk, "The derivation of performance expressions for communication protocols from timed Petri nets"; Computer Communication Review, vol.14, no.2, pp.210–217, 1984.

[RaPh] R.R. Razouk, C.V. Phelps, "Performance analysis using timed Petri nets"; Proc. Int. Conf. on Parallel Processing, Columbus OH, pp.126–128, 1984.

[Reis] W. Reisig, "Petri nets – an introduction"; Springer Verlag 1985.

[Sifa] J. Sifakis, "Use of Petri nets for performance evaluation"; in: "Measuring, modelling and evaluating computer systems", pp.75–93, North–Holland 1977.

[ZWCB] P. Zafiropulo, C.H. West, D.D. Cowan, D. Brand, "Towards analyzing and synthesizing protocols"; IEEE Trans. on Communications, vol.28, no.4, pp.651–661, 1980.

[Z85] W.M. Zuberek, "Extended D–timed Petri nets, timeouts, and analysis of communication protocols", Proc. ACM Annual Conference, Denver CO, 1985, pp.10–15.

[Z86] W.M. Zuberek, "M–timed Petri nets, priorities, preemptions, and performance evaluation of systems"; in: "Advances in Petri Nets 1985" (Lecture Notes in Computer Science 222), G. Rozenberg (ed.), pp.478–498, Springer Verlag 1986.

[Z87] W.M. Zuberek, "D–timed Petri nets and modelling of timeouts and protocols"; Trans. of the Society for Computer Simulation, vol.4, no.4, pp.331–357, 1987.