http://www-lipn.univ-paris13.fr/~petrucci/PAPERS
Workshop on Modelling of Objects, Components and Agents (MOCA'06), Turku, Finland,
Universität Hamburg Bericht 272, june 2006, pages 223-244.

Modular Discrete Pseudo-State Graphs for Time Petri Nets

S. Mazouz[1] and L. Petrucci[2]

[1] LSI, Computer Science Department USTHB
Bp 32 El-Alia, Algiers
ALGERIA
mazouz3@lsi-usthb.dz
[2] LIPN, CNRS UMR 7030, Université Paris XIII
99, avenue Jean-Baptiste Clément
F-93430 Villetaneuse, FRANCE
petrucci@lipn.univ-paris13.fr

Summary. This paper aims at extending modular state space analysis to time Petri nets. The basic model is time Petri nets, where the timing is a firing interval attached to each transition. They are here extended to allow for modules. A state is composed by both a marking and a clock valuation function. The state spaces of time Petri nets are in general infinite even if the net is bounded, because the transitions fire in a continuous time domain. For the analysis some techniques are necessary to reduce the state space to a finite one. In this paper, we consider state spaces obtained by restricting the behaviour of time Petri nets so that transitions fire only at integer times. A modular version of such a graph is proposed as well as an algorithm for its construction.

1 Introduction

Time Petri nets (TPNs) have been widely used for the modeling and verification of real-time systems [BD91]. The most classical method for analysing TPNs, as for classical Petri nets and many other formal models, is state space analysis. State space analysis is a useful approach for the automatic verification of finite state systems. Unfortunately, it suffers from the so-called state explosion problem: the number of states can grow exponentially with the size of the system. One approach to confine this problem is that of modular analysis, which takes advantage of the modular structure of a system specification.

Some extensions have been proposed to enrich time Petri nets with module constructs such as Communicating Time Petri Nets (CmTPN) [BV95], Compositional Time Petri net [Wan98] and Time Petri net systems [BB98]. The Compositional TPN and the communicating TPN consist of two basic elements: component TPN models and inter-component connections. Their analysis is based on an incremental approach. The analysis technique of the first model, based on reduction rules, requires to construct a timing order of the components and assumes that the underlying dependency graph is acyclic. But, from our point of view, this constraint may be too restrictive. For the second model, the component of each module is analysed under the assumption of a required interface which closes the modules with a set of timing restrictions providing a partial specification for the expected environment. But defining these interfaces is not straightforward.

A time Petri net system is a set of time Petri nets which can communicate through algebraic expressions. But there is no adequate analysis technique exploiting this modularisation.

This paper proposes the extension of modular Petri nets [CP00, LP04] to deal with timing issues. The main advantage of this modular Petri net model is its capability to decide behavioural properties from a modular state space. We aim at taking advantage of this characteristic to deal with real-time systems.

In this paper, we consider a variant of discrete-time model as a first step to deal with modularisation. In the discrete-time model, the time at which each transition fires is an integer. The main advantage is that its state space generation is very simple. Even if this model is not as accurate as the continuous one, qualitative and quantitative analysis can be performed [YO95].

The paper is organized as follows. In the next section, we recall the time Petri net model definition. In section 3, we introduce modular time Petri nets. Then, in the following two sections, we develop the modular discrete pseudo-state graph and an abstract algorithm for its construction. In section 6, we present some experimental results obtained by applying the time modular technique to a case study, and compare them to the flat state graph. Section 7 concludes this paper and gives directions for future work.

Note that a similar work [LP05] has been conducted on Timed Petri Nets which use global clocks instead of transition firing intervals. It leads to modular state spaces which are not finite but are built using a time limit. Hence, the model considered here is different and our modular discrete pseudo-state graphs are finite provided the net is bounded.

2 Time Petri Nets

Several timed extensions of Petri nets have been proposed, such as time Petri nets [MF76], timed Petri nets [Ram74, Sif77]. Among these, Time Petri nets

(TPNs) are most widely used for real-time system specification and verification. It has been shown in [BD91] that TPNs are sufficient to express most useful temporal constraints and are also more expressive than Ramchandani's Timed Petri nets.

First, we recall the definition of Time Petri Nets. The definitions here are mainly based on [PZ91, PZ98].

Definition 1 (Time Petri Net).
A Time Petri Net *is a 5-tuple* $TN = \langle P, T, W, M_0, I_s\rangle$ *where:*

1. P *and* T *are non-empty finite sets of* places *and* transitions*, respectively. These sets are disjoint:* $T \cap P = \emptyset$*;*
2. $W : (P \times T) \cup (T \times P) \longrightarrow \mathbb{N}$ *is the* arc-weight function*;*
3. $M_0 : P \longrightarrow \mathbb{N}$ *is the* initial marking*;*
4. $I_s : T \longrightarrow \mathbb{Q}^+ \times (\mathbb{Q}^+ \cup \{\infty\})$ *is the* static interval function*, where* $\mathbb{Q}^+$ *is the set of non-negative rational numbers. The* I_s *function associates with each transition* t *a temporal* static interval $[smin(t), smax(t)]$*, where* $smin(t)$ *and* $smax(t)$ *are called the* static minimal firing time *and the* static maximal firing time*, respectively.*

A marking is a function $M : P \longrightarrow \mathbb{N}$ where $M(p)$ denotes the number of tokens at place p. The pre- and post-sets of a transition t are given by ${}^\bullet t = \{p \in P/W(p,t) \geq 0\}$ and $t^\bullet = \{p \in P/W(t,p) \geq 0\}$.

The firing of a transition depends on both enabling and timing conditions.

Definition 2 (Enabling).
A transition t *is* enabled *in a marking* M *iff* $\forall p \in P, M(p) \geq W(p,t)$.

We denote by $enabled(M)$ the set of all transitions enabled in marking M. A transition t enabled in marking M is firable in the associated untimed Petri net but not necessarily in the time Petri net. This is denoted by $M[t\rangle_u$.

When firing a transition t, the transitions in conflict with t will first be disabled and may be re-enabled immediately in the new marking.

Definition 3 (Conflict).
Two transitions t_1 *and* t_2 *are* in conflict *in* M *iff* t_1 *and* t_2 *are enabled in* M *and* $\exists p \in {}^\bullet t_1 \cap {}^\bullet t_2, M(p) < W(p,t_1) + W(p,t_2)$.

As the firing of a transition is also conditioned by time, the TPN state must contain temporal information in addition to the marking. Two characterisations of TPN states are possible: clock state and interval state. The TPN clock state consists of a marking and a clock valuation function [PZ91]. Whereas the TPN interval state is a pair of marking and a firing interval [BD91]. In this paper we adopt the clock characterisation of states. In fact, we use a notion of pseudo-state as an abstraction (a cover) of clock state. The distinction between clock-state and clock pseudo-state is not in their definition but in their associated firing rules.

Definition 4 (Pseudo-State).
A pseudo-state *of a Time Petri Net is a pair $Z = (M, J)$, where M is a marking and $J : T \longrightarrow \mathbb{Q}^+ \cup \{\#\}$ is a* clock valuation function *where the symbol $\#$ is not in $\mathbb{Q}^+$.*

When a transition t becomes enabled, its clock is set to 0. Its evolution depends mainly on whether its static maximal firing time $smax(t)$ is equal or not to ∞. When $smax(t) < \infty$, the clock can go on until $smax(t)$ whereas it is stopped at $smin(t)$ if $smax(t) = \infty$. The transition t can fire if its clock is greater than or equal to its static minimal firing time $smin(t)$. But in the first case, when the clock reaches $smax(t)$, transition t must be fired immediately, without any delay. When a transition is disabled, its clock is set to $\#$, showing that the clock does not operate. The set of all pseudo-states is denoted by $\mathcal{Z}$.

Let $T_1 \subseteq T$ be a subset of transitions, J and J_1 clock valuation functions over T and T_1, respectively. We define the clock valuation function $J' = J[T_1/J_1]$ as follows :

$$J'(t) = J[T_1/J_1](t) = \begin{cases} J_1(t) & \text{if } t \in T_1 \\ J(t) & \text{otherwise} \end{cases}$$

Let J be a clock valuation function and dh a (non-negative) rational number, $J + dh$ is the clock valuation function obtained from J by increasing all the clocks of enabled transitions by the same delay dh.

The initial pseudo-state is $Z_0 = (M_0, J_0)$ where $\forall t \in enabled(M_0), J_0(t) = 0$ and $J_0(t) = \#$ otherwise. The pseudo-states of a TPN evolve, either due to time progressions or transitions firings.

Definition 5 (Behaviour). *Two kinds of* firings *are possible:*

- *In pseudo-state $Z = (M, J)$, a time delay $dh \in \mathbb{Q}^+$ can elapse, iff $\forall t \in enabled(M), J(t) + dh \leq smax(t)$. This leads to a new pseudo-state $Z' = (M', J')$ where:*
  1. $M' = M$
  2. $$\forall t \in T, J'(t) = \begin{cases} \# & \text{if } t \notin enabled(M') \\ J(t) + dh & \text{if } (t \in enabled(M') \wedge smax(t) < \infty) \\ & \vee (t \in enabled(M') \wedge smax(t) = \infty \\ & \quad \wedge J(t) + dh \leq smin(t)) \\ smin(t) & \text{otherwise} \end{cases}$$

  This kind of progression, when time elapses, is denoted by $Z \xrightarrow{dh} Z'$.
- *In pseudo-state $Z = (M, J)$, a transition t_f can fire iff it is both enabled and $J(t_f) \geq smin(t_f)$. In this case, the firing of transition t_f yields a new pseudo-state $Z' = (M', J')$ defined as follows:*
  1. $\forall p \in P, M'(p) = M(p) - W(p, t_f) + W(t_f, p)$
  2. $$\forall t \in T, J'(t) = \begin{cases} \# & \text{if } t \notin enabled(M') \\ J(t) & \text{if } t \in enabled(M) \wedge t \in enabled(M') \\ & \quad \text{and } t \text{ is not in conflict with } t_f \text{ in } M \\ 0 & \text{otherwise } (t \text{ is newly enabled in } M') \end{cases}$$

This kind of progression, by transition firing, is denoted by $Z \xrightarrow{t_f} Z'$.

For simplicity, we assume that no transition can be concurrently enabled with itself. Note that when a transition t_f fires, transitions in conflict with t_f are temporarily disabled. If they are enabled in the new marking, this is a new enabling and therefore their associated clock valuation becomes 0.

If $Z \xrightarrow{dh} Z'$ and $Z' \xrightarrow{t} Z''$, we write $Z[dh.t\rangle Z''$ or simply $Z[t\rangle Z''$ without indication about the time elapsed before t fires.

Let Z be a pseudo-state and $\sigma = t_1 t_2 \ldots t_n$ a finite sequence of transitions. Sequence σ is firable from Z, denoted by $Z[\sigma\rangle$, whenever there exists a finite sequence of pseudo-states Z_1, Z_2, $\ldots Z_n$ such that $Z = Z_1$ and $\forall i = 1..n : Z_i[t_i\rangle Z_{i+1}$. Hence, Z_{n+1} is reachable from Z by firing σ, denoted by $Z[\sigma\rangle Z_{n+1}$.

The firing rules above define a reachability relation among pseudo-states of a time Petri net. However, due to time density this set is in general infinite, even if it is bounded. Its analysis by enumerative techniques requires some state space contractions (state class graph, strong state class graph, atomic state class graph) [BD91, Dia01]. In this paper, we consider a discrete-time model [PZ98] as a first step to deal with modularisation.

Definition 6 (Discrete pseudo-state). *A pseudo-state* $Z = (M, J)$ *is called a* discrete pseudo-state *iff* $\forall t \in enabled(M), J(t) \in \mathbb{N}$. *By* $[Z\rangle$ *we denote the set of all discrete pseudo-states reachable from* Z.

Definition 7 (Discrete pseudo-state graph). *Let* $TN = \langle P, T, W, M_0, I_s\rangle$ *be a Time Petri Net. Its* discrete pseudo-state graph *is* $GI = (NI, AI)$ *where :*

1. NI *is the set of all reachable discrete pseudo-states*
2. $AI = \{(Z, t, Z') \in NI \times T \times NI / Z \xrightarrow{t} Z'\} \cup \{(Z, 1, Z')/Z \xrightarrow{1} Z'\}$.

The discrete pseudo-state graph is finite if the time Petri net is bounded. Such a graph preserves qualitative properties such as liveness, boundedness, linear temporal properties, as well as quantitative properties which are important for most real-time systems [PZ98, YO95].

Example 1. The discrete pseudo-state graph of the TPN given in figure 1 is depicted in figure 2 (where ... denotes ∞). The transition firings are represented by $\longrightarrow$ and time progressions by $\dashrightarrow$. In pseudo-state 2, tb_1 is the only firable transition. As its static maximal firing time is equal to ∞, it may fire at any time. This fact is represented in the pseudo-state space by a unit time progression arc to state 2.

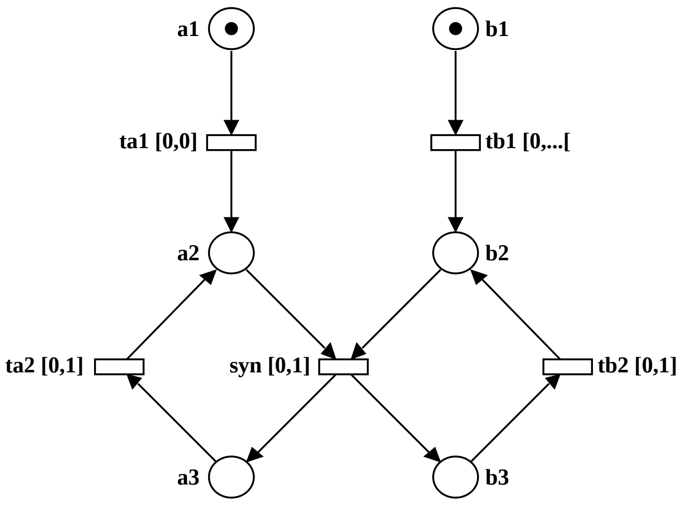
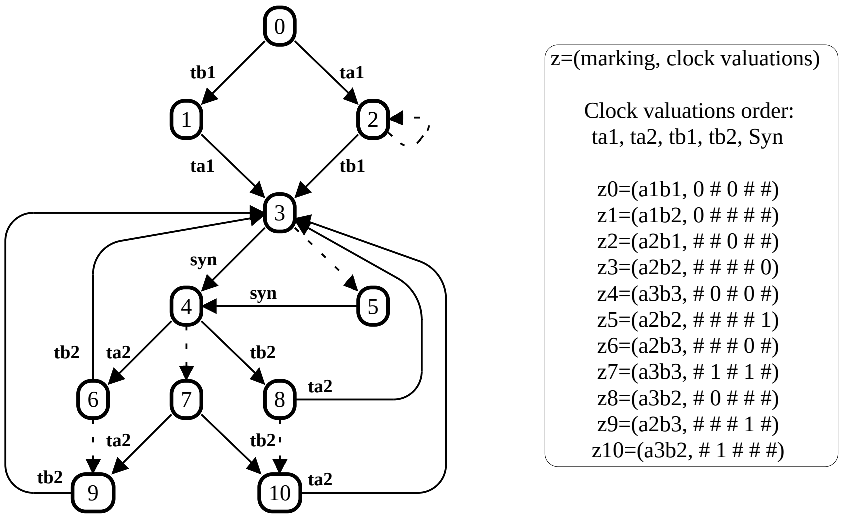


Fig. 1. Time Petri net N_1



z=(marking, clock valuations)

Clock valuations order:
ta1, ta2, tb1, tb2, Syn

z0=(a1b1, 0 # 0 # #)
z1=(a1b2, 0 # # # #)
z2=(a2b1, # # 0 # #)
z3=(a2b2, # # # # 0)
z4=(a3b3, # 0 # 0 #)
z5=(a2b2, # # # # 1)
z6=(a2b3, # # # 0 #)
z7=(a3b3, # 1 # 1 #)
z8=(a3b2, # 0 # # #)
z9=(a2b3, # # # 1 #)
z10=(a3b2, # 1 # # #)

Fig. 2. Discrete pseudo-state graph of TPN N_1

3 Modular Time Petri Nets

In this section, we define the modular time Petri net model. A modular time Petri net is composed of a finite set of modules. Each module is a time Petri net. The communication between modules is based on transition fusion sets and synchronisation rules. These rules, taken from Time Stream Petri

Nets [DS95], are introduced in order to solve temporal composition problems inside transition fusion sets.

Definition 8 (Modular Time Petri Net).
A Modular Time Petri Net *(MTPN) is a triple $MTN = (S, TF, Syn)$, where :*

1. *S is a finite set of* modules *such that:*
   a) *Each module $s \in S$ is a Time Petri Net $s = \langle P_s, T_s, W_s, M_{0s}, I_s \rangle$.*
   b) *The sets of nodes corresponding to different modules are pair-wise disjoint : $\forall s_1, s_2 \in S, s_1 \neq s_2 \Rightarrow (P_{s_1} \cup T_{s_1}) \cap (P_{s_2} \cup T_{s_2}) = \emptyset$.*
   c) *$P = \bigcup_{s \in S} P_s$ and $T = \bigcup_{s \in S} T_s$ are the sets of all places and all transitions of all modules.*
   d) *For a node $x \in (T \cup P)$, $S(x)$ denotes the module to which x belongs. We define $\forall p \in P : M_0(p) = M_{0S(p)}(p)$ and $\forall t \in T : I(t) = I_{S(t)}(t)$.*
2. *$TF \subseteq 2^T$ is a finite set of non-empty* transition fusion sets.
3. *$Syn : TF \longrightarrow \{"or", "strong - or", "weak - and", "pure - and"\}$ is the fusion temporal synchronisation function. The semantics of these synchronisation rules is illustrated in figure 3.*

Let $tf = \{t_1, t_2, t_3\}$ be a fused transition
$I_s(t_1) = [a_1, b_1], I_s(t_2) = [a_2, b_2]$, and $I_s(t_3) = [a_3, b_3]$

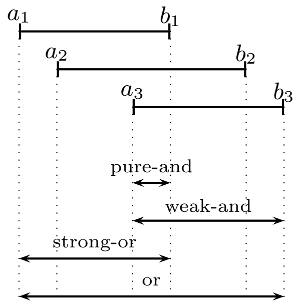


Fig. 3. Semantics of Synchronisation

For the sake of simplicity, we consider only the pure-and synchronisation rule. In the following, TF also denotes $\cup_{tf \in TF} tf$. The set $IT = T \setminus TF$ contains all *internal transitions*, i.e. non-fused transitions and IT* is a set of all finite sequences of internal transitions. IT_s is its restriction to module s.

Example 2. The system modeled by the time Petri net N_1 in figure 1 can be described in a modular way as shown in figure 4. It is composed of two modules, namely A and B, which share transition *syn*, corresponding to a synchronous action. Note that the transitions to be fused have the same name. In this example, there is only one transition fusion set. The two transitions fused together have different temporal static intervals in their respective modules.

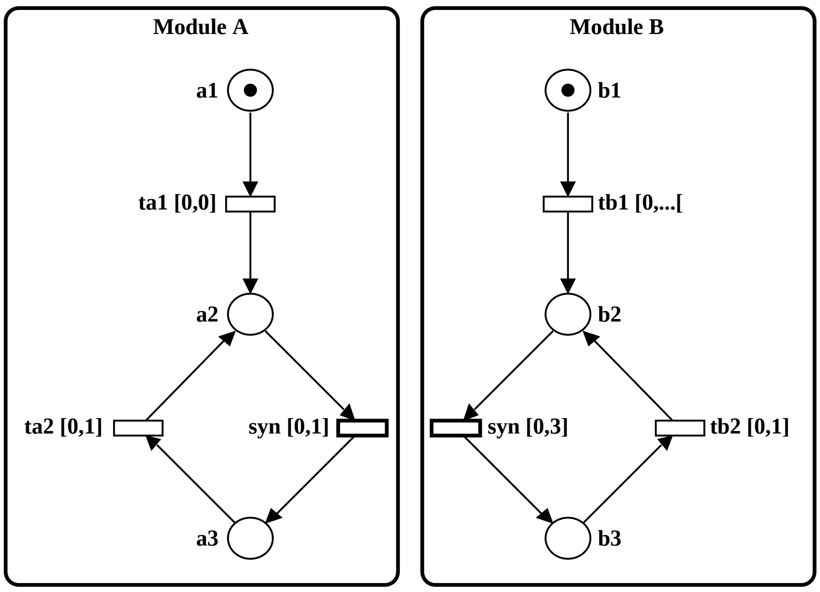


Fig. 4. Time Petri net N_1, composed of 2 modules

Since there are both fusion transitions and internal transitions in a modular net, we introduce a notion of transition group as a uniform way to refer to them.

Definition 9 (Transition group).
A transition group $tg \subseteq T$ *is a set of transitions such that it consists of a single non-fused transition* $t \in IT = T \setminus TF$ *or is equal to a transition fusion set* $tf \in TF$*. The set of transition groups is denoted by* TG*.*

Next, we extend the arc-weight function W and the static interval function I to transition groups:

1. $\forall p \in P, \forall tg \in TG : (W(p, tg) = \sum_{t \in tg} W(p, t)) \wedge (W(tg, p) = \sum_{t \in tg} W(t, p))$.
2. The static temporal interval of a transition group tg is computed with respect to its temporal synchronisation rule. As mentioned above, we consider only the pure-and rule. Hence, $\forall tg \in TG, \forall t \in tg : (smin(tg) = \max_{t \in tg} smin(t)) \wedge (smax(tg) = \min_{t \in tg} smax(t))$.

Definition 10 (Enabling).
A transition group tg is enabled *in a marking M, denoted by $M[tg\rangle$, iff:*
$\forall p \in P, W(p, tg) \leq M(p)$.

A pseudo-state of a modular time Petri net is defined in a similar manner as for time Petri nets. Hence, it is a pair of marking and clock valuation function that associates with each enabled transition group its local time and # with each disabled transition group. The firing rules are also similar where transition is replaced by transition group.

A modular time Petri net can be unfolded into an equivalent time Petri net.

Definition 11 (Equivalent Time Petri Net).
Let $MTN = (S, TF, Syn)$ be a modular time Petri net. Its equivalent time Petri net *is $TN^\diamond = (P^\diamond, T^\diamond, W^\diamond, M_0^\diamond, I^\diamond)$ where:*

1. $P^\diamond = P$
2. $T^\diamond = TG$
3. $\forall (x^\diamond, y^\diamond) \in (P^\diamond \times T^\diamond) \cup (T^\diamond \times P^\diamond), W^\diamond(x^\diamond, y^\diamond) = W(x^\diamond, y^\diamond)$
4. $\forall p^\diamond \in P^\diamond, M_0^\diamond(p^\diamond) = M_0(p^\diamond)$
5. $\forall t^\diamond \in T^\diamond, I^\diamond(t^\diamond) = I(t^\diamond)$

The set of places is preserved as well as their initial marking. The equivalent time Petri net has one transition per transition group. The weight associated with a pair (place, transition group) or (transition group, place) and the static temporal interval of a transition group are unchanged.

The following property states that a modular time Petri net and its equivalent time Petri net have the same behaviour.

Proposition 1. *Let MTN be a modular time Petri net and $TN^\diamond$ its equivalent time Petri net. Then, the following properties hold:*

1. $Z_0 = Z_0^\diamond$ *and* $\mathcal{Z} = \mathcal{Z}^\diamond$
2. $\forall Z_1, Z_2 \in \mathcal{Z}, \forall x \in TG \cup \mathbb{N} : Z_1 \xrightarrow{x}_{MTN} Z_2 \Leftrightarrow Z_1 \xrightarrow{x}_{TN^\diamond} Z_2$

Proof:

Follows from definitions 8, 11 and 5. $\diamond$

4 Modular Time Pseudo-State Graphs

In the following, we formally define modular discrete pseudo-state graphs.

Let σ be a finite sequence of internal transitions, Z a state and tf a fused transition. We introduce the following notations:

1. $[[Z\rangle = \{Z'/\exists\sigma \in IT^*, Z[\sigma\rangle Z'\}$
is the set of all pseudo-states reachable from Z by a sequence (possibly empty) of internal transitions only ($Z \in [[Z\rangle$). Note that a delay can occur before each transition in σ, but not after. Hence, the sequence always ends with an internal transition.
2. $[[Z\rfloor = \{Z'/\exists dh \in \mathbb{N}, \exists Z'' \in [[Z\rangle : Z'' \xrightarrow{dh} Z'\}$
is the set of all pseudo-states reachable from Z by a sequence (possibly empty) of internal transitions possibly followed by progressions of time. Note that $[[Z\rangle \subseteq [[Z\rfloor$.
3. Among the pseudo-states in sets $[[Z\rangle$ and $[[Z\rfloor$, we distinguish those where at least one fused transition is enabled. These subsets are denoted by $[[Z\rangle_{TF}$ and $[[Z\rfloor_{TF}$, respectively.

The set of transitions is augmented with an *abstract transition* denoted i, which represents an internal evolution of the net leading to a pseudo-state where at least one fused transition is enabled even if it is not firable. These states are relevant because time progression must be handled in a global manner. In the sequel, we refer to this kind of sequence as maximal sequence. Hence, we have chosen to represent explicitly this kind of evolution even if we can capture this information on the arcs as in [CP00]. The restriction of a pseudo-state $Z = (M, J)$ to a module s is denoted by $Z_s = (M_s, J_s)$ where M_s (respectively J_s) is the restriction of M (respectively J) to places (respectively internal transitions) of module s. For a local pseudo-state Z_s in a module s, $[Z\rangle_s$ denotes the set of all local pseudo-states reachable from Z_s by occurrences of local transitions of module s only.

Definition 12 (Modular Pseudo-State Graph). *Let $MTN = (S, TF, Syn)$ be a modular time Petri net. The* modular pseudo-state graph *of MTN is a pair $MPSG = ((PSG_s)_{s\in S}, SG)$ where:*

1. $PSG_s = (V_s, A_s)$ is the local pseudo-state graph *of module s defined by:*
   a) $V_s = \cup_{v\in(V_{SG})_s}[v\rangle_s$
   b) $A_s = \{(Z, t_s, Z') \in V_s \times IT_s \times V_s / Z \xrightarrow{t_s} Z'\}$
   $\quad\cup\{(Z, 1, Z') \in V_s \times \{1\} \times V_s / Z \xrightarrow{1} Z'\}$

2. $SG = (V_{SG}, A_{SG})$ is the synchronisation graph *of MTN:*
   a) V_{SG} can be defined inductively as follows:
      i. $Z_0 \in V_{SG}$
      ii. if $Z \in V_{SG}$ and $Z \xrightarrow{tf} Z'$ then $Z' \in V_{SG}$
      iii. if $Z \in V_{SG}$, $\exists tf \in TF$ enabled in Z and $Z \xrightarrow{1} Z'$ then $Z' \in V_{SG}$
      iv. if $Z \in V_{SG}$ and $Z' \in [[Z\rangle_{TF}$ then $Z' \in V_{SG}$
   b) $A_{SG} = \{(Z, tf, Z') \in V_{SG} \times TF \times V_{SG} / Z \xrightarrow{tf} Z'\}$
   $\quad\cup\{(Z, 1, Z') \in V_{SG} \times \{1\} \times V_{SG} / \exists tf \in TF$ enabled in $Z \wedge Z \xrightarrow{1} Z'\}$
   $\quad\cup\{(Z, i, Z')/Z' \in [[Z\rangle_{TF}\}$.

Explanation:

A modular state space is composed of one local pseudo-state graph per module and a synchronisation graph.

(1) The local pseudo-state graph of a module represents its internal activity independently of the other modules.
  (1a) The set of nodes of the pseudo-state graph of a module s contains all pseudo-states locally reachable from any node of the synchronisation graph.
  (1b) The arcs of the pseudo-state graph of a module correspond to internal firable transitions or local time progression of the module.

(2) The synchronisation graph captures the communication between modules.
  (2a) The nodes of the synchronisation graph contains in addition of the initial pseudo-state all pseudo-states reachable by:
   (2(a)ii) a fused transition from another pseudo-state
   (2(a)iii) time progression from another pseudo-state where at least one fused transition is enabled
   (2(a)iv) a sequence of internal transitions from another pseudo-state. Those states have at least one enabled fused transition.
  (2b) The arcs of the synchronisation graph represent the occurrences of a fused transition, of time progression and of internal evolution.

To prove that the modular state graph captures the behaviour of the Time Petri net, we first define the unfolding of a modular pseudo-state graph into an ordinary one. A global pseudo-state can be viewed as the product of local pseudo-states plus temporal information concerning the fused transitions. For this purpose, we define the following notations :

1. Let M_s be a marking of module s, M_s^* denotes the marking of the full system where all places of all other modules are empty.
2. Let J_F be a clock valuation function mapping from TF. Firing an internal transition t_s in a marking M can modify the marking and consequently enable or disable fused transitions. The new clock valuation function of fused transitions $J'_{F\#}$ is then (M' being the new marking):

$$\forall tf \in TF, J'_{F\#}(tf) = \begin{cases} J_F(tf) & \text{if } tf \in \mathit{enabled}(M) \\ & \quad \text{and } tf \text{ is not in conflict with } t_s \\ \# & \text{if } tf \notin \mathit{enabled}(M') \\ 0 & \text{otherwise } (tf \text{ is newly enabled}) \end{cases}$$

Definition 13 (Unfolded pseudo-state graph). *Let $MTN = (S, TF, Syn)$ be a modular time Petri net and $MPSG = ((PSG_s)_{s\in S}, SG)$ its modular pseudo-state graph. The* unfolded pseudo-state graph *of MPSG is $PSG = (V, A)$ where:*

1. $V = \bigcup_{v \in V_{SG}} [[v\rfloor$

2. $$\begin{aligned} A = &\{(Z, tf, Z') \in A_{SG}\} \\ \cup &\bigcup_{\substack{Z \in V, s \in S, \\ (Z_s, t, Z'_s) \in A_s}} \{(Z, t, Z') / M' = M - M_s^\diamond + M_s'^\diamond \\ &\qquad \wedge J' = J[IT_s/J'_s, TF/J'_{F\#}]\} \\ \cup &\bigcup_{Z \in V} \{(Z, 1, Z') / (Z \in V_{SG} \wedge (Z, 1, Z') \in A_{SG}) \\ &\qquad \vee [(\forall tf \in TF, tf \notin enabled(M)) \\ &\qquad\quad \wedge J'_F = J_F \wedge (\forall s \in S, (Z_s, 1, Z'_s) \in A_s))]\} \end{aligned}$$

Explanation:

(1) The unfolded graph contains all the pseudo-states reachable by internal transitions and time delays, from any of the nodes in the synchronisation graph.

(2) An arc of synchronisation graph labeled by a fused transition or 1 (representing time progression) is also an arc of the equivalent state graph. For each pseudo-state, if a local transition is firable, there is a corresponding arc in the unfolded graph. The marking and clock valuations are changed for the module concerned as specified in its local pseudo-state graph. The clock valuations of fused transitions are modified as specified previously. The firing of an internal transition cannot affect the internal transitions of other modules. For each pseudo-state without any enabled fused transition, if local time progression can occur for each module, there is a corresponding time progression arc in the unfolded graph. The pseudo-state for each module is changed as specified in its local pseudo-state graph. The clock valuations of fused transitions are unchanged, i.e. equal to #.

The following theorem states that the equivalent ordinary pseudo-state graph of $MPSG$ and the pseudo-state graph of the equivalent time Petri net of MTN are the same.

Theorem 1. *Let MTN be a modular time Petri net, $MPSG$ its modular pseudo-state graph and TN its equivalent time Petri net. Let PSG_{MPSG} be the unfolded pseudo-state graph of $MPSG$ and PSG_{TN} the pseudo-state graph of TN:*

$$PSG_{MPSG} \text{ and } PSG_{TN} \text{ are isomorphic.}$$

Proof:

Let $MTN = (S, TF)$ be a modular time Petri net, $PMSG = ((PSG_s)_{s \in S}, SG)$ its modular pseudo-state graph and TN its equivalent time Petri net. Let $PSG_{MPSG} = (V_{MPSG}, A_{MPSG})$ be the unfolded pseudo-state graph of $MPSG$ and $PSG_{TN} = (V_{TN}, A_{TN})$ the pseudo-state graph of TN.

To prove that PSG_{MPSG} and PSG_{TN} are isomorphic, we must find a bijective function f mapping from V_{MPSG} to V_{TN} such that: $\forall Z, Z' \in V_{MPSG}, \forall x \in TG \cup \{1\}, (Z, x, Z') \in A_{MPSG}$ iff $(f(Z), x, f(Z')) \in A_{TN}$. We can first prove $V_{TN} = V_{MPSG}$ and then we can take the identity as function f. We can proceed by double inclusion.

1. ($\subseteq$) Let $Z \in V_{TN}$. We prove that $Z \in V_{MPSG}$.
   We proceed by induction on the structure of V_{TN}. The base case is trivial. Let $Z \in V_{TN}$ (with $Z \neq Z_0$)). Two cases are possible for the induction:
   a) $\exists Z' \in V_{TN}, \exists t \in TG, Z' \xrightarrow{t}_{TN} Z$.
      From proposition 1, $Z' \xrightarrow{t}_{TN} Z \Leftrightarrow Z' \xrightarrow{t}_{MTN} Z$.
      By induction, $Z' \in V_{MPSG}$ and thus $Z \in V_{MPSG}$.
   b) $\exists Z' \in V_{TN}, Z' \rightarrow_{TN} Z$. This case is similar to the previous one.
2. ($\supseteq$) Let $Z \in V_{MPSG}$. We prove that $Z \in V_{TN}$:
   $Z \in V_{MPSG}$
   $\Leftrightarrow \exists Z' \in V_{SG}, Z \in [[Z'\rfloor$ (definition 13)
   $\Leftrightarrow \exists Z' \in V_{SG}, \exists \sigma \in IT^*, \exists dh \in \mathbb{N} : Z'[\sigma.dh\rangle_{MTN} Z$.
   It is trivial that $V_{SG} \subseteq V_{MPSG}$ (all nodes of the synchronisation graph are in the unfolded pseudo-state graph of $MPSG$). We must prove that $V_{SG} \subseteq V_{TN}$ (all nodes of synchronisation graph are also in the pseudo-state graph of TN).
   We can proceed by induction on the structure of V_{SG} that $V_{SG} \subseteq V_{TN}$. It follows that $V_{MPSG} \subseteq V_{TN}$. From proposition 1, we have $Z'[\sigma.dh\rangle_{MTN} Z \Leftrightarrow Z'[\sigma.dh\rangle_{TN} Z$ and from definition 7, we can conclude that $Z \in V_{TN}$.

Now, me must prove the property for the arcs:

$\forall Z, Z' \in V_{MPSG}, \forall x \in TG \cup \{1\} : (Z, x, Z') \in A_{MPSG}$ iff $(f(Z), x, f(Z')) \in A_{TN}$

1. ($\Rightarrow$) Let $Z, Z' \in V_{MPSG}$ and $x \in TG \cup \{1\}$ such that $(Z, x, Z') \in A_{MPSG}$. We prove that $(Z, x, Z') \in A_{TN}$.
   a) $x \in TF$ (x is a fused transition)
      $x \in TF$ and $(Z, x, Z') \in A_{MPSG}$
      $\Leftrightarrow x \in TF$ and $(Z, x, Z') \in A_{SG}$ (definition 13)
      $\Leftrightarrow x \in TF$ and $Z \xrightarrow{x}_{MTN} Z'$ (definition 12)
      $\Leftrightarrow x \in TF$ and $Z \xrightarrow{x}_{TN} Z'$ (proposition 1)
      $\Leftrightarrow x \in TF$ and $(Z, x, Z') \in A_{TN}$ (definition 7)
   b) $x = 1$ (time progression). There are two subcases:
      i. $Z \in V_{PSG}$ and $(Z, 1, Z') \in A_{MPSG}$ (there is at least one enabled fused transition).
         As $Z \in V_{PSG}$, it follows from definitions 13, 12, 7 and proposition 1 that $(Z, 1, Z') \in A_{TN}$.
      ii. $Z \notin V_{SG}$ and $(Z, 1, Z') \in A_{MPSG}$ (there is no enabled fused transition)

$\Leftrightarrow \nexists tf \in TF, tf \in \text{enabled}(Z)$, and $\forall s \in S, (Z_s, 1, Z'_s) \in A_s$
$\Leftrightarrow \text{enabled}(Z) = \cup_{s \in S}\text{enabled}(Z_s), J'_F = J_F = \#, \forall s \in S, (Z_s \rightarrow_{MTN} Z'_s)$
$\Rightarrow Z \rightarrow_{MTN} Z'$
$\Rightarrow Z \rightarrow_{TN} Z'$
$\Rightarrow (Z, 1, Z') \in A_{TN}$

c) $x = \{t_s\}$ with $t_s \in IT$ (x is an internal transition)
$\Rightarrow \exists s \in S, (Z_s, t_s, Z'_s) \in A_s$ and $Z' = (M - M_s^{\diamond} + M_s'^{\diamond}, J[IT_s/J'_s, TF/J_{F\#}])$
$\Rightarrow Z \xrightarrow{t_s}_{MTN} Z'$
$\Rightarrow Z \xrightarrow{t_s}_{TN} Z'$
$\Rightarrow (Z, t_s, Z') \in A_{TN}$

2. ($\Leftarrow$) Let $Z, Z' \in V_{TN} (= V_{MPSG})$ and $x \in TG \cup \{1\}$ such that $(Z, x, Z') \in A_{TN}$. We prove that $(Z, x, Z') \in A_{MPSG}$.
   a) $x \in TF$ (x is a fused transition). This case has already been handled in (1a).

   b) $x = 1$ (time progression). The case where $Z \in V_{SG}$ has been handled previously. The case where $Z \notin V_{SG}$ remains.
   $Z \notin V_{SG}$ and $(Z, 1, Z') \in A_{TN}$
   $\Rightarrow Z \notin V_{SG}$ and $Z \rightarrow_{TN} Z'$
   $\Rightarrow Z \notin V_{SG}$ and $Z \rightarrow_{MTN} Z'$
   $\Rightarrow$ There is no fused transition enabled at Z, $J_F = J'_F$ and $\forall s \in S, (Z_s, 1, Z'_s) \in A_s \Rightarrow (Z, 1, Z') \in A_{MPSG}$.
   c) $x = t_s$ with $t_s \in IT$. This case can be handled in a similar manner.

$\diamond$

5 Algorithm for Modular Time Petri Nets

In this section, we present an algorithm that constructs a modular discrete pseudo-state graph for a given modular Time Petri net. Before giving the construction algorithm, we start with an informal explanation on an example.

Example 3. Figures 5, 6 and 7 show the steps of the modular state space generation for the net N_1 in figure 4.

The construction starts by initialising the synchronisation graph with the first node corresponding to the initial state Z_0. In the first step, the local graphs contain all locally reachable pseudo-states of each module, from the initial state. Note that for each module, the local valuation clock of internal transitions is computed relative to the static firing intervals of its enabled internal transitions only, even if the marking enables fused transitions. For example, in state A_2 no internal transition is enabled so time can progress indefinitely. The fused transition *syn* is locally enabled but it may or may not be globally enabled since its enabling depends on both modules A and B.

In a second step, we find from the graphs of the modules the global states which enable at least one fused transition. These states are added to the synchronisation graph, linked to the previous state by an abstract transition i. The valuation clock of enabled fused transitions is computed from local valuation clocks attached to participating modules. Note that once a fused transition is locally enabled its local valuation clock is activated. This temporal information represents time elapsed since the module is ready to participate to a fused transition. In fact, a fused transition is really enabled if it is locally enabled in all participating modules. In this example, the combination of states $A_2 = (a_2, \#\ \#)$ and $B_2 = (b_2, \#\ \#)$ enables the fused transition *syn*. Its valuation clock is set to 0 because it becomes enabled.

Hence, we add node $(A_2B_2, 0)$ to the synchronisation graph, linked to the initial node by an arc labeled i. In this state, *syn* can be fired either immediately or after one time unit, leading to state $(a_3b_3, \#\ 0\ \#\ 0\ \#)$. The corresponding nodes are added to the synchronisation graph and the new local nodes to the local graphs. Thus, state $A_3 = (a_3, \#\ 0)$ and all states locally reachable from there are added to the local graph of module A. Transition ta_2 is enabled and can be fired yielding to an existing state A_2 or time can progress leading to a new state $A_4 = (a_3, \#\ 1)$. A similar construction is applied to module B.

Then, in a third step, state $(A_2B_2, 0)$ is locally reachable from state $(A_3B_3, \#)$ and we link them by an arc labeled i. Nothing new can then be added and the graph construction is finished.

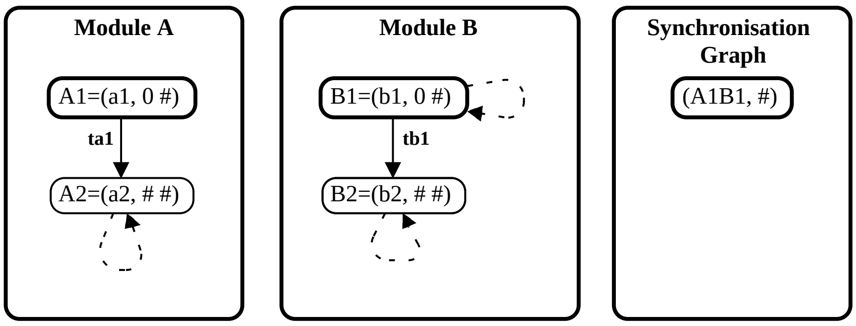


Fig. 5. After the first step of the modular generation

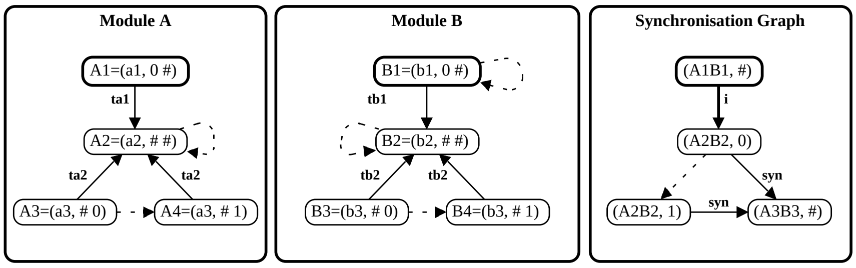


Fig. 6. After the second step of the modular generation

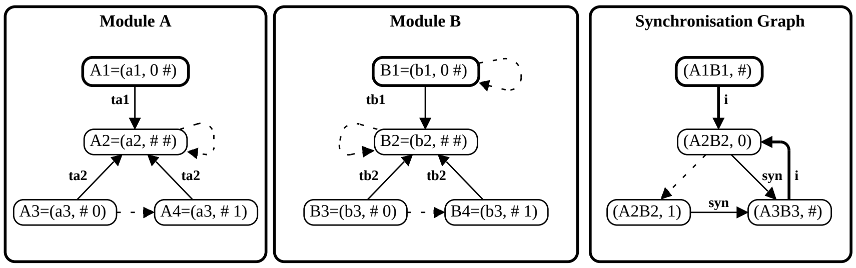


Fig. 7. After the third step of the modular generation

5.1 Modular state space generation algorithm

We now define an abstract algorithm to construct modular pseudo-state graphs. This algorithm does not represent abstract transition in the synchronisation graph and this information is captured on arcs as in [CP00]. The functions Node.Add(Z) and Arc.Add(Z, a, Z') are similar to those in [CP00]. Function Node.Add(Z) adds a node labelled by Z to the graph and to the set *Waiting*, provided it does not already exist. Similarly, Arc.Add(Z, a, Z') adds an arc to the graph. Function LocalNextI(Z) constructs the different local state spaces stepwise and produces the set of all pseudo-states reachable from Z by a sequence of internal transitions (or none). It may be followed by a fused transition (either immediately or after a delay). This function uses two other ones: Explore-Internal(`s, current-waiting`$_s$) and Explore-Succ(`s, succ-waiting`$_s$). The first one explores for each module the local pseudo-states reachable from any pseudo-state of its set `current-waiting`$_s$ before time is advanced and returns a set of pairs: the first element is the local pseudo-state reachable from `current-waiting`$_s$ pseudo-state and the second is the fused transition which is locally firable at this reachable pseudo-state. The second function gives the set of all local temporal successor pseudo-states of any `succ-waiting`$_s$ pseudo-state after one time unit. This set will be examined in the next step of the loop as the `current-waiting`$_s$ set. Variable `succ-waiting`$_s$ contains local successor pseudo-states of any pseudo-state of `current-waiting`$_s$. Node.Add (at the local level) uses an extended pseudo-state Z_{es} which is a pair of local pseudo-state and local valuation clocks of fused transitions. It adds the argument to both `current-waiting`$_s$ and `succ-waiting`$_s$, and also adds a node labelled by the first element only to the local pseudo-state graph. LocalNextI(Z) terminates when stability is reached: no new pseudo-state is found. Set `succ-waiting`$^+$ contains the local pseudo-states reachable by a non-empty sequence of internal transitions. Hence it is a subset of `succ-waiting`.

```
1: Set waiting $\longleftarrow \emptyset$
2: Node.Add($Z_0$)
3: **repeat**
4:   Select $Z \in$ waiting ;
5:   Next-i $\longleftarrow$ LocalNextI($Z$);
6:   **for all** $Z' \in$ Next-i **do**
7:     **for all** $tf \in TF$ **do**
8:       **if** $Z' \xrightarrow{tf} Z"$ **then**
9:         Node.Add($Z"$)
10:        Arc.Add($Z, (Z', tf), Z"$)
11:      **end if**
12:    **end for**
13:    **if** $Z' \longrightarrow Z"$ and $\exists tf \in TF, Z'[tf\rangle_u$ **then**
14:      Node.Add($Z"$)
15:      Arc.Add($Z, (Z', 1), Z"$)
16:    **end if**
17:  **end for**
18: **until** stable (waiting=$\emptyset$)
```

Fig. 8. Algorithm MSS() (modular state space generation)

5.2 Correctness

The correctness of the algorithm is mainly based on the correctness of function LocalNextI(Z). Indeed, this function returns a set of global pseudo-states reachable from Z by a maximal sequence of internal transitions: $[[Z\rangle_{TF}$. Computing this set from local pseudo-states only can lead to non-reachable global pseudo-states. The main problem is to find correct valuation clocks of fused transitions. For this purpose, an extended local pseudo-state is used where local valuation clocks of fused transitions are associated with local pseudo-state. Once a fused transition *tf* is locally enabled in a module s, its local clock is initialised and increases when time progresses. It represents time elapsed since module s is ready to participate to fused transition *tf*. The different local clocks of a fused transition are consistent since the time progression is applied unit by unit for all modules. From local extended pseudo-states of modules $(Z_{es})_{s \in S}$, we can compute a global pseudo-state Z'. We can deduce that a fused transition *tf* is globally enabled in Z' when it is locally enabled in all participating modules and in this case its global clock is set to the local clock associated with *tf* in the last module ready to participate to it. Since time progresses in modules independently of time constraints of fused transitions, we have to exclude from the global pseudo-states obtained those where for a fused transition *tf* the valuation clock is greater than its maximal firing time. We also exclude those where no fused transition is enabled because we are interested in maximal sequences only. Finally, we must verify that there exists a non-empty sequence of internal transitions yielding to Z' since time is at least increased by a unit. All these conditions guarantee that Z' is reachable from Z by a maximal sequence of internal transitions ending with an internal transition.

The algorithm MSS() uses a set *Waiting* of global pseudo-states from which it may be possible to reach states by a maximal sequence of internal transitions, where at least a fused transition is enabled. Function LOCALNEXTI(Z) computes this set for each state Z of *Waiting*. If from Z, we can reach a state Z' where a fused transition is enabled, its successors are built and added to set *Waiting* provided that they do not already exist. Each state handled is removed from *Waiting*. Hence, the termination of the algorithm MSS() depends on the termination of LOCALNEXTI(Z). This function terminates when there is no new pseudo-state in any module.

6 Experimental results

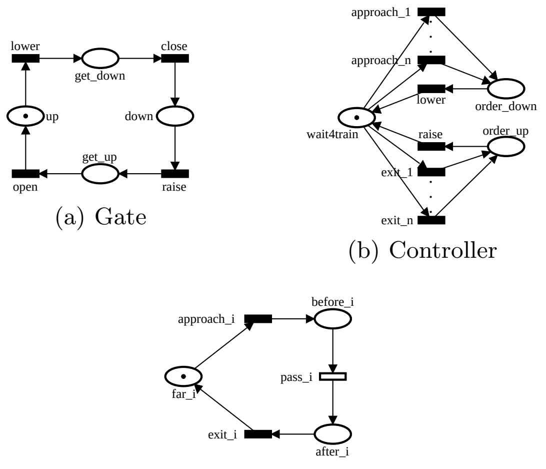
In order to show the benefits of our approach, we implemented a prototype tool. In this paper, we consider the example of a timed version of the railroad crossing problem from [Pet05]. The railroad crossing system (RCS) operates a gate. The area of crossing lies within a region of interest. When a train enters or leaves this region, the RCS is notified. The gate is modelled in figure 10(a), each train is described by a net presented in figure 10(c), while the whole system is synchronised through the controller shown in figure 10(b). The transitions with the same name are to be fused. The static interval firing times of transitions are given in table 1.

The results of the experiments are given in table 2. The first two columns give the size (number of nodes and arcs) of the flat pseudo-state graph whereas the next two columns give the size of the modular discrete pseudo-state graph.

The experimental results show that the more trains, the more significant the reduction is. Indeed, much interleaving is avoided when building the modular pseudo-state space. Note that in this system communication does not

```
1: Set current-Next-i $\longleftarrow Z$
2: for all $s \in S$ do
3:    current-waiting$_s \longleftarrow \emptyset$
4:    NODE.ADD($Z_s$)
5: end for
6: repeat
7:    for all $s \in S$ do
8:       trysyn$_s \longleftarrow$ EXPLORE-INTERNAL($s$,current-waiting$_s$)
9:    end for
10:   set-of-pairs= $\{(Z, Z')/Z \neq Z', Z' = \prod_{s \in S, Z'_{es} \in \text{succ-waiting}_s} Z'_{es}$ :
                  $(\exists s \in S, Z'_{es} \in$ succ-waiting$^{+}_s)$,
                  $(\exists tf \in TF, (tf \cap T_s) \neq \emptyset \Rightarrow (tf, Z'_s) \in$ trysyn$_s)\}$
11:   for all $(Z, Z') \in$ set-of-pairs do
12:      current-Next-i $\leftarrow$ current-Next-i $\cup\{Z'\}$
13:   end for
14:   {Time progression}
15:   for all $s \in S$ do
16:      current-waiting$_s$ = EXPLORE-SUCC($s$,succ-waiting$_s$)
17:   end for
18: until stable
19: return current-Next-i
```

Fig. 9. Algorithm LOCALNEXTI(Z)



(a) Gate

(b) Controller

(c) Train i

Fig. 10. The railway crossing system

involve all modules. Thus, the algorithm can be optimised so that the synchronisation graph does not explicit equivalent occurrences of fused transitions having different states for modules which do not participate in the synchronisation [LP04]. We could also use the notion of strongly connected compo-

Module	Transition	smin(t)	smax(t)
Gate	lower	0	∞
	raise	0	∞
	close	1	2
	open	1	2
Controller	lower	0	0
	raise	0	0
	approach-i	0	∞
	exit-i	0	∞
train-i	approach-i	0	∞
	exit-i	0	0
	pass-i	3	5

Table 1. Static Interval Firing Times

nents as a compact representation to reduce the size of the synchronisation graph [CP00].

Trains	V_{FSG}	A_{FSG}	V_{MSG}	A_{MSG}
1	33	51	33	43
2	150	278	142	269
3	358	690	274	533
4	687	1287	453	891

Table 2. Modular discrete graph and flat graph

7 Conclusion and Future Work

In this paper, we have extended the definition of modular Petri nets to time Petri nets in order to take advantage of the modular structure of real-time system specifications. We have restricted the time domain to integral time to examine whether the modular approach can be translated to time Petri nets and still be beneficial. A fundamental hypothesis on the modular Time Petri net model is that time is synchronic, i.e. time progresses at the same pace for all modules. A modular pseudo-state graph was defined, composed of one local pseudo-state graph per module and a synchronisation graph. It maintains both global and local temporal information. The local temporal information is relevant when there is no enabled fused transition. The main problem was to find the reachable global pseudo-states which enable fused transitions from the graphs of modules.

An algorithm to generate a modular pseudo-state graph was given and a prototype tool supporting this approach was implemented. Preliminary experimental results have been presented and demonstrate the benefits of the approach. It will be important to perform further experiments to see whether this benefit carries over to more realistic case studies. Verifying properties on the modular pseudo-state graph without unfolding to an ordinary pseudo-state graph is still an important issue. Extending this approach to dense time constitutes a further challenge.

References

[BB98] H. Boucheneb and G. Berthelot. Composition des réseaux de Petri temporels. *Technique et Science Informatiques*, 17(6), 1998.

[BD91] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using Time Petri Nets. *IEEE Transactions on Software Engineering*, 17(3):256–273, March 1991.

[BV95] G. Bucci and E. Vicaro. Compositional validation of time-critical systems using communicating time Petri nets. *IEEE Transactions on Software Engineering*, 21(12), December 1995.

[CP00] S. Christensen and L. Petrucci. Modular analysis of Petri nets. *The Computer Journal*, 43(3):224–242, 2000.

[Dia01] M. Diaz. *Les réseaux de Petri : Modèles fondamentaux.* Hermès, 2001.

[DS95] M. Diaz and P. Sénac. Time stream Petri nets: A model for timed multimedia information. In *Proc. 15th Int. Conf. Application and Theory of Petri Nets (ICATPN'94), Zaragoza, Spain, June 1994*, volume 815, pages 219–238. Springer, June 1995.

[LP04] C. Lakos and L. Petrucci. Modular analysis of systems composed of semiautonomous subsystems. In *Proc. 4th Int. Conf. on Application of Concurrency to System Design (ACSD'04), Hamilton, Canada, June 2004*, pages 185–194. IEEE Comp. Soc. Press, June 2004.

[LP05] C. Lakos and L. Petrucci. Distributed and modular state space exploration for timed Petri nets. In *Proc. Workshop on Practical Use of Coloured Petri Nets, Aarhus, Denmark*, pages 191–210, October 2005. Proceedings published as Report DAIMI-PB 576, Aarhus, DK.

[MF76] P. Merlin and D.J. Faber. Recoverability of communication protocols. *IEEE Transactions on Communications*, 24(9), 1976.

[Pet05] L. Petrucci. Cover picture story: Experiments with modular state spaces. *Petri Net Newsletter*, 68:Cover page and 5–10, April 2005.

[PZ91] L. Popova-Zeugmann. On time Petri nets. *Inform. Process. Cybernet(formerly: Elektronishe Informationsverbeitung Kybernetik)*, 27:227–244, 1991.

[PZ98] L. Popova-Zeugmann. Essential states in time Petri nets. Technical Report 96, Humboldt Universität Berlin, 1998.

[Ram74] C. Ramchandani. Analysis of asynchronous concurrent systems by timed Petri nets. Technical Report 120, Massachussets Institute for Technology, February 1974.

[Sif77] J. Sifakis. Use of Petri nets for performance evaluation. In *Measuring, Modeling and Evaluating Computer Systems*, pages 75–93. North-Holland, 1977.

[Wan98] J. Wang. *Time Petri Nets, Theory and application.* Kluwer Academic, 1998.

[YO95] T. Yoneda and Y. Okawa. Discrete analysis of time Petri nets. In *Proc. of PRFTS95*, pages 224–229, 1995.