Invariant-Based Performance Analysis of Timed Petri Net Models

W.M. Zuberek

Abstract In timed Petri nets, temporal properties are associated with transitions as transition firing times (or occurrence times). For net models which can be decomposed into a family of place invariants, performance analysis can be conveniently performed on the basis of its components. The paper presents an approach to finding place invariants of net models and proposes an incremental method which, for large models, can significantly reduce the required amount of computations.

Keywords Timed Petri nets · Place invariants · Performance analysis · Incremental model analysis

1 Introduction

Petri nets [6, 7, 10] have been proposed as a formalism for modeling and analysis of discrete-event systems with asynchronous, interacting components. Computer and communication networks, manufacturing systems and transportation networks are just a few examples of such systems. Popularity of net models is due to a simple and 'natural' representation of concurrent and asynchronous activities, typical for many discrete-event dynamical systems, that, however, cannot be modeled easily using queueing theory or other traditional modeling and evaluation techniques. Moreover, a well-developed mathematical foundations exists for the description and analysis of net models.

In order to study the performance aspects of Petri net models, the duration of activities must also be taken into account. Several types of Petri nets 'with time' have been proposed by assigning 'firing times' to transitions or 'enabling times' to places [1, 8, 11]. In timed Petri nets [2, 9, 12, 13], the events occur in 'real time',

W.M. Zuberek (✉)
Department of Computer Science, Memorial University,
St.John's, NL A1B 3X5, Canada
e-mail: wlodek@mun.ca

© Springer International Publishing Switzerland 2016

W. Zamojski et al. (eds.), *Dependability Engineering and Complex Systems*,
Advances in Intelligent Systems and Computing 470,
DOI 10.1007/978-3-319-39639-2_52

i.e., there is a (deterministic or stochastic) duration associated with each transition's firing, and different (concurrent) firings of transitions correspond to (concurrent) activities in the modeled systems. For timed Petri nets, the concept of 'state' and state transitions can be formally defined, and used to derive different performance characteristics of the model [13].

For a class of Petri net models, structural analysis, based on place invariants [10], is an attractive approach because it provides an analytical characterization of the model's performance, and also it eliminates the exhaustive analysis of the state space (which can be huge for large models). Structural analysis is based on the set of place invariants, which—for complex models—can be difficult to find. Incremental approach reduces the process of finding basic place invariants by first finding the invariants for very simple submodels of the original models, and then combining the submodels into more complex ones with invariants determined in a way that eliminates many steps of the direct approach to finding the invariants.

Section 2 recalls basic concepts of Petri nets and timed nets, their place invariants and (structural) performance analysis. Finding place invariants is discussed in Sect. 3 while Sect. 4 introduces the incremental approach and compares it with the direct approach of Sect. 3. Section 5 provides some concluding remarks.

2 Nets, Net Invariants, Timed Nets and Performance Analysis

A place/transition (ordinary, i.e., with no arc weights) net $\mathbf{N}$ is a triple $\mathbf{N} = (P, T, A)$ where P is a finite, nonempty set of places, T is a finite, nonempty set of transitions, and A is a set of directed arcs, $A \subseteq P \times T \cup T \times P$, such that for each transition there exists at least one place connected with it. For each place p (and each transition t) the input set, $Inp(p)$ (or $Inp(t)$), is the set of transitions (or places) connected by directed arcs to p (or t). The output sets, $Out(p)$ and $Out(t)$, are defined similarly.

A marked Petri net $\mathbf{M}$ is a pair $\mathbf{M} = (N, m_0)$ where $\mathbf{N}$ is a Petri net, $\mathbf{N} = (P, T, A)$, and m_0 is an initial marking function, $m_0 : P \to \{0, 1, \ldots\}$ which assigns a (nonnegative) integer number of tokens to each place of the net.

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

A transition t is enabled by a marking m iff m assigns at least one token to every input place of this transition. Every transition enabled by a marking m can fire (or occur). When a transition fires, a token is removed from each of its input places and a token is added to each of its output places. This determines a new marking in a net, new set of enabled transitions, and so on. The set of all markings that can be derived from the initial marking is called the set of reachable markings. If this set if finite, the net is bounded, otherwise it is unbounded.

A place p is shared iff it is an input place for more than one transition. A net is (structurally or statically) conflict-free if it does not contain shared places.

A marked net is (dynamically) conflict-free if for any marking in the set of reachable markings, and for any shared place, at most one of transitions sharing the place is enabled. Only bounded conflict-free nets are considered in this paper.

Each place/transition net $\mathbf{N} = (P, T, A)$ can be represented by a connectivity (or incidence) matrix $\mathbf{C} : P \times T \to \{-1, 0, +1\}$ in which places correspond to rows, transitions to columns, and the entries are defined as:

$$\forall p \in P \forall t \in T : \mathbf{C}[p,t] = \begin{cases} -1, & \text{if } t \in Out(p) - Inp(t), \\ +1, & \text{if } t \in Inp(p) - Out(p), \\ 0, & \text{otherwise.} \end{cases}$$

If a marking m_j is obtained from another marking m_i by firing a transition t_k then (in vector notation) $m_j = m_i + \mathbf{C}[k]$, where $\mathbf{C}[k]$ denotes the k-th column of $\mathbf{C}$, i.e., the column representing t_k.

Connectivity matrices disregard 'selfloops', that is, pairs of arcs (p, t) and (t, p); any firing of a transition t cannot change the marking of p in such a selfloop, so selfloops are neutral with respect to token count of a net. A pure net is defined as a net without selfloops [10].

A P-invariant (place invariant) of a net $\mathbf{N}$ is any nonnegative, nonzero integer (column) vector I which is a solution of the matrix equation

$$\mathbf{C}^T \times I = 0,$$

where $\mathbf{C}^T$ denotes the transpose of matrix $\mathbf{C}$. It follows immediately from this definition that if I_1 and I_2 are P-invariants of $\mathbf{N}$, then also any linear (positive) combination of I_1 and I_2 is a P-invariant of $\mathbf{N}$.

A basic P-invariant of a net is defined as a P-invariant which does not contain simpler invariants. All basic P-invariants I of ordinary nets are binary vectors [10], $I : P \to \{0, 1\}$.

A net $\mathbf{N}_i = (P_i, T_i, A_i)$ is a P_i-implied subnet of a net $\mathbf{N} = (P, T, A)$, $P_i \subset P$, iff:

(1) $A_i = A \cap (P_i \times T \cap T \times P_i)$,
(2) $T_i = \{t \in T | \exists p \in P_i : (p,t) \in A \vee (t,p) \in A\}$.

It should be observed that in a (pure) net $\mathbf{N}$, each P-invariant I of $\mathbf{N}$ determines a P_I-implied (invariant) subnet of $\mathbf{N}$, where $P_I = \{p \in P | I(p) > 0\}$; P_I is sometimes called the support of the invariant I; all nonzero elements of I select rows of $\mathbf{C}$, and each selected row i corresponds to a place p_i with all its input (+1) and all output (–1) arcs associated with it.

For the Petri net shown in Fig. 1, the connectivity matrix is:

C	t_1	t_2	t_3	t_4	t_5	t_6
p_1	-1	$+1$	0	0	0	0
p_2	$+1$	-1	0	0	0	0
p_3	0	-1	$+1$	0	0	0
p_4	0	$+1$	-1	0	0	0
p_5	$+1$	0	0	-1	0	0
p_6	0	0	-1	$+1$	0	0
p_7	0	0	$+1$	0	0	-1
p_8	0	0	0	0	-1	$+1$
p_9	-1	0	0	0	$+1$	0

It can be easily observed that the sum of rows 1 and 2, as well as 3 and 4 are equal to zero, so $\{p_1, p_2\}$ and $\{p_3, p_4\}$ are basic place invariants. Similarly, $\{p_5, p_6, p_7, p_8, p_9\}$ is also a basic place invariant. Subnets implied by these invariants are shown in Fig. 2.

The net shown in Fig. 1 has two more place invariants: $\{p_1, p_3, p_5, p_6\}$ and $\{p_2, p_4, p_7, p_8, p_9\}$.

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 which determines the duration of transition's firings.

A conflict-free timed Petri net $\mathbf{T}$ is a pair $\mathbf{T} = (\mathbf{M}, f)$ where:

M is a conflict-free marked Petri net, $\mathbf{M} = (\mathbf{N}, m_0)$, $\mathbf{N} = (P, T, A)$,

f is a firing time function which assigns the nonnegative (average) firing time $f(t)$ to each transition t of the net, $f : T \to \mathbf{R}^{\oplus}$, and $\mathbf{R}^{\oplus}$ denotes the set of non-negative real numbers.

The behavior of a timed Petri net can be represented by a sequence of 'states' and state transitions where each 'state' describes the distribution of tokens in places as well as firing transitions of the net; detailed definitions of states and state transitions are given in [13]. The states and state transitions can be combined into a graph of reachable states; this graph is a semi-Markov process defined by the timed net $\mathbf{T}$. For cyclic conflict-free timed nets, such state graphs are simple cycles which

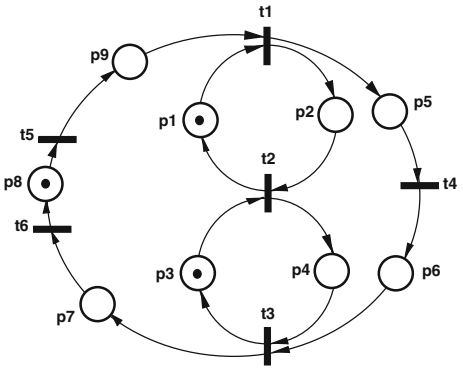

Fig. 1 Petri net

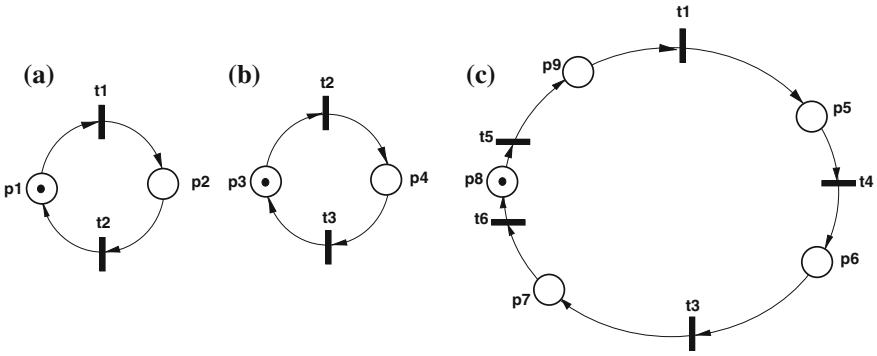

Fig. 2 Invariant implied subnets

represent the cyclic behavior of such nets. Each such timed Petri net contains a basic invariant subnet with the cycle time equal to the cycle time of the whole net. All other subnets, with smaller cycle times, will be subjected to some synchronization delays, imposed by the 'slowest' subnet that determines the cycle time of the whole net.

The cycle time of the net, τ_0, is thus equal to the maximum cycle time if its invariant subnets [11]:

$$\tau_0 = \max(\tau_1, \tau_2, \ldots, \tau_k)$$

where k is the number of subnets covering the original net, and each τ_i, $i = 1, \ldots, k$, is the cycle time of the subnet i, equal to the sum of occurrence times associated with the transitions divided by the total number of tokens assigned to the subnet:

$$\tau_i = \frac{\sum_{t \in T_i} f(t)}{\sum_{p \in P_i} m(p)}.$$

In many cases, the number of basic P-invariants can be reduced by removing from the analyzed net all these elements which do not affect the performance of models. Some of such reductions are discussed in [14].

3 Finding Place Invariants

Finding place invariants can be done in several ways [4, 5]. A polynomial algorithm for finding all place invariants can be derived from the property that the sum of rows of the connectivity matrix corresponding to a place invariant is equal to zero. For each place p_i, the algorithm starts with the i-th row of a connectivity matrix and uses other rows to eliminate nonzero elements in the original row. This is performed

by a recursive procedure "Eliminate" with three parameters, "w" which is the current working vector (initialized to the *i*-th row of the connectivity matrix), "u" which is a set of places constituting the invariant and "r" which ia a set of columns of the connectivity matrix (i.e., transitions) used for reductions of "w":

```
fun Eliminate (w : array [1..n_t] of int, u : set of int, r : set of int);
begin
    if w = 0 then
        add(u,Inv)
    else
        for i := 1 to n_t do
            if w[i] ≠ 0 then
                for j := 1 to n_p do
                    if w[i] + M[j,i] = 0 ∧ r ∩ check(w,j) = {} then
                        Eliminate(w + M[j,*],u ∪ { j },r ∪ { i })
                    fi
                od
            fi
        od
    fi
end
```

"Inv" is the set of invariants and "add(u,Inv)" adds the invariant "u" to the set. "Inv" if it is not there; n_t is the number of transitions and n_p is the number of places. The function "check(w, j)" returns the set of indices of nonzero elements of the sum of "w" and row "j" of "M".

"Eliminate" is invoked for consecutive places of the net model:

```
program Invariants1;
    Inv := {};
    for i := 1 to n_p do
        w := M[i,*];
        u := { i };
        r := {};
        Eliminate(w,u,r)
    od
end
```

The initial steps of finding place invariants for the model shown in Fig. 1 are presented in Table 1; the table shows the vector "w", the set "u" and the set "r" for consecutive invocations of "Eliminate").

Table 1 Finding place invariants for the net shown in Fig. 1

i	w	u	r	
1	−1,+1,0,0,0,0	{1}	{}	
	0.0,0,0,0,0	{1,2}	{1}	invariant {1,2}
	0,+1,0,−1,0,0	{1,5}	{1}	
	0,0,+1,−1.0,0	{1,5,3}	{1,2}	
	0,0,0,0,0,0	{1,5,3,6}	{1,2,3}	invariant {1,3,5,6}
2	+1,−1,0,0,0,0	{2}	{}	
	0,0,0,0,0,0	{2,1}	{1}	invariant {1,2}
	0,−1,0,0,0,+1	{2,9}	{1}	
	0,0,-1,0,0,+1	{2,9,4}	{1,2}	
	0,0,0,0,−1,+1	{2,9,4,7}	{1,2,3}	
	0,0,0,0,0,0	{2,9,4,7,8}	{1,2,3,5}	invariant {2,4,7,8,9}
3	0,−1,+,0,0,0	{3}	{}	
	−1,0,+1,0,0,0	{1,3}	{1}	
	0,0, +1,−1,0,0	{3,1,5}	{1,3}	
	…	…	…	…
9	−1,0,0,0,0,+1	{9}	{}	
	0,−1,0,0,0,+1	{9,2}	{1}	
	0,0,−1,0.0,+1	{9,2,4}	{1,2}	
	0,0,0,0,−1,+1	{9,2,4,7}	{1,2,3}	
	0,0,0,0,0,0	{9,2,4,7,8}	{1,2,3,5}	invariant {2,4,7,8,9}

The complete set of basic place invariants for the net shown in Fig. 1 is:

i	*place invariant*	*implied transitions*
1	$\{p_1, p_2\}$	t_1, t_2
2	$\{p_3, p_4\}$	t_2, t_3
3	$\{p_5, p_6, p_7, p_8, p_9\}$	t_1, t_3, t_4, t_5, t_6
4	$\{p_1, p_3, p_5, p_6\}$	t_1, t_2, t_3, t_4
5	$\{p_2, p_4, p_7, p_8, p_9\}$	t_1, t_2, t_3, t_5, t_6

so the cycle time is:

$$\tau_0 = \max(\tau_1, \tau_2, \tau_3, \tau_4, \tau_5)$$

where:

$$\begin{aligned}
\tau_1 &= f(t_1) + f(t_2),\\
\tau_2 &= f(t_2) + f(t_3),\\
\tau_3 &= f(t_1) + f(t_3) + f(t_4) + f(t_5) + f(t_6),\\
\tau_4 &= (f(t_1) + f(t_2) + f(t_3) + f(t_4))/2,\\
\tau_5 &= f(t_1) + f(t_2) + f(t_3) + f(t_5) + f(t_6).
\end{aligned}$$

Other performance characteristics can be derived in a similar way [3].

4 Incremental Approach

In many cases, the components of the model are known in advance and can be used for incremental approach to finding place invariants of a net model.

For the example shown in Fig. 2, place invariants of simple subnets are obvious (and even formally can be determined in a single step). Submodels (a) and (c) (i.e., subnets implied by place invariants $\{p_1, p_2\}$ and $\{p_5, p_6, p_7, p_8, p_9\}$) are combined using a single transition (t_1) which does not change the invariants. The combined subnet implied by $\{p_1, p_2, p_5, p_6, p_7, p_8, p_9\}$ is merged with the remaining subnet by transitions t_2 and t_3, as shown in Fig. 3. This integration can create new invariants, but all such invariants must contain places connected to t_2 and/or t_3, so only these places should be checked for new place invariants.

In general, if two subnets are merged by transitions in a set T_{shared}, new place invariants are found by the same procedure as before but restricted to places in the set $Inp(T_{shared})$ and $Out(T_{shared})$:

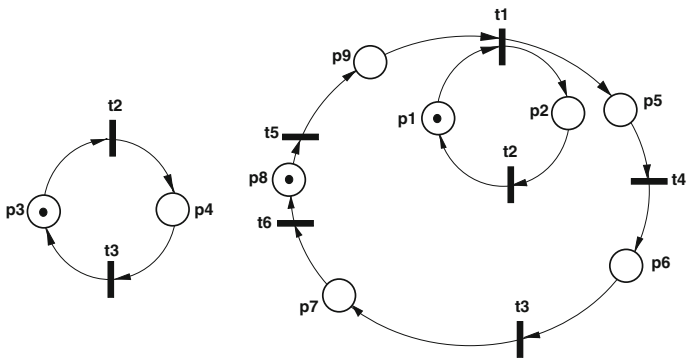

Fig. 3 Merging subnets

```
program Invariants2;
    Inv := {};
    for each p_i in Inp(T_shared) ∪ Out(T_shared) do
        w := M[i,*];
        u := { i };
        r := {};
        Eliminate(w,u,r)
    od
end
```

Analysis of the case shown in Fig. 3 corresponds to a part of Table 1 that includes places p_1 to p_4, p_6 and p_7, which is about one half of the total Table.

It should be observed that the gains of the incremental approach actually increase with the size of the model.

5 Concluding Remarks

Invariant-based performance analysis derives analytical characterization of the model's performance provided the model is covered by a family of conflict-free subnets. If this is the case, the subnets are implied by place invariants of the net model. Efficient method of finding place invariants uses an incremental approach of merging simple submodels into more complex ones.

It should be noted that the efficiency of the incremental approach depends upon ordering the merged models. For instance, if the first step of merging the submodels shown in Fig. 2 combines submodels (a) and (b) rather than (a) and (c), the potential advantages of the approach will be lost.

The process of finding place invariants can be simplified in several ways, for example, by model reductions. Each simple path in the model can be reduced to a single element because any place invariant must either contain the whole path or none of its elements. Similarly, parallel paths (i.e., simple paths originating and terminating in a single transition) can be merged because if a place invariant contains one of these paths then there must exist another invariant containing the other path—there is no need to repeat the steps of finding these two invariants independently.

A similar considerations in the context of deadlock analysis (and finding siphons in net models) showed that simple net reductions can significantly reduce the required computations [14].

Acknowledgments The Natural Sciences and Engineering Research Council of Canada partially supported this research through grant RGPIN-8222.

References

1. Ajmone Marsan, M., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: Modeling with Generalized Stochastic Petri Nets. Wiley and Sons (1995)
2. Holliday, M.A., Vernon, M.K.: A generalized timed Petri net model for performance evaluation. In: Proceedings of the International Workshop on Timed Petri Nets, Torino, Italy, pp. 181–190 (1985)
3. Jain, R.: The Art of Computer Systems Performance Analysis. Wiley & Sons (1991)
4. Krueckeberg, F., Jaxy, M.: Mathematical methods for calculating invariants in Petri nets. In: Rozenberg, G. (ed.) Advances in Petri Nets 1987 (Lecture Notes in Computer Science 266), pp. 104–131. Springer (1987)
5. Martinez, J., Silva, M.: Simple and fast algorithm to obtain all invariants of a generalized Petri net. In: Applications and Theory of Petri Nets (Informatik Fachberichte 52), pp. 301–310. Springer (1982)
6. Murata, T.: Petri nets: properties, analysis and applications. Proc. IEEE **77**(4), 541–580 (1989)
7. Peterson, J.L.: Petri Net Theory and the Modeling of Systems. Prentice-Hall (1981)
8. Popova-Zeugmann, L.: Time and Petri Nets. Springer (2013)
9. Ramchandani, C.: Analysis of Asynchronous Concurrent Systems by Timed Petri Nets. Project MAC Technical Report Mac–TR–120. Massachusetts Institute of Technology, Cambridge (1974)
10. Reisig, W.: Petri Nets—An Introduction (EATCS Monographs on Theoretical Computer Science 4). Springer (1985)
11. Sifakis, J.: Use of Petri nets for performance evaluation. In: Measuring, Modelling and Evaluating Computer Systems, pp. 75–93. North-Holland (1977)
12. Wang, J.: Timed Petri Nets. Kluwer Academic Publ. (1998)
13. Zuberek, W.M.: Timed Petri nets—definitions, properties and applications. Microelectron. Reliab. (Special Issue on Petri Nets and Related Graph Models) **31**(4), 627–644 (1991)
14. Zuberek, W.M.: Siphon-based verification of component compatibility. In: Proceedings of 4-th International Conference on Dependability of Computer Systems (DepCoS-09); Brunow Palace, Poland, pp. 123–132 (2009)