Parameterized study of a Time Petri Net

David DELFIEU, Medesu SOGBOHOSSOU, Louis-Marie TRAONOUEZ
Institute of Research in Communication and Cybernetics of Nantes
Nantes, 44300, France

and

Sebastien REVOL
STMicroeclectronics Centre commun CNET
Crolles, 38826, France

Abstract

Temporal analysis of Time Petri Nets encounters combinatory explosion. Most of actual works tackle with optimization and try to contain explosion. In in a different approach, this paper proposes a method allowing to study a part of a Time Petri Net defined by a scenario. A scenario is an unordered set of transitions extracted from a Petri net. This scenario is analyzed by a logic process based on Linear Logic. The essential contribution of this paper is to conduct the analyze under a strong semantic hypothesis.

This approach allows to compute symbolic expressions that define the firing domain of the scenario. Moreover, this approach is based on a rewriting process, which allows to introduce parameters in the Petri nets that are also found in the firing domains.

Keywords: Time Petri Nets, Linear Logic, Parameterized time analysis, Strong semantics, Scenario.

1. INTRODUCTION

Analyzing timing constraints in real time systems is generally based on an exhaustive and costly calculus of the state space. In formal models, Petri nets provide a good compromise between expressivity, communicability (graphical support) and formal aspect, giving decidable algorithm in the case of bounded Petri nets. Time Petri Nets (TPN) [9] are widely used to specify and verify real time systems, but trying to calculate state space, face to the problem of state explosion, with non academic problems. Actual works try to contain this problem by improvements on data storage and data manipulation (for example with Difference Bounded Matrix, zone graph, ...) [8, 4]. Another approach is based on the unfolding of the Petri net [2]. This latter allows to detect loops and limit the study to an independent and consistent part of the net. Moreover, true parallelism inherent to the approach prevents the expression of all the interleavings.

This work keeps some of these ideas. If it does not intend to unfold a Petri net, it proposes a way to restrict the process of analyze to a set of runs called a *scenario*. As well, it considers a true parallelism. A major profit is that the method presented in this paper uses a parametric approach enabling to consider a partial specification.

Recent works [10] on this approach have introduced a formal framework based on Linear Logic. It allows to transform a linear algebra calculus in a proof system and to treat time as a symbolic parameter. Based on resources, this logical system allows to easily modelize state changes. But these works were limited to weak semantics of TPN, which rejects urgency and thus is useless in the design of real time systems. The contribution of this paper is to define an algorithm in a strong semantic context.

The first part presents the process of formalization of a real time application driving to a linear logic proof, in a weak semantic hypothesis. The second part describes the contribution of this paper which is to reintroduce strong semantics in the proof process.

2. TIME PETRI NETS AND LINEAR LOGIC: STATE OF THE ART

Time Petri Nets (TPN) [9] modelize in a formal way a wide range of real time applications. The analyze of these models is generally based on the calculus of state space (class graph [1] or zone graph [3]) and face to the problem of state explosion. The approach expressed in this paper proposes to traduce TPN in a logic system based on Linear Logic formulae. The temporal analysis is then achieved by a proof process (through sequent calculus).

2.1 Time Petri Nets

There are several ways to associate time to a Petri net. It can be associated to places, transitions or arcs. In Time Petri Nets, a time interval, called *enabling duration*, is associated to transitions (in the case of T-TPN). This interval represents the minimum and maximum delay needed after enabling the transition, to effectively realize the firing. This interval can possibly be open with an infinite upper bound. Formally, a TPN can be defined as follow:

Definition 1 (Time Petri Net) *A Time Petri Net is a tuple $N = (P,\ T, {}^\bullet(.),\ (.)^\bullet,\ M_0,\ \alpha,\ \beta)$ where:*

- *P is a set of* places,
- *T is a set of* transitions, *with $P \cap T = \emptyset$,*
- *${}^\bullet(.) \in (\mathbb{N}^P)^T$ is the* backward *incidence mapping,*
- *$(.)^\bullet \in (\mathbb{N}^P)^T$ is the* forward *incidence mapping,*
- *$M_0 \in \mathbb{N}^P$ is the* initial *marking,*
- *$\alpha \in (\mathbb{R}^+)^T$ and $\beta \in (\mathbb{R}^+ \cup \{\infty\})^T$ are functions that give to each transition, respectively its* soonest *firing time and* latest *firing time (with $\alpha \leq \beta$).*

Parameterized approach One the interest of the approach will be to treat parameters instead of numerical values. In that way, instead of considering the numerical values of the enabling duration of the transitions, for a transition $t_i \in T$, the approach will consider two parameters $d_{i_{min}}$ and $d_{i_{max}}$ to represent $\alpha(t_i)$ and $\beta(t_i)$. Then, it will be possible to evaluate the results by assigning a numerical value to each parameter. One of the benefit is that several different evaluations can be done without re-analyzing the TPN.

TPN semantics A *marking* M of a TPN is a mapping in $\mathbb{N}^P$ that associates with each place p the number $M(p)$ of *tokens* contained in this place. A transition t is *enabled* by a marking M if and only if $M \geq {}^\bullet t$. The firing of this transition will produce the new marking $M' = M - {}^\bullet t + t^\bullet$. TPN semantics are classically described in terms of transitions systems, with two types of transitions: a discrete transition which fires a transition of the net, and a continuous transition which elapses time. With the introduction of time in Petri nets, there exists different semantics to determine when a transition can fire. They can be classified in two categories: weak and strong firing semantics (called in this paper WFS and SFS).

A clock $\nu(t)$ is associated with each enabled transition. In a weak semantic, a transition t can be fired if $\alpha(t) \leq \nu(t) \leq \beta(t)$, whatever the clock values of the other transitions. In particular, if the clock of a transition reaches its latest firing time, the firing is not forced, and can be lost to reserve the tokens, for example to fire another transition in conflict.

On the contrary, in a strong semantic, if the clock of a transition reaches its latest firing time, the firing of the transition becomes urgent, which means no more time can elapse before firing the transition. Thus, to determine if an enabled transition can be fired at a date D, it is not enough to look at the clock of this transition, but it is also necessary to look at all the other clocks of enabled transitions, to verify if any of them will exceed the latest firing time of its transition.

The expressivity of this two semantics are not the same. To modelize a real time application, its necessary to modelize urgency of events. Thus, weak semantics appear to be inadapted to describe real time applications, and strong semantics will be preferred. A relevant example of the difference of expressivity between the two semantics is the modelization of a watchdog. A watchdog is a common pattern in real time systems used to modelize alarms. A TPN version of a watchdog is describe on the figure **??**.

- The watchdog is armed when a token comes in place P_1; it triggers the clock of newly enabled transition t_2; the alarm time is here fixed to 10 unit of time.
- If a token comes in P_2 before this deadline, the transition t_1 is instantaneously fired and the watchdog is realeased; the system follows its nominal execution, with a token in place P_3.
- Otherwise, after 10 units of time, the alarm is triggered by the firing of transition t_2; then, the system is in an alarm state with the token in place P_4.

This expected behavior of the TPN model is only possible with a strong semantic. In a weak semantic, if a token comes in P_2 before the latest firing time of t_2, the firing of t_1 can be lost, and the system can still evolve in alarm mode with the firing of t_2; in the same way, if the alarm time is reached, the urgent firing of t_2 can be lost to wait for the coming of a token in P_2, and the system can still go on nominal mode. Only strong semantics can modelize the urgency of transitions t_1 and t_2.

Scenario of firings in a TPN Classical methods in TPN verification are states based methods. A state in a TPN is the conjunction of a marking and the clock values of the enabled transitions. These methods (class graph [1] or zone graph [3]) computes the states space of the net and gather the states into classes or zones. Thus, the base element in these methods (which is either the class or the zone) represents a set of states of the net. This paper exposes a different approach, rather based on events.

Definition 2 (Event) *An event e in a TPN $N = (P,\ T, {}^\bullet(.),\ (.)^\bullet,\ M_0,\ \alpha,\ \beta)$, is the firing of a transition $t \in T$ at a symbolic firing date D_e.*

The object in study in this approach will not be the state space of the net but the scenarii of the net.

Definition 3 (Scenario) *A* scenario *E in a TPN N is a unordered set of events e. Two events can fire the same transition. Thus, a scenario can also be seen has a multiset of transitions.*

A scenario is *valid* if a firing sequence can be constructed with all the transitions appearing in the scenario. Then, as a scenario is unordered like firing sequences are, a scenario represents all the firing sequences which use the transitions of the scenario and which only differ by the firing order between these transitions. Scenarii correspond to the notion of process in the theory of unfoldings of petri net [2].

2.2 From Time Petri Nets to a subset of Linear Logic

Linear Logic has been introduced in 1987 by J-Y Girard [5] as a non monotonic logic, in which the value of the propositions may change during the time. This property allows to represent dynamic systems and resources manipulation. In particular, several works [6, 10, 7] have exhibited links between Linear Logic and Petri nets.

In these works, Petri nets are translated into a fragment of Linear Logic containing only two connectors (on the nine that compose the logic). The first one is the Times connector, represented by the symbol $\otimes$. It corresponds to the classical connector $\wedge$, however it is not an idempotent connector, which means $A \otimes A \neq A$. The second connector is the linear implication $\multimap$. It expresses the consumption of a resource and its transformation into another resource.

Translation of the net structure The initial marking of the Petri net is translated into a Linear Logic formula using only $\otimes$ connectors. To each marked place of the network corresponds a resource. The initial marking of the net is then a conjunction of resources, separated by $\otimes$ connectors.

For example, a marking:

$$M_0 = \{P_1^3, P_2, P_3\}$$

is translated into:

$$P_1 \otimes P_1 \otimes P_1 \otimes P_2 \otimes P_3 \equiv P_1^{\otimes 3} \otimes P_2 \otimes P_3$$

As for transitions, they are translated into Linear Logic formulae that use the $\multimap$ connector. Thus, each transition t is translated into the formula:

$$^\bullet T \multimap T^\bullet$$

where $^\bullet T$ and $T^\bullet$ are the Linear Logic formulae of the previous marking and next marking.

In this paper, the following notations will be adopted: $t_0, t_1, \ldots, t_n$ are transitions and their corresponding linear logic formulae are: $T_0, T_1, \ldots, T_n$; the places $P_0, P_1, \ldots, P_m$ will be represented by the same name in linear logic.

Firing a transition The firing of a transition is carried out by a sequent. A sequent can be defined by:

$$P_0, P_1, \ldots, P_n \vdash C$$

where P_i and C are linear logic formulae. At the left of the turnstile the formula is a conjunction of the premises P_i; at the right C is the conclusion. Thus, let's consider the transition t defined by $^\bullet t = \{P_i, Pj\}$ and $t^\bullet = \{P_k\}$. The sequent:

$$P_i \otimes P_j, P_i \otimes P_j \multimap P_k \vdash P_k$$

expresses the firing of t.

By extension, let consider a sequence of transitions $s = t_0, t_1, \ldots, t_n$, whose linear logic formulae are $T_0, T_1, \ldots, T_n$, an initial marking $M = \{P_i, P_{i+1}, \ldots, P_j\}$, and a final marking $M' = \{P_k, P_{k+1}, \ldots, P_l\}$, such as $M \xrightarrow{s} M'$. The sequent:

$$P_i \otimes P_{i+1} \otimes \ldots \otimes P_j, T_0, T_1, \ldots, T_n \vdash P_k \otimes P_{k+1} \otimes \ldots \otimes P_l$$

represents the firing of the sequence s.

It must be noticed that transitions specified in a sequent are not ordered. In fact the coma is commutative. Thus, a sequent can represent several transitions sequences that only differ by their firing order. Consequently, a linear sequent is the direct representation of a scenario.

2.3 Sequent calculus

Linear sequents would be useless without sequent calculus. This process allows to prove a sequent by applying rules that simplify the sequent in one or two other sequents. For example, to prove the sequent:

$$\Gamma , \Delta , F \multimap G \vdash H$$

where F, G and H are linear logic formulae, whereas Γ and Δ are series of linear logic formulae (that constitute the context), the rule $\multimap_L$ is applied:

$$\frac{\Gamma \vdash F \quad \Delta , G \vdash H}{\Gamma , \Delta , F \multimap G \vdash H} \multimap_L$$

In this rule, the sequent $\Gamma , \Delta , F \multimap G \vdash H$ is replaced by the simplified sequents on top; the proof process is then inferred on those sequents. This rule is called left imply rule and corresponds to the firing of a transition. Three other rules will be used in the method:

- the left times rule $\otimes_L$:

$$\frac{\Gamma , F , G \vdash H}{\Gamma , F \otimes G \vdash H} \otimes_L$$

- the right times rule $\otimes_R$:

$$\frac{\Gamma \vdash F \quad \Delta \vdash G}{\Gamma , \Delta \vdash F \otimes G} \otimes_L$$

- the identity rule id, an axiom rule which allows to terminate proofs:

$$\frac{}{F \vdash F} id$$

An important theorem proved in [6] is:

Theorem 1 *Let $s = t_1, \ldots, t_n$ be a transitions sequence in the underlying Petri net, $T_1, ..., T_n$ the linear logic formulae of these transitions, m and m' two markings, and M, M' the linear logic formulae associated to these markings:*

$$m \xrightarrow{s} m' \Longleftrightarrow M , T_{t_1}, ..., T_{t_n} \vdash M'$$

This means that proving a sequent that corresponds to a scenario between two markings m and m' in a Petri net, is equivalent to proving the accessibility between these two markings.

2.4 Firing dates and temporal labels

Let define how a firing date is associated to an application of the left imply rule. A production date is associated to each resource (either a token or a marking); this production date is the firing date of the fired transition that produced the resource. When applying the left imply rule, the firing date of the new fired transition is computed and becomes the temporal label of the produced resources.

Unlike the state space calculus, this approach is based on a rewriting process allowing to use symbolic temporal variables or parameters. So, transition firing dates are represented by symbolic variables D_i^k, for the k^{th} firing of the transition t_i. The enabling interval bounds are represented by parameters $d_{i_{min}}$ or $d_{i_{max}}$. Then, temporal labels are symbolic expressions, that use these variables and the operator '+' and 'max'.

2.5 Proof algorithm

The following algorithm in weak semantic [10] simulates the run of a scenario in the underlying Petri net and computes symbolic expressions that are the firing dates of the transitions. This algorithm allows to prove a sequent corresponding to a Petri net, and thus to valid a scenario in *WFS*, but moreover, considering a Time Petri Net, it allows to compute the firing dates of the transitions.

The firing date of a transition is the maximum between the production dates of the consumed resources, increased by the enabling duration of the transition.

However, this algorithm does not deal with conflicting situations, and thus it checks if the scenario under study is an event graph [1].

It is important to notice that this algorithm captures the true parallelism through the *max* operator instead of exploring all the interleavings of parallel transitions. However, this algorithm works in weak semantic and consequently cannot deal with situations like timeout or watchdog.

[1]i.e. two transitions have no common input or output places

```
Check if the studied scenario is an event graph
Apply left ⊗ rule: temporal labels are initialized at null
while left ⊸ rule can be applied do
    - Apply left ⊸ rule to the first candidate transition according to lexical order: the firing date is the maximum between the production dates of the consumed tokens, increased by the enabling duration of the transition. The resources produced receive this date as their production date.
    - Terminate the proof of the left sequent by applying the rules right ⊗ or identity
    - Apply, if necessary, the left ⊗ rule to the right sequent: the production date of the disconnected tokens is the one of the marking.
end
```

Algorithm 1: Proof algorithm with temporal labels

This algorithm produces the set of inequations that firing dates must satisfy. This set will be called *firing domain* of the scenario. The firing domain of a scenario is the set of the firing domains of all the events in the scenario. The firing domain of an event is a set of inequations giving to its firing date, upper and lower bounds. These inequations can be manipulated to exhibit significant temporal properties upon some events because they are driving to particular states under analyze.

3. A STRONG SEMANTIC ALGORITHM

With the objective to develop verification methods for real time systems, it has been shown that only a SFS can be used in the modelization of the system into a TPN. Thus, verification methods must be adapted to SFS. The adaption of the previous algorithm to strong semantic is exposed in this section, and constitutes the contribution of this paper. This method is available in T-TPN (a time interval is associated to each transition), with a multi-server hypothesis (a clock is associated to each subset of tokens enabling a transition).

This adaptation modifies several elements. First, it becomes necessary to establish the possible conflicts between transitions. Indeed, conflict situations are the only ones in which SFS and WFS differ. In these cases firing dates must be modified to take into account the urgency of some transitions.

During this section, theorical developments will be illustrated with the figure **??**. On this example, with the previous algorithm which is under WFS hypothesis, the firing interval of t_2 is $[2, 7]$, but will be $[2, 5]$ in a SFS hypothesis.

3.1 Taking transitions conflicts into account

To provide a strong semantic algorithm, it is necessary to identify conflicts between transitions, when two or more transitions share one or more input places.

Moreover, the developed method take into account a more wide notion of conflict. It allows to consider transitions which are in the case of *indirect conflict*. t_i and t_j are in indirect conflict if and only if, t_i and t_j are not in direct conflict, and $\exists t_k$ so that t_i and t_k are in direct (or indirect) conflict and t_k and t_j are in direct (or indirect) conflict.

Indeed, transitions in indirect conflict can influence each other in the calculus of latest firing date. As shown in the figure **??**, where t_1 and t_3 are in indirect conflict, whether t_3 is fired before t_1 or not, t_1 can be fired either between $[1, 6]$ or between $[1, 4]$.

As a consequence, to identity conflicts, the notion of *conflicts group* is defined by transitivity: every pair of transitions in conflict are in the same conflicts group. In the example above, there is only one conflicts group that contains the three transitions. Then, to compute the firing dates in a conflicts group, the firing date of a transition is lower than the minimum of the dates $D_{i_{max}}$ of all enabled transition in the conflicts group of the transition.

3.2 Modification of the firing order

Interleavings must be reintroduced to fire transitions in the same conflicts group. For example, in the TPN of the figure **??**, the two firing order: $t_1; t_3$ and $t_3; t_1$ must be distinguished, because according to the first fired transition, the firing dates will be different. Between transitions in conflict belonging to different groups or transitions not in conflict, interleaving is not necessary.

In addition, during the proof, to be sure to take into account all the possible influences, it is necessary to delay to the maximum the firings of transitions in structural conflict. This allows to activate the maximum number of conflicts. That is why, the transitions that are not in conflict are always fired in priority.

3.3 New firing dates calculus

The last change to the algorithm concerns the calculus of the firing dates. To take into account urgency of transitions, the maximal firing date is modified. To achieve this, the following notation is adopted: D_i is the firing date of a transition. Then, for two transitions t_1 and t_2 in conflict, with respectively scheduled dates between $[D1_{min}, D1_{max}]$ and $[D2_{min}, D2_{max}]$ (the ones computed by the previous algorithm), effective firing dates couldn't exceed the minimum of latest dates: $min(D1_{max}, D2_{max})$. Consequently, in the symbolic expressions computed the operator 'min' is introduced.

Moreover, due to the reintroduction of interleavings between transitions in conflicts, the minimal firing date is also modified. If t_1 and t_2 are in the same conflicts group, and if the sequence $t_1; t_2$ is fired, the total order impose to t_2 that: $D_2 \geq D1_{min}$, and so minimal firing date of t_2 becomes: $max(D1_{min}, D2_{min})$, since t_2 cannot be fired before t_1 for such a sequence. More generally, for a given order of firings between transitions belonging to the same conflict group, minimum firing date of the ith rank transition is the maximum of the minimum dates for the consumed transitions of rank lower or equal to i.

3.4 The algorithm in strong semantic

According to the listed modifications needed to adapt the previous weak semantic algorithm to strong semantic, the following algorithm 2 has been developed.

In this algorithm, the term "classical dates calculus" refers to dates calculus in weak semantics whereas "new dates calculus" refers to aforementioned principle of dates calculus.

In comparison to the WFS algorithm, the proof realized by this new algorithm can be different, because the order of firing is modified. Anyway, the provability of the sequent remains unchanged. What is changed, are the firing dates of the transitions. However, a scenario with a given order between conflicting transitions, may not correspond to a possible firing sequence. To valid a scenario, the proof of the corresponding sequent is not sufficient, but the following condition must also be checked: for each conflicting transition t_i, its firing dates must verify $D_{i_{min}} < D_{i_{max}}$.

```
Create conflicts groups
Apply the rule $\otimes_L$
while rule $\multimap_L$ can be applied do
    - Apply the rule $\multimap_L$ in priority to candidate
    transitions that are not in conflict
    if Transition is not in conflict then
        Apply the classical dates calculus
    else
        Apply the new dates calculus
    end
    - Finalize the proof of the left sequent
    constructed, by applying the rules $\otimes_R$ or id
    - Apply, if necessary, the rule $\otimes_L$ to the right
    sequent
end
```

Algorithm 2: Proof algorithm in strong semantics

3.5 Explanation of the algorithm through the example

Building conflicts groups

- There is only one conflicts group: $G = (t_1, t_2)$

In this conflicts group, one transition in conflict is chosen to compose the scenario: t_2. Thus, the scenario under analyze is composed by the firings of transitions t_2 and t_3. The sequent corresponding to this scenario is:

$$P_0 \otimes P_2 \otimes P_3, T_2, T_3 \vdash P_1 \otimes P_4$$

Applying the rules Then, the algorithm try to prove the sequent by applying rules of linear sequent calculus.

Deconnexion of resources The first step is to apply the rule $\otimes_L$ to disconnect the resources of the initial marking:

$$\frac{\overbrace{P_0, P_2, P_3, T_2, T_3 \vdash P_1 \otimes P_4}^{A}}{P_0 \otimes P_2 \otimes P_3, T_2, T_3 \vdash P_1 \otimes P_4} \otimes_L$$

The production dates of all resources are initialized to 0.

Firing a transition The $\multimap_L$ rule is applied to firable transitions which are not in conflict:

- the rule is applied to t_3

$$\frac{\dfrac{}{P_0 \vdash P_0}\, id \quad \overbrace{P_1, P_2, P_3, T_2 \vdash P_1 \otimes P_4}^{B}}{\underbrace{P_0, P_2, P_3, T_2, T_3 \vdash P_1 \otimes P_4}_{A}} \multimap_L$$

Date calculus The fired transition t_3 is not in a conflicts group. Thus, its firing date (which is the production date of P_1) belongs to the interval defined by the following values :

- Minimum values:

$$D_{3min} = dprod(P_0) + d_{3min} = d_{3min}$$

- Maximum values:

$$D_{3max} = dprod(P_0) + d_{3max} = d_{3max}$$

d_{3min} and d_{3max} are the bounds of the enabling interval of t_3.

Firing a transition

- The $\multimap_L$ rule is applied to t_2, the only transition left in the sequent.

$$\frac{P_2, P_3 \vdash P_2 \otimes P_3 \quad P_1, P_4 \vdash P_1 \otimes P_4}{\underbrace{P_1, P_2, P_3, T_2 \vdash P_1 \otimes P_4}_{B}} \multimap_L$$

Date calculus The fired transition t_2 is in the conflicts group G, because t_2 is in conflict with t_1. To compute the bounds of its firing date, the transition t_1 must also be taken into account.

- In the conflicts group G, no transition has already been fired. Thus, the lower bound of D_2 is not influenced:

$$D_{2min} = d_{2min}$$

- However, the upper bound is the minimum between classical values (without conflit) of t_1 and t_2.
  - Classical date at latest of t_2 : d_{2max}.
  - Classical date at latest of t_1: $D_3 + d_{1max}$.

  Then, latest firing date of t_2 becomes:

$$D_{2max} = min(D_3 + d_{1max}, d_{2max})$$

End of the proof The proof tree is achieved by trivial applications of the rules $\otimes_R$ and *id*.

Conclusion of this example At the end of the algorithm, the following informations are provided:

- The correctness of the proof gives the accessibility of the final marking from the initial marking in the underlying Petri net.
- For each event (transition firing), its firing domain is computed, which means the expressions of its soonest and latest firing date are computed (symbolic expressions).

The scenario in study is valid if:

1. The proof is correct.
2. The firing domain of all the events in the scenario is so that, the lower bound D_{min}, and the upper bound D_{max} verify $D_{min} \leq D_{max}$. In practice, only the event whose transition was in conflict must be checked, because in all other cases this verification is trivial.

For the scenario t_2, t_3 of this example, the two bounds of the event firing t_2, which was in conflict, are $D_{2min} = d_{2min}$ and $D_{2max} = min(D_3 + d_{1max}, d_{2max})$. As $d_{2min} \leq d_{2max}$, only the constraint $d_{2min} \leq D_3 + d_{1max}$ must be checked. And so the parameters must verify:

$$d_{2min} \leq d_{3max} + d_{1max}$$

The numerical applications on this example give the following results:

- Firing domain of t_2: $D_2 \in [2, 5]$.
- Firing domain of t_3: $D_3 \in [1, 3]$.
- Validity of the scenario:

$$d_{2min} = 2 \leq d_{3max} + d_{1max} = 3 + 2$$

These results correspond to a strong semantic hypothesis. The firing domain of t_2 : $[2, 5]$ is scaled down in WFS compared to $[2, 7]$ in SFS.

CONCLUSION

This latter method allows to compute the firing domain of a scenario in a Time Petri Net. This firing domain provides for each event of the scenario inequations that represent either the lowest bound of the firing interval of the event or its latest bound. These inequations use symbolic expressions that are constructed with the symbolic variables D_i of the firing dates of the event, with parameters $D_{j_{min}}$ and $D_{j_{max}}$ that represent the enabling intervals of the transitions, and with the operator $+$, **min** and **max**. These firing domains are expressed under a strong semantics hypothesis. To achieve this a former algorithm in weak semantic has been adapted to the strong semantics.

In comparison with classical model-checking methods, one of the benefit of this approach is that it allows to consider parameterized Time Petri Net. This can be useful in the design of real time applications to provide conception guides, rather than just verifying the model of the system.

A lack of the method is that it does not take into account conflicts of tokens, in case of multi-enabling of transitions, where tokens production dates are not completely ordered. It is shown that expression of scenario duration can depend on the order of consumption of the tokens in the proof. A perspective to deal with this limitation can be to stamp tokens.

Finally, these algorithms have been successfully implemented in a tool LLbox [2].

[2] Available at http://llbox.rts-software.org/

References

[1] Bernard Berthomieu and Michel Diaz. "Modeling and verification of time dependent systems using time petri nets". **IEEE Trans. Softw. Eng.**, 17(3):259–273, 1991.

[2] Thomas Chatain and Claude Jard. "Complete finite prefixes of symbolic unfoldings of safe time Petri nets". In **ICATPN**, volume 4024 of *LNCS*, pages 125–145, june 2006.

[3] Guillaume Gardey, Olivier (H.) Roux, and Olivier (F.) Roux. "A zone-based method for computing the state space of a time Petri net". In **In Formal Modeling and Analysis of Timed Systems, (FORMATS'03)**, volume 2791 of *Lecture Notes in Computer Science*, pages 246–259, Marseille, France, September 2003. Springer-Verlag. Copyright Springer-Verlag.

[4] Guillaume Gardey, Olivier (H.) Roux, and Olivier (F.) Roux. "State space computation and analysis of time Petri nets". **Theory and Practice of Logic Programming (TPLP). Special Issue on Specification Analysis and Verification of Reactive Systems**, 6(3):301–320, 2006.

[5] Jean Yves Girard. "Linear logic". **Theorical Computer Science**, 50, 1987.

[6] Francois Girault. **Formalisation en logique linéaire du fonctionnement des réseaux de Petri**. PhD thesis, Université Paul Sabatier, Toulouse, France, 1997.

[7] Luis Allan Künzle. **Raisonnement Temporel Basé sur les Réseaux de Petri pour des Systèmes Manipulant des Ressources**. PhD thesis, Université Paul Sabatier, Toulouse, France, 1997.

[8] Didier Lime and Olivier (H.) Roux. "Model checking of time Petri nets using the state class timed automaton". **Journal of Discrete Events Dynamic Systems - Theory and Applications (DEDS)**, 16(2):179–205, 2006.

[9] P. M. Merlin. **A Study of the Recoverability of Computing Systems**. PhD thesis, Department of Information and Computer Science, University of California, 1974.

[10] Brigitte Pradin-Chézalviel, Robert Valette, and Luis Allan Kunzle. "Scenario durations characterization of t-timed petri nets using linear logic". In **PNPM'99, 8th International Workshop on Petri Nets and Performance Models**, pages 208–217, Zaragoza, Spain, 1999.