Using Zone Graph Method for Computing the State Space of a Time Petri Net

Guillaume Gardey, Olivier H. Roux, and Olivier F. Roux

IRCCyN (Institut de Recherche en Communication et Cybernétique de Nantes),
1, rue de la Noë B.P. 92101
44321 NANTES cedex 3 (France)
{guillaume.gardey,olivier-h.roux,olivier.roux}@irccyn.ec-nantes.fr,
http://www.irccyn.ec-nantes.fr

Abstract. Presently, the method to verify quantitative time properties on Time Petri Nets is the use of observers. The state space is then computed to test the reachability of a given marking. The main method to compute the state space of a Time Petri Net has been introduced by Berthomieu and Diaz [BD91]. It is known as the "state class method". We present in this paper a new efficient method to compute the state space of a bounded Time Petri Net as a marking graph, based on the region graph method used for Timed Automaton [AD94]. The algorithm is proved to be exact with respect to the reachability of a marking and it computes a graph which nodes are exactly the reachable markings of the Time Petri Net. The tool implemented computes faster than Tina, a tool for constructing the state space using classes, and allows to test on-the-fly the reachability of a given marking.

Keywords: Time Petri Nets, Zone, State Space, Reachability Analysis, Verification.

1 Introduction

Frameworks

The theory of Petri Nets provides a general framework to specify the behavior of reactive systems and time extensions have been introduced to take also temporal specifications into account. The two main time extensions of Petri Nets are Time Petri Nets (TPN) [Mer74] and Timed Petri Nets [Ram74]. While a transition can be fired within a given interval for TPN, in Timed Petri Nets, transitions are fired as soon as possible. There are also numerous way of representing time. It could be relative to places, transitions, arcs or tokens. TPN are mainly divided in P-TPN, A-TPN and T-TPN where a time interval is relative to places (P-TPN), arcs (A-TPN) or transitions (T-TPN). Finally, Time Stream Petri Nets [DS94] were introduced to model multimedia applications.

Concerning the timing analysis of these three models in order to verify properties, few studies have been realized.

Recent works [AN01,dFRA00] consider Timed Arc Petri Nets where each token has a clock representing its "age". Using a backward exploration algorithm [AJ01,FS98], they proved that the coverability and boundedness are decidable for this class of Petri Nets. However, they assume a lazy (non-urgent) behavior of the net: the firing of a transition may be delayed even if its time becomes greater than its latest firing time, disabling the transition.

In [Rok93,RM94], ROKICKI considers an extension of labeled Petri Nets called Orbitals Nets: each transition of the TPN (safe P-TPN) is labeled with a set of events (actions). The state-space is constructed using a forward algorithm very similar to ALUR and DILL region based method. ROKICKI finally uses partial order method to reduce time and space requirements for verification purpose. The semantics used is not formally defined and seems to differ from another commonly adopted proposed by KHANSA [KDC96].

Others approaches aim at translating a TPN into a Timed Automaton (TA) in order to use efficient existent tools on TA. In [CEP00], CORTÈS *et al.* propose to transform an extension of T-TPN into the composition of several TA. Each transition is translated into an automaton (not necessarily identical due to conflict problems) and it is claimed that the composition capture the behaviour of the TPN. In [CR03], CASSEZ and ROUX propose another structural approach: each transition is translated into a TA using the same pattern. The authors prove the two models are timed-bisimilar. In [SA01], SAVA and ALLA compute the graph of reachable markings of a T-TPN. The result is a TA. Nevertheless, they assume the TPN is bounded and does not include ∞ as latest firing time, no proof is given of the timed-bisimilarity between the two models. In [LR03], LIME and ROUX propose a method for building the state class graph of a bounded T-TPN as a TA. The resulting TA is timed-bisimilar and has much lower clocks than previous methods which is of importance for TA model-checking.

Such translations show that TCTL and CTL are decidable for T-TPN and that developed algorithms on TA may be extended to T-TPN.

In this paper, we consider T-TPN in which a transition can be fired within a time interval. For this model, boundedness is undecidable and works report undecidability results, or decidability under the assumption of boundedness of the TPN (as for reachability, decidability [Pop91]). Boundedness and other results are obtained by computing the state-space.

Related Work

State Space Computation of a T-TPN. The main method to compute the state-space of a TPN is the state class graph [Men82,BD91]. A state class C of a TPN is a pair (M, D) where M is a marking and D a set of inequalities called the firing domain. A variable x_i of the firing domain represents the firing time of the enabled transition t_i relatively to the time when the class C was entered in and truncated to nonnegative times. The state class graph preserves markings [BV03] as well as traces and complete traces but can only be used to check untimed reachability properties and is not accurate enough for checking *quantitative* real-time properties. An alternative approach has been proposed by YONEDA

et al. [YR98] in the form of an extension of equivalence classes (atomic classes) which allow CTL model-checking. LILIUS [Lil99] refined this approach so that it becomes possible to apply partial order reduction techniques that have been developed for untimed systems. BERTHOMIEU and VERNADAT [BV03] propose an alternative construction of the graph of atomic classes of YONEDA applicable to a larger class of nets. In [OY97], OKAWA and YONEDA propose another method to perform CTL model-checking on T-TPN, they use a region based algorithm on safe TPN without ∞ as latest firing time. Their algorithm is based on the one of [AD94] and aim at computing a graph conserving branching properties. Nevertheless, the algorithm used to construct the graph seems inefficient (their algorithm do code regions) and no result can be exploited to compare with others methods.

Zone Based Algorithm. Another model used to represent timed systems are Timed Automaton (TA) introduced by ALUR and DILL [AD94]. They introduce the construction of the state space based on regions, *i.e.* a representation of clocks values using equivalence classes. The state space is built by analyzing successors of the initial region (forward analysis). Actually efficient forward (and backward) algorithms using regions do not code regions but zones, a finite convex union of regions because regions suffer of a combinatorial explosions and are quite uneasy to manipulate. This method (forward analysis + zone) is implemented in tools like UPPAAL [LPY97] or KRONOS [Yov97] and is efficiently used to model-check CTL or TCTL properties on timed systems.

Nevertheless, recent works proved the limitations of the data structure used to represent zones: Difference Bounded Matrices (DBM). BOUYER [Bou02,Bou03] proved that the use of DBM made in the forward algorithm leads to an over-approximation of the state-space: some states are said to be reachable while, indeed, they are not.

Contributions

Our aim is to compute efficiently the state space of a bounded T-TPN in order to verify quantitative timing properties.

The paper is devoted to present a different approach to compute the state space of an unsafe bounded T-TPN based on the TA region graph method. Although regions encoding is based on the use of zones implemented with DBM, the algorithm is proved to be exact with respect to the reachability of a marking.

In section 2, we first recall the semantics of T-TPN and present the state class method and its limitations for model-checking. We propose in section 3 a forward algorithm to compute the state space of a bounded T-TPN and prove it is exact with respect to the set of reachable markings. We then present in section 4 some details on the implemented tool and we give an example of the use of observers to check properties on T-TPN. We also compare our tool with a tool using the state class method (TINA). Our experimental tests give encouraging results.

2 Definitions

2.1 Time Petri Nets

Definition 1 (T-TPN). *A Time Petri Net is a tuple* $(P, T, {}^\bullet(.), (.)^\bullet, \alpha, \beta, M_0)$ *defined by:*

- $P = \{p_1, p_2, \ldots, p_m\}$ *is a non-empty set of* places,
- $T = \{t_1, t_2, \ldots, t_n\}$ *is a non-empty set of* transitions,
- ${}^\bullet(.) : T \to \mathbb{N}^P$ *is* the backward incidence function,
- $(.)^\bullet : T \to \mathbb{N}^P$ *is* the forward incidence function,
- $M_0 \in \mathbb{N}^P$ *is* the initial marking *of the Petri Net,*
- $\alpha : T \to \mathbb{Q}^+$ *is the function giving* the earliest firing time *for a transition,*
- $\beta : T \to \mathbb{Q}^+ \cup \{\infty\}$ *is the function giving* the latest firing time *for a transition.*

A Petri Net marking M is an element of $\mathbb{N}^P$ such that for all $p \in P$, $M(p)$ is the number of tokens in the place p.

A marking M enables a transition t if the number of tokens in the corresponding places is greater or equal to the valuation of incoming arcs: $M \geq {}^\bullet t_i$. The set of transitions enabled by a marking M is $enabled(M)$.

A transition t_k is said to be *newly* enabled by the firing of a transition t_i if $M - {}^\bullet t_i + t_i^\bullet$ enables t_k and $M - {}^\bullet t_i$ does not enabled t_k. If t_i remains enabled after its firing then t_i is newly enabled. The set of transitions newly enabled by a transition t_i for a marking M is noted $\uparrow enabled(M, t_i)$.

$v \in (\mathbb{R}_{\geq 0})^T$ is a vector of clocks valuations. v_i is the time elapsing since the transition t_i has been newly enabled.

The semantics of T-TPN is defined as a Timed Transition Systems (TTS). Firing a transition is a discrete transition of the TTS, waiting in a marking, the continuous transition.

Definition 2 (Semantics of a T-TPN). *The semantics of a T-TPN is defined by the Timed Transition System* $\mathcal{S} = (Q, q_0, \to)$*:*

- $Q = \mathbb{N}^P \times (\mathbb{R}_{\geq 0})^T$
- $q_0 = (M_0, \bar{0})$
- $\to \in Q \times (T \cup \mathbb{R}) \times Q$ *is the transition relation including a discrete transition and a continuous transition.*
  - *The continuous transition is defined* $\forall d \in \mathbb{R}_{\geq 0}$ *by:*

$$(M, v) \xrightarrow{e(d)} (M, v') \text{ iff } \begin{cases} v' = v + d \\ \forall k \in [1, n]\ M \geq {}^\bullet t_k \Rightarrow v'_k \leq \beta(t_k) \end{cases}$$

  - *The discrete transition is defined* $\forall t_i \in T$ *by:*

$$(M, v) \xrightarrow{t_i} (M', v') \text{ iff } \begin{cases} M \geq {}^\bullet t_i \\ M' = M - {}^\bullet t_i + t_i^\bullet \\ \alpha(t_i) \leq v_i \leq \beta(t_i) \\ \forall k \in [1, n]\ v'_k = \begin{cases} 0 \text{ if } t_k \in \uparrow enabled(M, t_i) \\ v_k \text{ otherwise} \end{cases} \end{cases}$$

2.2 The State Class Method

The main method to compute the state space of a Time Petri Net is the state class method introduced by BERTHOMIEU and DIAZ in [BD91].

Definition 3 (State class). *A State Class C of a TPN is a pair (M, D) where M is a marking and D a set of inequalities called the firing domain. A variable x_i of the firing domain represents the firing time of the enabled transition t_i relatively to the time when the class C was entered in.*

The state class graph is computed iteratively as follows.

Definition 4 (State Class Method). *Given a class $C = (M, D)$ and a firable transition t_j, the successor class $C' = (M', D')$ by the firing of t_j is obtained by:*

1. *Computing the new marking $M' = M - {}^\bullet t_j + t_j^\bullet$.*
2. *Making variable substitution in the domain: $\forall i \neq j,\ x_i \leftarrow x'_i + x_j$.*
3. *Eliminating x_j from the domain using for instance the Fourier-Motzkin method.*
4. *Computing a canonical form of D' using for instance the Floyd-Warshall algorithm.*

In the state class method, the domain associated to a class is relative to the time when the class was entered and as the transformation (time origin switching) is irreversible, absolute value of clocks cannot be obtained easily. The graph produced is an abstraction of the state space for which temporal information has been lost and generally, the graph has more states than the number of markings of the TPN. Transitions between classes are no longer labeled with a firing constraint but only with the name of the fired transition: the graph is a representation of the untimed language of the TPN.

2.3 Limitations of the State Class Method

As a consequence of the graph construction, sophisticated temporal properties are not easy to check. Indeed, the domain associated to a marking is made of relative values of clocks and the function to compute domains is not bijective. Consequently, domains can not be easily used to verify properties involving constraints on clocks.

In order to get rid of these limitations, several works aim to construct a different state class graph by modifying the equivalence relation between classes. To our knowledge, proposed methods [BV03] depend on the property to check. Checking LTL or CTL properties will lead to construct different state class graphs.

Another limitation of methods and proposed tools to check properties is the need to compute the whole state graph while only the reachability of a given marking is needed (safety properties). The graph is then analyzed by a model checker. Using observers is even more costly: actually, for each property to be

checked, a new state class graph has to be built and the observer can dramatically increase the size of the state space.

In the next section we will present another method to compute the state space of a bounded T-TPN. The resulting graph keeps in memory temporal information and will allow to test on-the-fly temporal properties. The graph is also more compact: it has exactly as many nodes as the number of reachable markings of the TPN.

3 A Forward Algorithm to Compute the State Space of a Bounded T-TPN

The method we propose in this paper is an adaptation, proved to be exact, of the region based method for Timed Automaton [AD94,Rok93].

First, we define a *zone* as a convex union of regions as defined by ALUR and DILL [AD94]. For short, considering n clocks, a zone is a convex subset of $(\mathbb{R}_{\geq 0})^n$. A zone could be represented by a conjunction of constraints on clocks pairs: $x_i - x_j \sim c$ where $\sim \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{N}$.

3.1 Our Algorithm: One Iteration

Given the initial marking and initial values of clocks, timing successors are iteratively computed by letting time pass or by firing transitions.

Let M_0 be a marking and Z_0 a zone. The computation of reachable markings from M_0 according to the zone Z_0 is made as follows:

- Compute the possible evolution of time (future): $\overrightarrow{Z_0}$. This is obtained by setting all upper bounds of clocks to infinity.
- Select only the possible valuations of clocks for which M_0 could exist, *i.e.* valuations of clocks must not be greater than the latest firing time of enabled transitions :

$$Z_0' = \overrightarrow{Z_0} \cap \left\{ \bigwedge_i \{x_i \leq \beta_i \mid t_i \in enabled\,(M_0)\} \right\}$$

  So, Z_0' is the maximal zone starting from Z_0 for which the marking M_0 exists.
- Determine the firable transitions: t_i is firable if $Z_0' \cap \{x_i \geq \alpha_i\}$ is a non empty zone.
- For each firable transition t_i leading to a marking M_{0i}, compute the zone entering the new marking:

  $Z_i = (Z_0' \cap \{x_i \geq \alpha_i\})\,[X_e := 0]\,,$ where X_e is the set of newly enabled clocks.

  This means that each transition which is newly enabled has its clock reset. Then, Z_i is a zone for which the new marking M_{0i} is reachable.

3.2 Convergence Criterion

To ensure termination, a list of zones is associated to each reachable marking. It will keep track of zones for which the marking was already analyzed or will be analyzed. At each step, we compare the zone currently being analyzed to the ones previously computed. If the zone is included in one of the list there is no need to go further because it has already been analyzed or it will lead to compute a subgraph.

3.3 Overapproximation on Zones

An algorithm to enumerate reachable markings for a bounded TPN could be based on the described algorithm but, generally, it will lead to a non-terminating computation. Though the number of reachable markings is finite for a bounded TPN, the number of zones in which a marking is reachable is not necessarily finite (see figure 1).

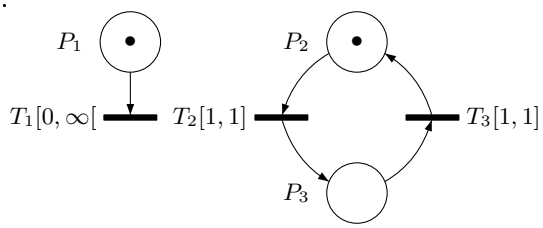

Fig. 1. Time Petri Net with an unbounded number of zones

Let's consider the infinite firing sequence: $(T_2, T_3)^*$. The initial zone is $\{x_1 = 0 \wedge x_2 = 0 \wedge x_3 = 0\}$ (where x_i is the clock associated to T_i), the initial marking $M_0 = (P_1, P_2, P_3) = (1, 1, 0)$. By letting time pass, M_0 is reachable until $x_2 = 1$. When $x_2 = x_1 = 1$ the transition T_2 has to be fired. The zone corresponding to clock values is : $Z_0 = \{0 \leq x_1 \leq 1 \wedge x_1 - x_2 = 0\}$. By firing T_2 and then T_3, the net returns to its initial marking. Entering it, values of clocks are: $x_1 = 2$, $x_2 = 0$ and $x_1 - x_2 = 2$. Indeed, T_1 remains enabled while T_2 and T_3 are fired and x_2 is reset when T_3 is fired because T_2 became newly enabled. Given these new values, the initial marking can exists while $x_2 \leq 1$ *i.e.* for the zone: $Z_1 = \{2 \leq x_1 \leq 3 \wedge x_1 - x_2 = 2\}$. By applying infinitely the sequence (T_2, T_3), there exists an infinite number of zones for which the initial marking is reachable.

Actually, the number of zones is not bounded because infinity is used as latest firing time (T_1). If infinity is not used as latest firing time, all clocks are bounded and so, the number of different zones is bounded [AD94]. The "naive" algorithm is then exact and can be used to compute the state space of a bounded T-TPN.

We will propose a more general algorithm which computes the state space of a T-TPN as defined in section 2, *i.e.* with infinity as latest firing time allowed. It will be based on the use of an operator on zones which construct an equivalence class. The resulting equivalence class will have a finite number of classes.

3.4 Approximation

A common operator on zones is the *k-approx* operator. For a given k value, the use of this operator allows to create a finite set of distinct zones as presented in [AD94]. The algorithm proposed is an extension of the one presented in section 3.1. It consists in applying the *k-approx* operator on the zone resulting from the last step.

This approximation is based on the fact that once the clock associated to an unbounded transition ($[\alpha, \infty[$) has reached the value α, its precise value does not matter. Using *k-approx* (with $k = \alpha$) allows to regroup all zones $[x, \infty[, x \geq \alpha$ in one equivalence class.

Unfortunately recent works on Timed Automaton [Bou02,Bou03] have proved that this operator generally leads to an overapproximation of the reachable localities of TA. Nevertheless, for a given class of TA (diagonal-free), there is no overapproximation of the reachable localities.

Results of BOUYER are directly extensible for T-TPN and we could assert the following theorem:

Theorem 1. *A forward analysis algorithm using k-approx on zone is exact with respect to TPN marking reachability for bounded TPN.*

A detailed presentation of the result of BOUYER and the demonstration of this theorem is presented in appendix A.

As the approximation is only needed for T-TPN with infinity as latest firing time, the following theorem can be asserted:

Corollary 1. *For a bounded T-TPN without infinity as latest firing time, a forward analysis algorithm using zones computes the exact state-space of the T-TPN.* Proof is given in appendix A.

4 The Tool: MERCUTIO

4.1 Presentation

We have implemented the algorithm to compute all the reachable markings of a bounded T-TPN using DBM to encode zones. The tool implemented (MERCUTIO) is integrated into ROMEO [Rom00], a software for TPN edition and analysis.

As boundedness of T-TPN is undecidable, MERCUTIO offers stopping criteria: number of reached markings, computation time, bound on the number of tokens in a place.

4.2 Performances

We have compared our tool with TINA [BV]. TINA is developed by BERTHOMIEU and it is the most efficient tool we know for the state class construction of a TPN. The results are given in table 1. We used the last stable version TINA (2.5.1).

Time Petri Net	TPN (places / trans.)	TINA	MERCUTIO
Example 1 (oex15)	16 / 16	10.5 s	1.4 s
Example 2 (oex7)	22 / 20	30.5 s	2.4 s
Example 3 (oex8)	31 / 21	29 s	2.4 s
Example 4 (P6C7)	21 / 20	31.6 s	8.5 s
Example 5 (P10C10)	32 / 31	4.2 s	1.8 s
Example 6 (landing gear)	107 / 101	18 min 27 s	27.8 s
Example 7 (Gate Controller - 3 trains)	20 / 23	2 s	$<$1 s
Example 8 (Gate Controller - 4 trains)	24 / 29	3 min 8 s	9 s

Table 1. TINA 2.5.1 – MERCUTIO (Pentium II, 400 MHz, 256Mb)

Though the method is not the same, it can give a time reference to compute all reachable markings of a T-TPN.

Examples 1 to 5 come from real-time systems (parallel tasks [1], periodic tasks[2–3], producer-consumer [4–5]). Example 6 is a larger system representing a simplified landing gear, it counts 107 places and 101 transitions. It is a T-TPN representation of a landing gear case study published in [BC02]. Examples 7 and 8 are the classical level crossing example (3 and 4 trains).

For this set of examples and for all nets we have tested, our tool performs better than TINA.

4.3 Reachability Analysis – Observers

TPN observers are a method to model check TPN. It consists in adding to the Petri Net places and transitions to model the property to check. The property is transformed in testing for the reachability of a given marking. Then, as for the construction of the state class graph, it is possible to check properties on TPN with observers.

In [TST97], the authors present generic observers to model properties like absolute or relative time between the firing of transitions, causality or simultaneity.

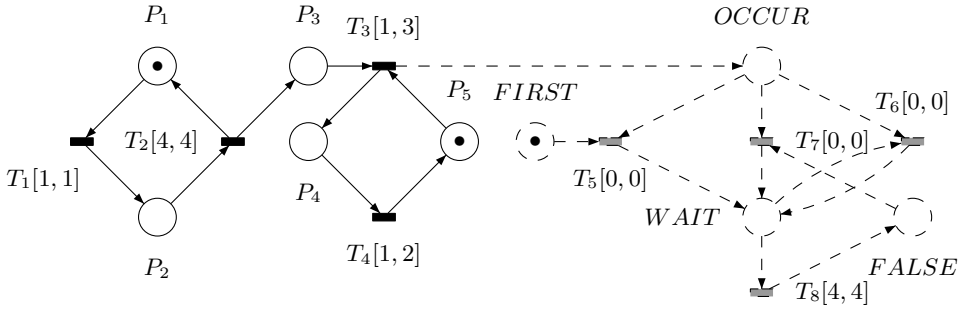

Fig. 2. Example of a TPN and an observer (dash point).

Let's consider the net of figure 2. It represents a simple TPN and an observer in dash point. The observer allows to check the property: "2 successive occurrences of T_3 always append in less than 4 time units". The property is false if a run such that the place $FALSE$ as a token exists.

By constructing the state space and look for a run with a token in $FALSE$ allows to conclude that the property is false.

Generally there is no need to compute the whole state space: the algorithm can be stopped at the first marking verifying a property. In its current release, MERCUTIO can perform an on the fly analysis of the TPN. Providing a set of constraints on reachable markings, MERCUTIO will stop at the first marking verifying constraints provided.

5 Conclusions

In this paper we proposed an efficient method based on the region graph approach to compute the state space of a bounded T-TPN. The implemented algorithm computes the graph of reachable markings by the use of zones coded with DBM, and we proved it is exact with respect to reachability. Tests on several examples show that our implementation is faster than the most efficient tool we know to compute the state class graph (TINA).

Nevertheless, observers are still a not easy way to model-check a Petri Net. For each new property an observer has to be built and then the state space has to be computed. As the exact state space is computed by our method for a bounded TPN without infinity as latest firing time, we are involved in realizing an on-the-fly TCTL model-checker. Despite the overapproximation issue for TPN with infinity as latest firing time, we think possible, by choosing an appropriate parameter for the approximation, to perform model-checking.

It could also be possible to improve MERCUTIO performances (memory needs) by using compacter data structures. Though DBM is an efficient way to represent zones, several works offers data structures based on Binary Decision Diagram (CDD, RED, CRD) to minimize memory needs. We are currently working on such improvements.

References

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[AJ01] Parosh Aziz Abdulla and Bengt Jonsson. Ensuring completeness of symbolic verification methods for infinite-state systems. *Theoretical Computer Science*, 256:145–167, 2001.

[AN01] Parosh Aziz Abdulla and Aletta Nylén. Timed petri nets and bqos. In *22nd International Conference on Application and Theory of Petri Nets (ICATPN'01)*, volume 2075 of *Lecture Notes in Computer Science*, pages 53–72, Newcastle upon Tyne, United Kingdom, june 2001. Springer-Verlag.

[BC02] F. Boniol and F. Carcenac. Une étude de cas pour la vérification formelle de propriétés temporelles. FAC'2002, Mar 2002. http://www.laas.fr/ francois/SVF/FAC02.

[BD91] Bernard Berthomieu and Michel Diaz. Modeling and verification of time dependent systems using time petri nets. *IEEE transactions on software engineering*, 17(3):259–273, 1991.

[BDFP00] Patricia Bouyer, Catherine Dufourd, Emmanuel Fleury, and Antoine Petit. Are timed automata updatable? In *Proc. 12th International Conference on Computer Aided Verification (CAV'2000)*, volume 1855 of *LNCS*, pages 464–479. Springer Verlag, July 2000.

[Bou02] Patricia Bouyer. Timed automata may cause some troubles. Technical report, LSV, July 2002.

[Bou03] Patricia Bouyer. Unteamable timed automata! In *Proc. 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS'2003)*, volume 2607 of *LNCS*, pages 620–631, Berlin, Germany, February 2003. Springer Verlag.

[BV] B. Berthomieu and F. Vernadat. TINA. http://www.laas.fr/tina.

[BV03] Bernard Berthomieu and François Vernadat. State class constructions for branching analysis of time petri nets. In *9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2003)*, pages 442–457. Springer–Verlag, Apr 2003.

[CEP00] Luis Alejandro Cortes, Petru Eles, and Zebo Peng. Verification of embedded systems using a petri net based representation. In *13th International Symposium on System Synthesis (ISSS 2000)*, pages 149–155, Madrid, Spain, September 2000.

[CR03] Franck Cassez and Olivier (H.) Roux. Traduction structurelle des réseaux de petri temporels vers les automates temporisés. In *4ieme Colloque Francophone sur la Modélisation des Systèmes Réactifs, (MSR'03)*, Metz, France, oct 2003. To appear.

[dFRA00] David de Frutos Escrig, Valentín Valero Ruiz, and Olga Marroquín Alonso. Decidability of properties of timed-arc petri nets. In *21st International Conference on Application and Theory of Petri Nets (ICATPN'00)*, volume 1825 of *Lecture Notes in Computer Science*, pages 187–206, Aarhus, Denmark, june 2000. Springer-Verlag.

[DS94] M. Diaz and P. Senac. Time stream petri nets: a model for timed multimedia information. *Lecture Notes in Computer Science*, 815:219–238, 1994.

[FS98] Alain Finkel and Philippe Schnoebelen. Fundamental structures in well-structured infinite transitions systems. In *3rd Latin American Theoretical Informatics Symposium (LATIN'98)*, volume 1380 of *Lecture Notes in Computer Science*, pages 102–118, Campinas, Brazil, april 1998. Springer-Verlag.

[KDC96] Wael Khasa, J.-P. Denat, and S. Collart-Dutilleul. P-Time Petri Nets for manufacturing systems. In *International Workshop on Discrete Event Systems, WODES'96*, pages 94–102, Edinburgh (U.K.), august 1996.

[Lil99] Johan Lilius. Efficient state space search for time petri nets. In *MFCS Workshop on Concurrency '98*, volume 18 of *ENTCS*. Elsevier, 1999.

[LPY97] Kim G. Larsen, Paul Pettersson, and Wang Yi. UPPAAL in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1(1–2):134–152, Oct 1997. http://www.uppaal.com/.

[LR03] Didier Lime and Olivier (H.) Roux. State class timed automaton of a time petri net. In *The 10th International Workshop on Petri Nets and Performance Models, (PNPM'03)*. IEEE Computer Society, Sept. 2003. To appear.

[Men82] Miguel Menasche. *Analyse des réseaux de Petri temporisés et application aux systèmes distribués.* PhD thesis, Université Paul Sabatier, Toulouse, France, 1982.

[Mer74] P. M. Merlin. *A study of the recoverability of computing systems.* PhD thesis, Department of Information and Computer Science, University of California, Irvine, CA, 1974.

[OY97] Yasukichi Okawa and Tomohiro Yoneda. Symbolic ctl model checking of time petri nets. In Scripta Technica, editor, *Electronics and Communications in Japan*, volume 80, pages 11–20, 1997.

[Pop91] Louchka Popova. On time petri nets. *Journal Information Processing and Cybernetics, EIK*, 27(4):227–244, 1991.

[Ram74] C. Ramchandani. *Analysis of asynchronous concurrent systems by timed Petri nets.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1974. Project MAC Report MAC-TR-120.

[RM94] Tomas G. Rokicki and Chris J. Myers. Automatic verification of timed circuits. In *International Conference on Computer-Aided Verification (CAV'94)*, pages 468–480. Springer–Verlag, 1994.

[Rok93] Tomas G. Rokicki. *Representing an Modeling Circuits.* PhD thesis, Stanford University, 1993.

[Rom00] Romeo. http://www.irccyn.ec-nantes.fr/irccyn/d/fr/equipes/tempsreel/logs. *A tool for Time Petri Nets Analysis*, 2000.

[SA01] A. T. Sava and H. Alla. Commande par supervision des systèmes à évènements discrets temporisées. *Modélisation des systèmes réactifs (MSR 2001)*, pages 71–86, 2001.

[TST97] Joël Toussaint, Françoise Simonot-Lion, and Jean-Pierre Thomesse. Time constraint verifications methods based time petri nets. In *6th Workshop on Future Trends in Distributed Computing Systems (FTDCS'97)*, pages 262–267, Tunis, Tunisia, 1997.

[Yov97] Sergio Yovine. Kronos: A verification tool for real-time systems. *International Journal of Software Tools for Technology Transfer*, 1(1–2):123–133, Oct 1997. http://www-verimag.imag.fr/TEMPORISE/kronos/.

[YR98] Tomohiro Yoneda and Hikaru Ryuba. Ctl model checking of time petri nets using geometric regions. *IEICE Transactions on Information and Systems*, E99-D(3):297–396, march 1998.

A Proof of theorem 1 and 2

The proof of the theorem is a consequence of works of BOUYER on Timed Automaton.

A.1 Timed Automaton and Overapproximation.

In [Bou02,Bou03] the author presents an exact algorithm with respect to reachability for Timed Automaton using an operator called $Closure_k$. Then, it is proved that the operator *k-approx*, commonly used on DBM, leads to a zone included in $Closure_k$. The operator $Closure_k$ has the same aim that *k-approx*, it divides the clock space into a finite number of regions so that the computed number of zones is finite.

Let $\mathcal{A}$ be an updatable timed automaton and k the greatest constant appearing in constraints on clocks of $\mathcal{A}$. $\mathcal{R}_k$ is a finite set of regions as defined in [BDFP00] (similar to region's definition of ALUR and DILL in [AD94]).

The operator $Closure_k$ is defined by:

$$Closure_k(Z) = \cup\{R \in \mathcal{R}_k \mid R \cap Z \neq \emptyset\}$$

Let:

$$Post_{i_k}(Z) = up_{i_k}\left(g_{i_k} \cap \overrightarrow{Z}\right)$$
$$Post'_{i_k}(Z) = Post_{i_k}(Closure_k(Z))$$

where g_{i_k} is a conjunction of diagonal-free constraints on clocks and up_{i_k} a function assigning values to some clocks. Diagonal-free constraints are constraints which do not involved comparisons between clocks, constraints are only of the form: $x_i \sim n, n \in \mathbb{N}, \sim \in \{<, \leq, =\geq, >\}$.

Let:

$$M_1 = Post_{i_n} \circ Post_{i_{n-1}} \circ \cdots \circ Post_{i_1}(Z)$$
$$M_2 = Post'_{i_n} \circ Post'_{i_{n-1}} \circ \cdots \circ Post'_{i_1}(Z)$$
$$M_3 = Closure_k(M1)$$

M_1 is the exact zone computed which results from a sequence of transition $i_1, \ldots, i_n$. M_2 is the zone computed by a forward algorithm using $Closure_k$.

Using properties on $Closure_k$, it is proved that $M_1 \subseteq M_2 \subseteq M_3$. Precisely, if M_1 is empty then $Closure_k(M_1)$ is also empty and then, M_2 is also empty. An algorithm using $Closure_k$ as operator is exact with respect to reachability.

Nevertheless, it is not easy to compute $Closure_k$ and DBM is known to be an efficient data structure to perform operations on zones.

It is then proved that for any zone Z,

$$Z \subseteq \textit{k-approx}(Z) \subseteq Closure_k(Z)$$

k-approx is computed by replacing in the DBM all values greater than k by ∞ and all values lower than $-k$ by $-k$.

Consequently, any algorithm using *k-approx* is also exact with respect to reachability for Updatable Timed Automata with diagonal-free constraints. The proof of this theorem is mainly based on the fact that constraints appearing in automaton are diagonal-free.

A.2 Reachability of a marking for a TPN using DBM.

Theorem 1. *A forward analysis algorithm using k-approx on DBM is exact with respect to TPN marking reachability for bounded T-TPN.*

Proof. We will prove that operations on zones needed are a subset of the ones described in [Bou02,Bou03]. Indeed, we choose k as:

$$k = Max\left(max\left(\alpha\left(t_i\right)\right)_{t_i \in T}, max\left(\beta\left(t_i\right)\right)_{t_i \in T}\right)$$

The reset function is a particular case of the update function up_{i_k} and the intersection we made on zones are intersection with diagonal-free constraints. Actually, we intersect each clock with the value of the latest firing time of the associated transition. Thus, operations on zones for TPN verify the hypothesis of the demonstration presented in [Bou02,Bou03]. □

Corollary 1. *For a bounded T-TPN without infinity as latest firing time, a forward analysis algorithm using zones computes the exact state-space of the T-TPN.*

Proof. As infinity is not used as latest firing time, all clocks are bounded and so, the number of different zones is bounded [AD94]. The state space computed is then exact.