APPLYING GLOBAL TIME PETRI NET ANALYSIS ON THE EMBEDDED SOFTWARE CONTEXT

Leticia Mara Peres[*]
lmperes@inf.ufpr.br

Luis Allan Künzle[*]
kunzle@inf.ufpr.br

Eduardo Todt[*]
todt@inf.ufpr.br

[*]Departamento de Informática
Universidade Federal do Paraná
Curitiba, Paraná, Brasil

RESUMO

Aplicação da Análise Global de Redes de Petri Temporais no Contexto de Software Embarcado
Redes de Petri e suas propriedades algébricas são usadas para modelar e analisar sistemas envolvendo paralelismo, concorrência e sincronização. Este artigo apresenta uma aplicação da técnica de Tempo Global (GTT - *global time technique*) que é uma abordagem para construir grafos de classes de redes de Petri temporais baseada nos tempos relativo e global. Além da construção deste grafo de classes propomos uma análise de escalonabilidade do tipo temporal quantitativa para políticas de prioridade fixa e *earliest deadline first* (EDF). Propomos que a análise de cenários, ou a duração de itinerários de comportamento, de um sistema pode ser feita usando esta técnica.

PALAVRAS-CHAVE: Rede de Petri Temporal, Análise Quantitativa, Software Embarcado.

ABSTRACT

This paper presents an application of Global Time technique (GTT) which is an approach to construct class graphs of Time Petri nets based on relative and global time. Besides the constructing of GTT class graph we propose a schedulability analysis of quantitative time type for fixed priority policies and Earliest Deadline First (EDF). We propose that analysis of scenarios, or behavior itineraries duration of a system, can be done using this approach.

KEYWORDS: Time Petri Net, Quantitative Analysis, Embedded Software.

1 INTRODUCTION

Petri nets (PN) (Murata, 1989) and their algebraic properties are used to model and analyze systems involving parallelism, concurrency and synchronization. Several extensions of basic formalism have been proposed in order to increase their modeling capacity. In this work we are interested in Time Petri nets (TPN), where quantitative time restrictions can be considered (Merlin, 1974).

TPN are used in different applications of embedded system verification, scheduling and synthesis methods. Cortes et al. (Cortes et al., 2000) define a method of modeling and verification of embedded software using PRES+, a type of TPN. After the modeling, the authors outline a set of PN analysis - reachability, time and behavioral analysis. Regarding time analysis the authors propose translating PRES+ into hybrid automata and using symbolic model checking (Clarke et al., 1999), a qualitative method, to prove the correctness of the system. Lime and Roux (Lime and Roux, 2003) propose SETPN to model real-time systems, especially embedded

Artigo submetido em 10/03/2011 (Id.: 01294)
Revisado em 11/05/2011, 15/07/2011, 15/08/2011
Aceito sob recomendação do Editor Associado Prof. Carlos Roberto Minussi

systems. They present a set of PN design patterns to model tasks with preemptive scheduling and provide a method using a polyhedron representation and later difference bound matrix. Yet, they describe observers that give a numeric result for the computation of response times of tasks. In (Lime and Roux, 2008), the authors use their scheduling TPN design patterns of (Lime and Roux, 2003) to model tasks and deal with Fixed Priority and Earliest Deadline First policies, with the possibility of using round-robin for tasks with the same priority. In (Cortes et al., 2000) and (Lime and Roux, 2008) the nets are translated into linear hybrid automata and timed properties are verified using the symbolic model checking Hytech tool (Henzinger et al., 1997).

The prevalent technique in the literature of TPN analysis is based on the graph of state classes (Berthomieu and Menasche, 1983), (Berthomieu and Diaz, 1991) and (Berthomieu and Vernadat, 2003). This is an enumerative method which generates a reachable state space using the behavior of a TPN. In this method, each class is a graph node and groups the state set with the same marking, and each time interval encompasses possible firing instants for each enabled transition in the class. Each enabled transition can fire in each class once, generating another class, and this possible firing is an arc graph linking these two classes. This method finds the relative time interval during which the system remains in a particular class.

Wang *et al.* at (Wang et al., 2000), propose to find the TPN global time using a class graph based on (Berthomieu and Menasche, 1982). The global time corresponds to absolute time of the accumulation of time since the beginning of net execution, or initial marking, until the observed class, i.e., referring time information to the chaining of events represented by transitions firing sequences. In this method, the unique time is calculated with no adjustments for concurrent enabled transitions.

In this paper we propose an application of the Global Time Technique (GTT). GTT was proposed by Lima et al. in (Lima et al., 2005), (Lima et al., 2006), and (Lima et al., 2008), and is an extension of (Berthomieu and Menasche, 1982) and (Wang et al., 2000) methods, and generates a TPN class graph with two types of time that are useful and usable in a Petri net simulation: relative and global time. The novelty of the work presented in this paper is propose an algorithm that handles a k-limited TPN and an application which makes possible verify the schedulability of real-time embedded software. A k-limited TPN is one that its places can contain at most k tokens, $k \geq 1$. GTT computes global time information using relative time intervals of (Berthomieu and Menasche, 1982) and adjusting the obtained time considering the persistence or not of enabled transitions. GTT handles concurrent events reducing the increase of imprecision when compared to the simple sum of relative times, due to adjustments of this technique. The global time information is correct for both limits of time interval of any class, even when the net represents concurrency among events (Lima et al., 2006), (Lima et al., 2008) e (Mattar Junior et al., 2007).

This paper is a revised and extended version of a work published in Portuguese at XVIII Congresso Brasileiro de Automática (CBA2010) proceedings (Peres et al., 2010). It concerns the application of global time TPN on the embedded software context and is a result of studies about TPN analysis using interval algebra initiated by (Lima et al., 2005), (Lima et al., 2006), and (Lima et al., 2008). We use some procedures and calculations defined by (Mattar Junior et al., 2007) which presented studies about firing sequence global time analysis of 1-limited TPN.

The remainder of this paper is organized as follows. Section 2 defines basic concepts of interval operations, TPN and class graph. Section 3 establishes GTT, gives an algorithm for computing the GTT class graph, and presents an example of its application. Section 4 presents an application of GTT on the context of embedded software verification, while Section 5 concludes the article.

2 BASIC CONCEPTS

The Global Time Technique (GTT) is a TPN analysis based on operations on numeric intervals. In this section we define the basic concepts used by GTT.

2.1 Numeric Intervals

Definition 1 *(Intervals and Operations)* Given two rational numbers, a and b, such that $a \leq b$. We denote $[a, b]$ as the set $\{x \in \mathbb{Q} : a \leq x \leq b\}$, defined as a closed interval from a to b. An interval $[c, d]$ is denoted as not proper when $d < c$, with c and d rationals.

Let a, b, c and d be rational numbers, given the intervals $[a, b]$ and $[c, d]$, proper or not proper, we define the following operations:

$$[a, b] + [c, d] = [a + c, b + d]$$
$$[a, b] - [c, d] = [max\{0, a - d\}, max\{0, b - c\}]$$
$$[a, b] \ominus [c, d] = [max\{0, a - c\}, max\{0, b - d\}]$$

Since irrational numbers are not computable at real computers, we must use rational numbers instead of real numbers.

2.2 Time Petri nets

Definition 2 *(Time Petri Net)* A time Petri net is a tuple $TPN = (P, T, Pre, Pos, e)$ (Berthomieu and Diaz, 1991), where:

- P is a finite set of places, $P \neq \emptyset$,
- T is a finite set of transitions, $T \neq \emptyset$,
- Pre is an weight function of arcs from places to transitions, $Pre : (P \times T) \rightarrow \mathbb{N}$,
- Pos is an weight function of arcs from transitions to places, $Pos : (T \times P) \rightarrow \mathbb{N}$,
- M_0 is the initial marking, and
- $e : T \rightarrow (\mathbb{Q}^+ \times (\mathbb{Q}^+ \cup \{\infty\}))$ such as $e(t) = [a, b]$, where $t \in T$ and $[a, b]$ is an interval.

A *marking* is an assignment of tokens to places that defines the state of the net. The marking of a place $p \in P$ is denoted $M(p)$, such as $M : P \rightarrow \mathbb{N}$. The initial marking of a TPN is denoted by M_0.

Definition 3 *(Enabled transition)* A transition $t \in T$ is enabled in a marking M_k iff for all $p \in P$, $M_k(p) \geq Pre(p, t)$.

Definition 4 *(Firing)* If t is an enabled transition in a marking M_{k-1}, then t can fire. The firing of t changes de state of the net and produces a new marking. The new marking M_k is given by $M_k(p) = M_{k-1}(p) - Pre(p, t) + Pos(t, p)$, for all $p \in P$. The firing of t is denoted by $M_{k-1}[t\rangle M_k$. If the firing of t is succeeded by one or more transition firings, we say that this sequence of firings form a firing sequence s. The firing of s in a marking M_j produces a sequence of new markings ended by M_k, denoted by $M_j[s\rangle M_k$.

A TPN has one static time interval $e(t) = [a, b]$, with $a \leq b$, associated to each transition $t \in T$. The limits a and b represent, respectively, the earliest and the latest possible firing time of transition t, counted from the instant when t is enabled.

Definition 5 *(State class)* A state class of a TPN is a tuple $c_k = (M_k, W_k)$, where M_k is the marking that defines de state of the net and W_k is the time information set associated with this marking. W_k will be further detailed in the Definition 14.

We say that a transition $t \in T$ is enabled (fires) in a class c_k iff t is enabled (fires) in M_k. When a transition t fires in a certain class c_{k-1}, at level $k-1$, the TPN reaches a new marking and a new class c_k at level k.

Definition 6 *(State class graph)* The state class graph is a directed graph $S = (C, A)$ where each node $c \in C$ is a state class and each arc $a \in A$ connects one class c_{k-1}, at level $k-1$, to an immediately succeeding class c_k. Each arc is labeled with one transition $t \in T$ which is fired in c_{k-1}. The root node of the class graph is the start class c_0, that has the initial marking M_0.

The firing of t also produces a new class c_k in the graph S from the class c_{k-1} and is denoted by $c_{k-1}[t\rangle c_k$. We say that class c_{k-1} is immediately preceding to c_k.

One firing sequence s is reflected in the class graph S. The succeeded firing of one or more transitions in a TPN, from a class c_k to another class c_{k+n}, is represented by $c_k[s\rangle c_{k-n}$, where $n \geq 0$ is the sequence size.

Definition 7 *(Newly enabled transition)* A transition $t \in T$, enabled in a certain class c_k, is a *newly enabled* transition in c_k if t satisfies one of following:

- t was not enabled in class c_{k-1}; or
- the firing of t originated the class c_k, that is, t fired in the class c_{k-1} and it was re-enabled in c_k.

Definition 8 *(Persistent transition)* A transition $t \in T$, enabled in a certain class c_k, is a *persistent* transition in c_k if t was enabled in a class c_{k-1}, immediately preceding to c_k and t did not fired in c_{k-1}. Persistent is equivalent to not newly enabled.

3 GLOBAL TIME TECHNIQUE

Particularly for the Global time technique (GTT), the information set W_k of class c_k has, for each enabled transition, two types of time information: relative and global. Relative time information refers to the accumulated time since the transition had been enabled until reaches the class c_k. Global time information refers to the accumulated time since the initial marking, or start class c_0, until c_k (Lima et al., 2005). Firstly in this section we define GTT elements which compound relative and global time information and then we redefine the state class and the class graph. We present also an algorithm to generate GTT state class graph for a k-limited TPN and definitions of computing total time for a firing sequence and the permanence time in a state.

Definition 9 *(Relative time interval)* Let $r_k(t_i)$ be the relative time interval of a transition t_i calculated in a class c_k such that $c_{k-1}[t_f\rangle c_k$, defined as:

$$r_k(t_i) = \begin{cases} e(t_i) & \text{case 1,} \\ r_{k-1}(t_i) - r_{k-1}(t_f) & \text{case 2,} \end{cases}$$

where cases 1 and 2 are, respectively:

1. t_i is *newly enabled* in c_k;
2. t_i is *persistent* in c_k.

The relative time interval $r_k(t_i)$ of the enabled transition t_i of a class c_k is used to identify which transitions are firable at class c_k.

Definition 10 *(Firable transition)* A transition t_f with $r_k(t_f) = [a_f, b_f]$ is firable in c_k iff t_f is enabled in c_k and there is no other transition t_i with $r_k(t_i) = [a_i, b_i]$ enabled in c_k such that $b_i < a_f$.

Definition 11 *(Persistence coefficient)* A persistence adjustment coefficient $ac_k(t_i)$ of an enabled transition t_i in a class c_k such that $c_{k-1}[t_f\rangle c_k$, is defined as:

$$ac_k(t_i) = \begin{cases} r_{k-1}(t_i) \ominus r_{k-1}(t_f) & \text{case 1,} \\ ac_{k-1}(t_i) \ominus r_{k-1}(t_f) & \text{case 2,} \\ r_{k-1}(t_i) \ominus ac_{k-1}(t_f) & \text{case 3,} \\ ac_{k-1}(t_i) \ominus ac_{k-1}(t_f) & \text{case 4,} \end{cases}$$

where cases 1 to 4 are respectively:

1. t_i and t_f are both newly enabled in c_{k-1};
2. t_i is persistent and t_f is newly enabled, both in c_{k-1};
3. t_i is newly enabled and t_f is persistent, both in c_{k-1};
4. t_i and t_f are both persistent in c_{k-1}.

The persistence adjustment coefficient $ac_k(t_i)$ is used to adjust the computing of global time and prevents the increase of imprecision, as proved by (Mattar Junior et al., 2007).

Definition 12 *(Global time interval)* A global time interval $g_k(t_i)$ of firing of a firable transition t_i in a class c_k such that $c_{k-1}[t_f\rangle c_k$ is:

$$g_k(t_i) = \begin{cases} e(t_i) & \text{case 1,} \\ g_{k-1}(t_f) + r_k(t_i) & \text{case 2,} \\ g_{k-1}(t_f) + ac_k(t_i) & \text{case 3,} \end{cases}$$

where cases 1 to 3 are respectively:

1. $k = 0$;
2. $k \neq 0$ and t_i is newly enabled in c_k;
3. $k \neq 0$ and t_i is persistent in c_k.

The global time interval $g_k(t_f)$ of firing of a transition t_f in a class c_k is a time interval counted from the initial marking until the firing instant of t_f in the class c_k.

However, in order to take a conservative character, supposing the firing of all firable transitions, it is necessary to adjust the interval upper bound of $g_k(t_i)$.

Definition 13 *(Upper bound adjustment)* Let t_f be the fired transition in a class c_k such that $c_k[t_f\rangle c_{k+1}$. The upper bound of global time interval $g_k(t_f) = [a_f, b_f]$ of the fired transition t_f, must be adjusted by the lowest upper bound of intervals calculated to all t_i in the class c_k, generating a new $g_k(t_f)$, such as:

$$g_k(t_f) = [a_f, \bar{b}],$$

where $\bar{b} = min\{b_i \mid g_k(t_i) = [a_i, b_i],\ \forall t_i$ firable in $c_k\}$.

Definition 14 *(GTT state class)* A GTT state class c_k of a TPN, is a tuple $c_k = (M_k, W_k)$, with $W_k = (H, E, F, R, G)$, where:

- M_k is the marking of c_k;
- H is the enabled transition set in c_k;
- E is one tuple $\langle E_0, E_1, E_2\rangle$ which contains persistence information of transitions of class c_k where: E_0 is the set of newly enabled transition in c_k, E_1 is the set of persistent transitions in c_k that was newly enabled in c_{k-1}, and E_2 is the set of persistent transition in the classes c_k that was also persistent in c_{k-1};
- F is the firable transition set in c_k;
- R is the relative domain set which contains relative time intervals $r_k(t_i)$ of all enabled transitions t_i in c_k.
- G is one tuple $\langle ac_k, g_k\rangle$ which corresponds to the global time domain of c_k, where $ac_k(t_i)$ is the persistence adjustment coefficient of all persistent transitions t_i in c_k and $g_k(t_j)$ is the global time interval of all firable transitions t_j in c_k.

Definition 15 *(GTT state class graph)* The GTT state class graph is a state class graph where each node is a *GTT state class* and each arc is labeled with one fired transition, as in the Definition 6.

3.1 The class graph algorithm

The following algorithm builds a class graph according to GTT for a k-limited TPN.

3.1.1 The input data

- The net structure, represented by one input incidence matrix $pre = [Pre(p,t),\ p \in P,\ t \in T]$ and one output incidence matrix $pos = [Pos(t,p),\ p \in P,\ t \in T]$;
- The static time information $e(t)$, for all $t \in T$, represented by one array e with size $|T|$;
- The initial marking of the net represented by one array M_0 with size $|P|$.

3.1.2 The body

0) define $k = 0$;

Starting construction of class c_k:

1) determine the net marking M_k: if $k = 0$, $M_k \leftarrow M_0$; else M_k is already determined by the firing of transition t_f at step 7 from previous cycle;
2) determine the set H of enabled transitions t_i from M_k, according to the Definition 3;
3) determine the $\langle E_0, E_1, E_2\rangle$ tuple, according to the Definition 14;
4) compute the set R of relative time intervals $r_k(t_i)\ \forall t_i \in H$, according to the Definition 9;
5) determine the set F of firable transitions t_i comparing elements in R, according to the Definition 10;
6) determine the G tuple, computing: a) $ac_k(t_i)$, $\forall t_i \in \{E_1 \cup E_2\}$, according to the Definition 11; and b) $g_k(t_i)$, $\forall t_i \in F$, according to the Definition 12;

Firing of all $t_i \in F$ of class c_k:

7) $\forall t_i \in F$ execute all following steps: a) adjust interval $g_k(t_i)$, according to the Definition 13 being $t_f = t_i$; b) create one successor class c with level $k+1$ and one arc connecting the class c_k with the new successor class c_{k+1}, and label the arc with t_i; c) update the new marking M_{k+1} for the TPN, according to the Definition 4; and d) for all new successor classes with level $k+1$, assign the new global time $g_{k+1} = g_k(t_i)$;
8) increase $k \leftarrow k+1$ and $\forall t_i \in F$: $t_f \leftarrow t_i$ and execute steps 1 to 8;

3.1.3 The end

The algorithm is executed until all firable transitions in all classes have been fired.

3.2 Global time for a firing sequence

The global time of a firing sequence s of $c_0[s\rangle c_k$, according to the Definition 4, is obtained during construction of GTT class graph and is the resulting global time interval $g_{k-1}(t_f)$, being t_f the last transition fired to reach class c_k, that is, $c_{k-1}[t_f\rangle c_k$.

3.3 Permanence time in a class

The permanence time in a class refers to the minimum and maximum time that the system, represented by the TPN, remains in the state represented by the class.

Definition 16 *(Permanence time in a class)* We consider that the firing of all firable transitions are mandatory, i.e., each firable transition must be fired until its upper bound of time interval has reached. The permanence time of a net in a certain reachable class c_k is given by:

$$ic_{c_k} = [\overline{a}, \overline{b}]$$

where $\overline{a} = min\{a_i \mid r_k(t_i) = [a_i, b_i]\}$ and $\overline{b} = min\{b_i \mid r_k(t_i) = [a_i, b_i]\}$, $\forall t_i$ firable in c_k.

4 APPLICATION

We propose the application of GTT as an analytical method for verifying the schedulability of real-time embedded software. We based this application on the works of model-checking verification of Lime and Roux of 2003 (Lime and Roux, 2003) and 2008 (Lime and Roux, 2008). This last work defines a special TPN with a scheduling layer and, among other things, allows mapping each place of Petri net to an embedded software task in order to verify schedulability. We propose to use parts of this layer to model a TPN and, instead of model-checking qualitative analysis, to use GTT quantitative analysis. (Lime and Roux, 2003) proposed associate some kind of embedded software tasks to transitions and places of the net. We propose now to use the structures of this work as design patterns on the context of our analytical method. Design patterns are general and reusable descriptions of problems and solutions of TPN models of some kind of embedded software tasks (Naedele and Janneck, 1998). Then, from the task models in TPN, we generate the GTT state class graph and analyze firing sequences that satisfy “Earliest Deadline First” and “Fixed Priority” scheduling policies in order to verify schedulability.

4.1 Mapping between system tasks and TPN

Given, according to (Lime and Roux, 2008) and definitions of section 2:

- $\tau \in$ *Tasks, being Tasks the set of tasks of the embedded system, where there is no task migration between processors.*
- *Sched:Procs* $\mapsto$ $\{$*FP,EDF*$\}$ *maps a processor to a scheduling policy, being FP "Fixed Priority" and EDF "Earliest Deadline First" ;*
- Π*: Tasks* $\mapsto$ *Procs maps a task to its processor;*
- ϖ*: Tasks* $\mapsto \mathbb{N}$, *for* $Sched(\Pi(\tau)) = FP$, *gives the priority of the task on the processor;*
- δ*: Tasks* $\mapsto (\mathbb{Q}^+ \times (\mathbb{Q}^+ \cup \{\infty\}))$, *for* $Sched(\Pi(\tau)) = EDF$, *gives the deadline interval of the task relative to its activation time;*
- $\gamma : P \mapsto$ *Tasks* $\cup\{\phi\}$ *maps each place of the TPN to a task, where* ϕ *denotes that the place is not mapped to any real task.*

For each transition $t \in T$, there is *at most* one place p such that $p \in Pre(t)$ and $\gamma(p) \neq \phi$. If $\forall p \in Pre(t)$, $\gamma(p) = \phi$, then t is not related to any real task and t is *part of* ϕ, denoted by $\gamma(t) = \phi$. Otherwise, for each transition t, t *is part of* the task τ, denoted $t \in \tau$, if one of its input places is mapped to $\tau : t \in \tau \Leftrightarrow \exists p \in Pre(t)$, such that $\gamma(p) = \tau$. So, $\gamma(t)$ is the task such that $t \in \tau$ (Lime and Roux, 2008).

Each task τ is modeled by a subnet of the TPN composed of places mapped to τ by γ and transitions t with static time $e(t)$, which are parts of τ. At most one instance of each task is active at a given instant, which is expressed by the restriction that at most one place mapped to τ by γ is marked at a given instant (Lime and Roux, 2008).

At Figures 1, 2 and 3 we present examples of design patterns to illustrate how Petri nets are modeled. These figures present some synchronization service modeling of embedded software tasks and are 1-limited Petri nets. These examples are shown just to illustrate the possibility of application of real time scheduling policies using TPN. More details can be found in (Lime and Roux, 2003).

We propose, after the TPN modeling, to generate a GTT state class graph, as presented at section 3.

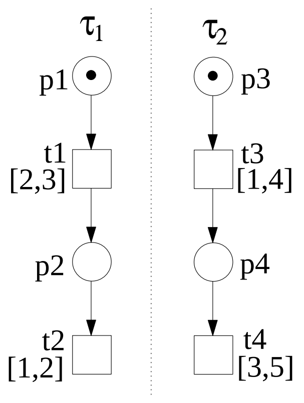

Figure 1: TPN with two concurrent tasks on one processor (Lime and Roux, 2003).

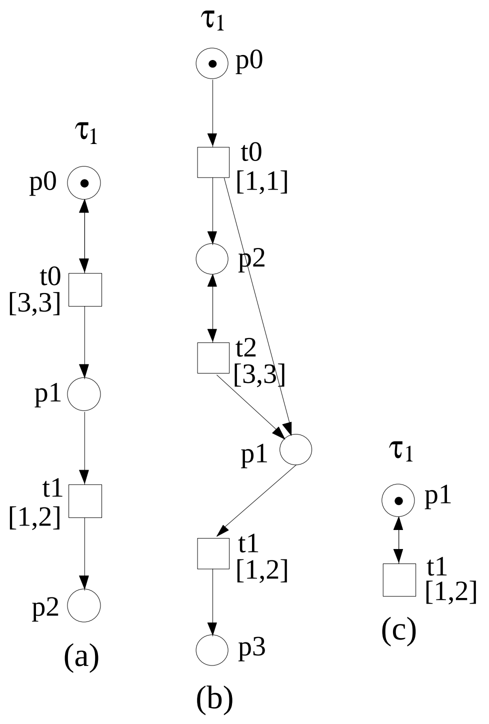

Figure 2: TPN of activation schemata: (a) periodic activation schema, (b) delayed periodic activation schema, and (c) cyclic activation schema (Lime and Roux, 2003).

4.2 Mapping between scheduling policies and the GTT graph

After to create a GTT state class graph, it is necessary to enumerate the graph, i.e. generate paths from the graph where a path is a firing sequence. We propose to enumerate paths according some criterion.

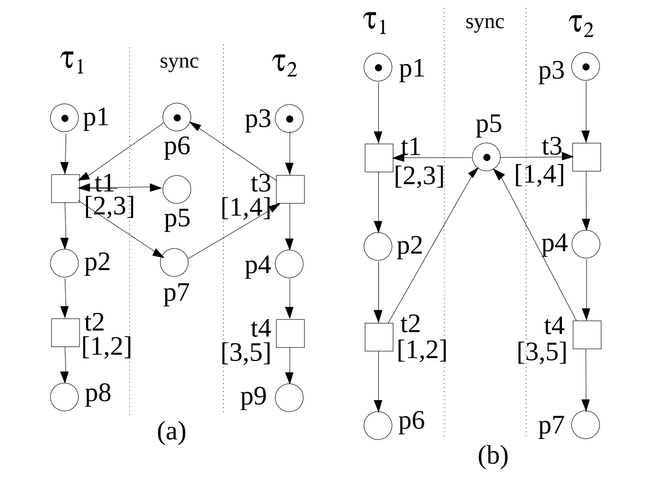

Figure 3: TPN of synchronization: (a) model for memorized events and (b) model for shared resources using a semaphore (based on Lime and Roux, 2003).

We have established scheduling policy based criteria using "Earliest Deadline First" and "Fixed Priority" scheduling policies. For *Sched:Procs* $\mapsto \{CE\}$, where CE is "Cyclic Executive", the TPN model represents only one task which is typically realized as an infinite loop in *main()*(Lime and Roux, 2003), as shown in the Figure 2(c). Because *CE* has not a specific criterion in order to be satisfied, this policy is achieved only by modeling TPN and it is not necessary to formalize this function in relation to the path generation of the GTT state class graph.

4.2.1 Fixed Priority ($Sched(\Pi(\gamma(t))) = FP$)

The function ϖ*: Tasks* $\mapsto \mathbb{N}$ guides the enumeration of firing sequence. The firing sequence consists of firable transitions t_f and the priorities of task executions associated to transitions t_i and, for the function ϖ, $t_f = t_i$, for all t_i with highest priority. In the case of tasks with the same priority at some point, one of these criterion can guide the enumeration of the firing sequence among the processes with the same priority: a FIFO choice; an earliest deadline first considering the static time $e(t_i)$ for each transition t_i, as we present in the following section; or a random choice.

At the end of this enumeration, we already have the total time for a complete or partial firing sequence according to GTT, as presented in 3.2.

4.2.2 Earliest Deadline First ($Sched(\Pi(\gamma(t))) = EDF$)

Another type of firing sequence can be enumerated on class graph, using the Earliest Deadline First scheduling policy. Then, the function δ*: Tasks* $\mapsto (\mathbb{Q}^+ \times (\mathbb{Q}^+ \cup \{\infty\}))$ guides this enumeration. Our criterion is to choose the transition which has the lowest deadline given by $\delta(\tau)$ as following.

Let $\delta(\tau)$ of a transition t_i calculated in a class c_k such that $c_{k-1}[t_f > c_k$, and defined as: $\delta(\tau) = LFT(r_k(t_i))$. The latest firing time (*LFT*) of the time $r_k(t_i)$ for each transition t_i is the guide for firing sequence enumeration.

As the *FP* policy, at the end of enumeration already has the total time for a firing sequence, as presented in 3.2.

4.3 Examples

The Figures 4, 5, 6, and 7 refer to GTT state class graph of the Figures 1, 2(a), 2(b), and 3(b), respectively. The format of GTT state class graph presented in the Figures 4, 5, 6, and 7 is: each node is a rectangle and represents one class $c_k \in C$. Each arc is a transition fired in the class c_k and generated the successor class c_{k+1}. Each node has a data header and lines with some elements of tuple W_k. The header has the following data, separeted by colons: the class name in the format $Ck_$n, where k is the level and n is a class identifier, and the class global time in the format $gk_$n, where k is the level and n is a class identifier. The next lines of node present the time information of all enabled transition t_i: the name of transition t_i; its relative time $r_k(t_i)$; and its global time $g_k(t_i)$. Each arc is labeled with one fired transition which is firable in the class c_k. In this article, our graphs are generated until the fifth level of classes.

Considering the TPN of the Figure 1. The task τ_1 has priority $\varpi = 1$ and one preemption point. The task τ_2 has also one preemption point, but priority $\varpi = 2$. Then, $\varpi(t_1) = 1$, $\varpi(t_2) = 1$, $\varpi(t_3) = 2$ and $\varpi(t_4) = 2$. The TPN class graph according to GTT is presented in the Figure 4. For $Sched(\Pi(\tau)) = FP$, the firing sequence is: t_3, t_1, t_4, t_2. It is interesting to note that, according to the Definition 10, t_1 is the only one firable in the class $C1\_2$, even t_4 being enable. The global time of this sequence is $g_{4\_12} = [4,5]$. For $Sched(\Pi(\tau)) = EDF$, the firing sequence can be t_1, t_2, t_3, t_4, with global time $g_{4\_10} = [6,9]$, or t_1, t_3, t_2, t_4, with global time $g_{4\_11} = [5,9]$.

In the Figure 2(a), the TPN represents one task τ_1 has priority $\varpi = 3$ and $\varpi(t_0) = 3$, $\varpi(t_1) = 3$. The corresponding GTT graph is presented in the Figure 5. There is one possible firing sequence, for both $Sched(\Pi(\tau)) = FP$ and $Sched(\Pi(\tau)) = EDF$, because in the classes $C1\_1$ and $C3\_3$ only t_1 is firable when t_0 and t_1 are enabled.

The same considerations of the previous example can be made about the Figure 2(b): the TPN represents one task τ_1 has priority $\varpi = 3$ and $\varpi(t_0) = 3$, $\varpi(t_1) = 3$. The corre-

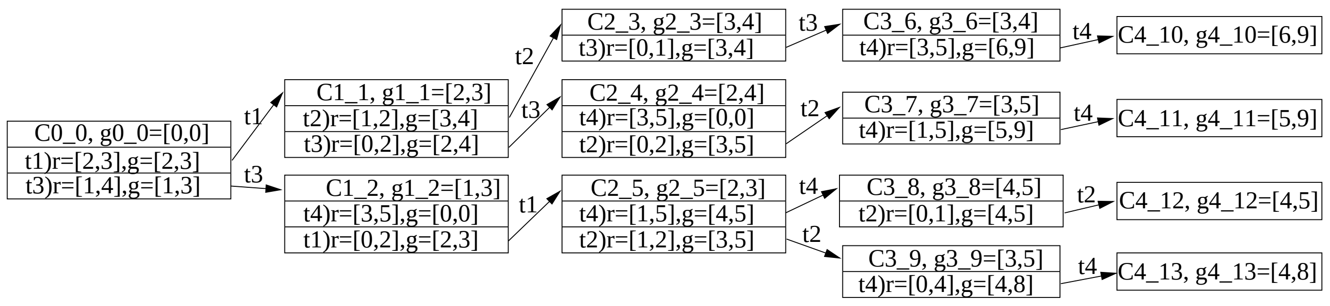

Figure 4: GTT state class graph, with 5 levels, of TPN in the Figure 1.

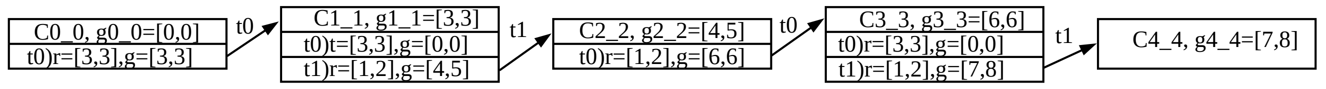

Figure 5: GTT state class graph, with 5 levels, of TPN in the Figure 2(a).

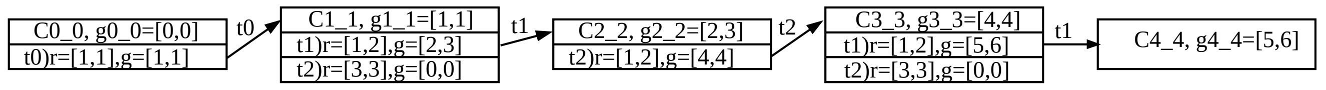

Figure 6: GTT state class graph, with 5 levels, of TPN in the Figure 2(b).

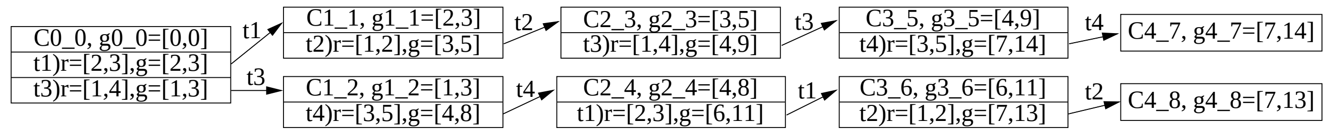

Figure 7: GTT state class graph, with 5 levels, of TPN in the Figure 3(b).

sponding GTT graph is presented in the Figure 6 and there is one possible firing sequence, for both $Sched(\Pi(\tau)) = FP$ and $Sched(\Pi(\tau)) = EDF$, and also occurs in the classes $C1\_1$ and $C3\_3$ only t_1 is firable when t_1 and t_2 are enabled.

For the TPN of the Figure 3(b): the task τ_1 has priority $\varpi = 1$ and one preemption point controlled by the semaphore (place p_5). The task τ_2 has also one preemption point, but priority $\varpi = 2$. Then, $\varpi(t_1) = 1$, $\varpi(t_2) = 1$, $\varpi(t_3) = 2$ and $\varpi(t_4) = 2$. The corresponding GTT graph is presented in the Figure 7. For both $Sched(\Pi(\tau)) = FP$ and $Sched(\Pi(\tau)) = EDF$, the firing sequence is: t_3, t_4, t_1, t_2 with the global time of this sequence $g_{4\_8} = [7, 13]$.

5 CONCLUSIONS

This paper presents the results of research whose main objective is to apply the Global Time Technique (GTT) analysis, based on works (Lima et al., 2005), (Lima et al., 2006), (Lima et al., 2008), and (Mattar Junior et al., 2007) in the verification of time constraints. We had chosen TPN because sequencing, timing, communication, and competition properties in the system can be represented and verified in this model. The construction of class graph (Berthomieu and Menasche, 1982), and also GTT state class graph, of a TPN allows the verification of several net properties, such as reachability and transitions firing sequences. GTT avoids the increase of imprecision in time information when analyzing the time of transition firing sequences (Mattar Junior et al., 2007) which represent the system behavior itineraries. This happens when the modeled system presents many concurrent or persistent transitions. Also, GTT state classes describe intervals in both global, based on the simulation beginning, and relative, based on the class entry moment, time information. This increases the analysis power of our approach.

The essence of this approach is to verify scheduling by analyzing scenarios generated using GTT from TPN. We apply and exemplify the technique by using parts of this layer to model a TPN and, instead of model checking qualitative analysis, to use GTT quantitative analysis.

We apply the technique by using some design patterns of TPN that represent a set of tasks and their interactions in embedded software. There are patterns of this kind in the literature representing a set of tasks and their interactions as proposed by (Lime and Roux, 2003) and may be tasks on one processor, cyclic tasks synchronized *via* a semaphore, semaphore for mutual exclusion and CAN bus access.

We consider this approach useful in embedded software verification of analysis of scheduling and time properties. The using of design patterns for the representation of tasks and their communication permits a rapid modeling that can be further evolved until software synthesis. Our approach suggests that the scheduling according to Fixed Priority or Earliest Deadline First may be done using GTT and its class graph. This application returns a time interval that can represent the release or start time of execution, as first value, and the deadline time, as second value.

The main contribution of our work is to apply the global time technique to verify real-time embedded systems based on tasks with fixed priority and earliest deadline first. The main limitation of the proposed approach is the endless enumeration of classes in cyclic nets, according to the indefinite increase of the global time. For the application in the context of verification, this problem is currently treated by limiting the number of execution cycles of tasks, reflected by the class levels during the generation of graph. Even with this limitation, the analysis is still useful inasmuch it corresponds to the repetition of the initial critical instant for real-time systems based on cyclic tasks. Yet, in order to overcome this limitation, we are working on the concept of *unfolding* of (McMillan, 1995) and (Esparza et al., 1996) and equivalent classes to perform reachability and firing sequence analysis. An equivalent class groups classes on which global time is increased by a constant interval. This interval is the duration of a regular cycle in the net.

6 ACKNOWLEDGMENTS

This work was supported by CNPq and CAPES, Brazil.

REFERENCES

Berthomieu, B. and Diaz, M. (1991). Modeling and verification of time dependent systems using time Petri nets, *IEEE Trans. Softw. Eng.* **17**(3): 259–273.

Berthomieu, B. and Menasche, M. (1982). A state enumeration approach for analyzing time petri nets, *3rd European Workshop on Applications and Theory of Petri Nets*, Varenna, Italy.

Berthomieu, B. and Menasche, M. (1983). An enumerative approach for analyzing time petri nets, *IFIP 9th World Computer Congress*, Vol. 9, Paris, pp. 41–46.

Berthomieu, B. and Vernadat, F. (2003). State class construction for branching analysis of time petri nets, *Proceedings of Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2003)*, Wasaw, Poland.

Clarke, E. M., Grunberg, O. and Peled, D. (1999). *Model checking*, The MIT Press, Cambridge, England.

Cortes, L. A., Eles, P. and Peng, Z. (2000). Verification of embedded systems using a Petri net based representation, *ISSS '00: Proceedings of the 13th international symposium on System synthesis*, IEEE Computer Society, Washington, DC, USA, pp. 149–155.

Esparza, J., Romer, S. and Vogler, W. (1996). An improvement of McMillan's unfolding algorithm, *Tools and Algorithms for Construction and Analysis of Systems*, pp. 87–106.
URL: *citeseer.ist.psu.edu/article/esparza96improvement.html*

Henzinger, T. A., Ho, P.-H. and Wong-toi, H. (1997). Hytech: A model checker for hybrid systems, *Software Tools for Technology Transfer* **1**: 460–463.

Lima, E. A., Lüders, R. and Künzle, L. A. (2005). Análise de redes de petri temporais usando tempo global, *In: VII Simpósio Brasileiro de Automação Inteligente - SBAI*.

Lima, E. A., Lüders, R. and Künzle, L. A. (2006). Interval analysis of time Petri nets, *In: 4th CESA Multiconference 4th CESA Multiconference on Computational Engineering in Systems Applications*, Beijing - China.

Lima, E. A., Lüders, R. and Künzle, L. A. (2008). Uma abordagem intervalar para a caracterização de intervalos de disparo em redes de petri temporais, *SBA Controle & Automação* **19**(4): 379. in Portuguese, http://dx.doi.org/10.1590/S0103-17592008000400002.

Lime, D. and Roux, O. H. (2003). Expressiveness and analysis of scheduling extended time Petrinets, *5th IFAC Int. Conf. on Fieldbus Systems and Applications,(FET'03)*, Elsevier Science, Aveiro, Portugal, pp. 193–202.

Lime, D. and Roux, O. H. (2008). Formal verification of real-time systems with preemptive scheduling, *Journal of Real-Time Systems* **41**(2): 118–151.

Mattar Junior, N., Künzle, L. A., Silva, F. and Castilho, M. (2007). Análise da duração de seqüências de disparos de transições em redes de Petri temporais, *Anais do SBAI 2007 - VIII Simpósio Brasileiro de Automação Inteligente*, Florianópolis.

McMillan, K. L. (1995). A technique of state space search based on unfolding, *Formal Methods in System Design: An International Journal* **6**(1): 45?65.

Merlin, P. (1974). *A Study of Recoverability of Computer Systems*, PhD thesis, University of California, Irvine.

Murata, T. (1989). Petri nets: Properties, analysis and applications, *Proceedings of the IEEE* **77**(4): 541–580.

Naedele, M. and Janneck, J. W. (1998). Design patterns in petri net system modeling, *4th IEEE International Conference on Engineering Complex Computer Systems (ICECCS'98)*, Monterey, California.

Peres, L. M., Künzle, L. A. and Todt, E. (2010). Aplicação da análise global de redes de petri temporais no contexto de software embarcado, *In: XVIII Congresso Brasileiro de Automática (CBA2010)*, Bonito, Brazil.

Wang, J., Deng, Y. and Xu, G. (2000). Reachability analysis of real-time systems using time petri nets, *IEEE Trans. on Systems, Man and Cybernetics-Part B : Cybernetics* .