

---

# Problèmes d'accessibilité et espaces d'états abstraits des réseaux de Petri temporels à chronomètres

**Bernard Berthomieu<sup>1</sup> — Didier Lime<sup>2,3</sup> — Olivier Henri Roux<sup>3</sup> — François Vernadat<sup>1</sup>**

<sup>1</sup> LAAS-CNRS, 7, Avenue du Colonel Roche, 31077 Toulouse Cedex, France

<sup>2</sup> Aalborg University - CISS, Fredrik Bajers Vej 7B, 9220 Aalborg East, Denmark

<sup>3</sup> IRCCyN, 1, rue de la Noë, BP 92101, 44321 Nantes Cedex 3, France

---

*RÉSUMÉ.* Plusieurs extensions des réseaux de Petri temporels ont été proposées pour modéliser la suspension et la reprise des actions dans les systèmes temps réel. En utilisant une classe simple de réseaux de Petri à chronomètres (SwTPN), nous montrons d'abord que l'accessibilité d'états dans tous ces modèles est indécidable, même pour des réseaux bornés. Nous proposons alors un semi-algorithme de construction d'une représentation exacte du comportement des SwTPN, basé sur la méthode connue des classes d'états. Nous discutons ensuite des méthodes de surapproximation assurant l'arrêt de la construction sur une sous-classe des SwTPN bornés et proposons une surapproximation paramétrable basée sur une quantification des polyèdres représentant les domaines temporels des classes d'états.

*ABSTRACT.* Several extensions of Time Petri nets have been proposed for modeling suspension and resumption of actions in timed systems. Using a simple class of TPN extended with stopwatches (SwTPN), we first prove that state reachability in all these models is undecidable, even when bounded. A semi-algorithm is then proposed for building exact representations of the behavior of SwTPN, based on the known state class method for Time Petri nets. Next, we discuss overapproximation methods ensuring termination of the construction on a subclass of bounded SwTPN, and propose a parameterizable overapproximation based on a quantization of the polyhedra representing temporal information.

*MOTS-CLÉS :* réseaux de Petri temporels, chronomètres, classes d'états, accessibilité, décidabilité, approximation, vérification et modélisation des systèmes temps réel.

*KEYWORDS:* time Petri nets, stopwatches, state classes, reachability, decidability, approximation, real-time systems modeling and verification.

## 1. Introduction

Modéliser certains systèmes temps réel nécessite d'exprimer la suspension et la reprise d'actions. Pour répondre à ces besoins, plusieurs modèles basés sur la notion de chronomètre (stopwatch) ont été proposés. Parmi les extensions des automates temporisés (TA), les automates à chronomètres (SWA) sont définis comme une sous-classe des automates hybrides linéaires (LHA) pour laquelle les dérivées des variables par rapport au temps ne peuvent prendre que deux valeurs exprimant la progression (1) ou la suspension (0). Le problème de l'accessibilité est équivalent pour les SWA et les LHA (Cassez *et al.*, 2000) et il a été démontré indécidable pour les LHA (Alur *et al.*, 1995). Aucune sous-classe décidable des SWA préservant des capacités suffisantes de modélisation n'ayant été identifiée, obtenir une abstraction finie de l'espace d'états se fait par l'utilisation de surapproximations qui caractérisent un ensemble d'états incluant l'espace exact mais pouvant être plus important. Ces surapproximations fournissent des conditions suffisantes pour les propriétés de sûreté.

Les réseaux de Petri Temporels (TPN) (Merlin, 1974) constituent un autre modèle largement répandu pour les systèmes temps réel. Les TPN étendent les réseaux de Petri par des intervalles temporels associés aux transitions. Des abstractions de l'espace d'états des TPN préservant diverses classes de propriétés peuvent être calculées en termes de classes d'états (Berthomieu *et al.*, 1983) (Berthomieu *et al.*, 1991) (Berthomieu *et al.*, 2003). Ces classes représentent des ensembles d'états par un marquage et un polyèdre capturant l'information temporelle. L'accessibilité d'états est indécidable pour les TPN, mais est décidable pour les TPN bornés (suffisants en pratique).

Plusieurs extensions des TPN ont été proposées pour permettre d'exprimer la suspension et de la reprise d'actions : les Scheduling-TPN (Roux *et al.*, 2002) (Lime *et al.*, 2003), les Preemptive-TPN (Bucci *et al.*, 2004) et les TPN à hyperarcs inhibiteurs (IHTPN) (Roux *et al.*, 2004). Les deux premiers ajoutent ressources et priorités au modèle TPN, les IHTPN introduisent des arcs inhibiteurs qui contrôlent la progression des transitions. L'accessibilité d'états dans tous ces modèles est indécidable, mais le problème reste ouvert pour des réseaux bornés (d'un grand intérêt pratique).

Pour toutes ces extensions, des semi-algorithmes calculant une abstraction de l'espace d'états en termes de classes d'états sont disponibles. Comme pour les SWA, aucune sous-classe décidable suffisamment expressive n'a été identifiée. Une méthode de surapproximation a été proposée, consistant à approximer les polyèdres des classes par le plus petit représentable par une matrice de différences (DBM) le contenant. La méthode est efficace, mais les surapproximations obtenues sont souvent grossières.

Dans cet article, nous introduisons un modèle simple de TPN à chronomètres, les SwTPN, que l'on peut voir comme une simplification des IHTPN. Les SwTPN étendent les TPN avec des *arcs activateurs* qui contrôlent la progression des transitions. Nous démontrons ensuite que le problème de l'accessibilité d'états pour ces réseaux est indécidable, même lorsque ceux-ci sont bornés. Il s'ensuit que beaucoup de propriétés intéressantes de ces réseaux sont indécidables, et que ces problèmes sont également indécidables pour toutes les extensions des TPN mentionnées ci-dessus.

Classiquement, la preuve réduit le problème de l'accessibilité d'états des SwTPN bornés à celui de l'arrêt d'une machine à deux compteurs.

Les algorithmes de calcul des graphes de classes d'états pour les TPN s'adaptent facilement aux SwTPN. Ils conduisent à des abstractions exactes de l'espace d'état, mais, par le résultat d'indécidabilité ci-dessus, le caractère borné du SwTPN n'implique pas le caractère fini des graphes de classes. Pour cela, nous proposons une nouvelle méthode de surapproximation, basée sur une quantification des polyèdres représentant l'information temporelle dans les classes. En ajustant un paramètre, le comportement exact du SwTPN peut être approché aussi étroitement que souhaité.

L'article est organisé comme suit : la section 2 présente les réseaux de Petri à chronomètres et le semi-algorithme de calcul du graphe des classes d'états. Un exemple est présenté en section 3. L'indécidabilité de l'accessibilité d'états pour ces réseaux est établie dans la section 4. La section 5 présente la méthode de quantification des polyèdres pour le calcul de surapproximations de l'espace d'états des SwTPN bornés, et discute quelques résultats obtenus à partir d'une implémentation expérimentale.

## 2. Réseaux de Petri à chronomètre

### 2.1. SwTPN, états, graphes d'états

Soit  $\mathbf{I}^+$  l'ensemble non vide des intervalles réels avec bornes rationnelles non négatives. Pour  $i \in \mathbf{I}^+$ ,  $\downarrow i$  représente sa borne inférieure, et  $\uparrow i$  sa borne supérieure (si elle existe) ou  $\infty$ . Pour tout  $\theta \in \mathbf{R}^+$ ,  $i \dot{-} \theta$  représente l'intervalle  $\{x - \theta \mid x \in i \wedge x \geq \theta\}$ .

**Définition 1** Un Réseau de Petri temporel à chronomètres (SwTPN) est un  $n$ -uplet  $\langle P, T, \mathbf{Pre}, \mathbf{Post}, \mathbf{Sw}, m_0, I_s \rangle$ , tel que  $\langle P, T, \mathbf{Pre}, \mathbf{Post}, m_0, I_s \rangle$  est un réseau de Petri temporel et  $\mathbf{Sw} : T \times P \rightarrow \mathbf{N}$  est une fonction appelée fonction d'activation.

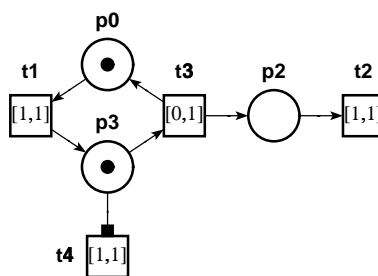


Figure 1. Un SwTPN

Les réseaux de Petri temporels étendent les réseaux de Petri par  $I_s : T \rightarrow \mathbf{I}^+$ , appelée fonction *Intervalle Statique*. La fonction  $\mathbf{Sw}$  associe un entier à chaque  $(t, p) \in T \times P$ . Les valeurs supérieures à 0 sont représentées par des arcs particuliers

appelés *arcs activateurs*, orienté par un "diamant". La figure 1 montre un SwTPN. L'arc de la place  $p_3$  vers la transition  $t_4$  est un arc activateur de poids 1.

Une transition  $t$  est *sensibilisée* par le marquage  $m$  ssi  $m \geq \mathbf{Pre}(t)$ . De plus, une transition sensibilisée par  $m$  est *active* ssi  $m \geq \mathbf{Sw}(t)$ , sinon elle est dite *suspendue*. Les états et leur relation de transition temporisée sont définis comme suit :

**Définition 2** Un état d'un SwTPN est un couple  $s = (m, I)$  tel que  $m$  est un marquage et  $I$  (Intervalle) est une fonction qui associe un intervalle temporel dans  $\mathbf{I}^+$  à chaque transition sensibilisée par  $m$ . Nous noterons  $(m, I) \xrightarrow{t@\theta} (m', I')$  ssi  $\theta \in \mathbf{R}^+$  et :

- 1)  $m \geq \mathbf{Pre}(t) \wedge m \geq \mathbf{Sw}(t) \wedge \theta \geq \downarrow I(t) \wedge$   
 $(\forall k \in T)(m \geq \mathbf{Pre}(k) \wedge m \geq \mathbf{Sw}(k) \Rightarrow \theta \leq \uparrow I(k))$
- 2)  $m' = m - \mathbf{Pre}(t) + \mathbf{Post}(t)$
- 3)  $(\forall k \in T)(m' \geq \mathbf{Pre}(k) \Rightarrow$   
 $I'(k) = \mathbf{si } k \neq t \wedge m - \mathbf{Pre}(t) \geq \mathbf{Pre}(k)$   
 $\mathbf{alors si } m \geq \mathbf{Sw}(k) \mathbf{ alors } I(k) \dot{-} \theta \mathbf{ sinon } I(k)$   
 $\mathbf{sinon } I_s(k)$

Nous avons  $s \xrightarrow{t@\theta} s'$  si le tir de  $t$  à partir de  $s$  à la date (relative)  $\theta$  conduit à  $s'$ . (1) assure que  $t$  tire dans son intervalle temporel à moins qu'elle ne soit désensibilisée par le tir d'une autre transition, et qu'elle est active. (2) est la règle de transformation de marquage classique. (3) signifie que les transitions nouvellement sensibilisées sont associées à leurs intervalles de tir statiques alors que les transitions persistantes (celles restant sensibilisées lors du tir) ont leur intervalle inchangé si elles étaient suspendues, ou décalé de  $\theta$  et tronqué à zéro si elles étaient actives. Les transitions qui restent sensibilisées lors de leur propre tir sont considérées comme nouvellement sensibilisées.

Le *graphe d'états* d'un SwTPN est l'ensemble des états accessibles depuis son état initial  $(m_0, I_0)$ . Une *exécution*, ou *échancier*, est une séquence de transitions successivement tirables, chacune associée à sa date (relative) de tir.

## 2.2. Classes d'états d'un SwTPN

Comme pour les TPN, le nombre d'états d'un SwTPN est potentiellement infini. Les constructions de graphes des classes d'états pour les TPN, fournissant des abstractions de leur espaces d'états, sont aisément adaptables aux SwTPN. Nous adaptons ici la construction de (Berthomieu *et al.*, 1983), qui préserve les propriétés *LT L*.

Une classe d'états est définie par un couple  $(m, D)$ , où  $m$  est un marquage et  $D$  un domaine de tir décrit par un système d'inéquations  $A\phi \leq b$ . La variable  $\phi_i$  représente la date à laquelle la  $i^{me}$  transition sensibilisée par  $m$  peut être tirée. Notons  $(m, D = \{A\phi \leq b\}) \cong (m', D' = \{A'\phi \leq b'\})$  quand  $m = m'$ , et  $D$  et  $D'$  ont mêmes ensembles de solutions. Le graphe des classes d'états d'un SwTPN est construit de la manière suivante :

**Algorithme 1 (Calcul des classes d'états)**

Pour toute séquence de tir  $\sigma$ , un couple  $C_\sigma$  peut être calculé comme indiqué ci-dessous. L'ensemble des classes d'états est le plus petit ensemble  $C$  incluant  $C_\epsilon$  et tel que, lorsque  $C_\sigma \in C$  et  $\sigma.t$  est tirable, alors soit  $C_{\sigma.t} \in C$ , soit  $C_{\sigma.t}$  est équivalent par  $\cong$  à un couple de  $C$ . Il y a un arc étiqueté  $t$  entre les classes  $C_\sigma$  et  $c$  ssi  $c \cong C_{\sigma.t}$ .

– Le couple initial est  $C_\epsilon = (m_0, \{Eft_s(t) \leq \underline{\phi}_t \leq Lft_s(t) \mid \mathbf{Pre}(t) \leq m_0\})$

– Si  $\sigma$  est tirable et  $C_\sigma = (m, D = \{A\underline{\phi} \leq \underline{b}\})$ , alors  $\sigma.t$  est tirable ssi :

(i)  $m \geq \mathbf{Pre}(t) \wedge m \geq \mathbf{Sw}(t)$  ( $t$  est sensibilisée et active par  $m$ )

(ii) L'ensemble des solutions du système

$D \cup \{\underline{\phi}_t \leq \underline{\phi}_i \mid i \neq t \wedge m \geq \mathbf{Pre}(i) \wedge m \geq \mathbf{Sw}(i)\}$  n'est pas vide

– Si  $\sigma.t$  est tirable, alors  $C_{\sigma.t} = (m', D')$  est calculé à partir de  $C_\sigma = (m, D)$  par :

$m' = m - \mathbf{Pre}(t) + \mathbf{Post}(t)$

$D'$  obtenu par :

1) Les contraintes de tir pour  $t$  dans (ii) (ci-dessus) sont ajoutées à  $D$ .

2) Pour toute  $k$  sensibilisée par  $m'$ , une variable  $\underline{\phi}'_k$  est introduite, telle que :

$\underline{\phi}'_k = \underline{\phi}_k - \underline{\phi}_t$  si  $k \neq t$ ,  $m - \mathbf{Pre}(t) \geq \mathbf{Pre}(k)$ , et  $m \geq \mathbf{Sw}(k)$

$\underline{\phi}'_k = \underline{\phi}_k$  si  $k \neq t$ ,  $m - \mathbf{Pre}(t) \geq \mathbf{Pre}(k)$ , et  $\neg(m \geq \mathbf{Sw}(k))$

$\underline{\phi}'_k \in I_s(k)$  sinon

3) Les variables  $\underline{\phi}$  sont éliminées.

Pour les TPN (sans chronomètre), l'ensemble des systèmes  $D$  que l'on peut obtenir par l'algorithme 1 est fini (Berthomieu *et al.*, 1983). Ainsi, les TPN bornés admettent des graphes de classes d'états finis. De plus, les systèmes  $D$  sont des systèmes de différences, pour lesquels des formes canoniques peuvent être calculées efficacement.

Malheureusement, ces propriétés ne sont plus vraies en présence de chronomètres. Considérons comme exemple le réseau de la figure 1. Par de simples arguments temporels, on peut montrer que ce réseau est borné. Pour ce réseau, les séquences  $t_3.t_1.(t_3.t_2.t_1)^n.t_2.t_4$ , pour tout  $n \in \mathbf{N}$ , sont tirables à partir de l'état initial, et conduisent par l'algorithme 1 à un ensemble infini de classes, toutes ayant le même marquage. Le tir de  $\sigma_n$  conduit à la classe suivante :

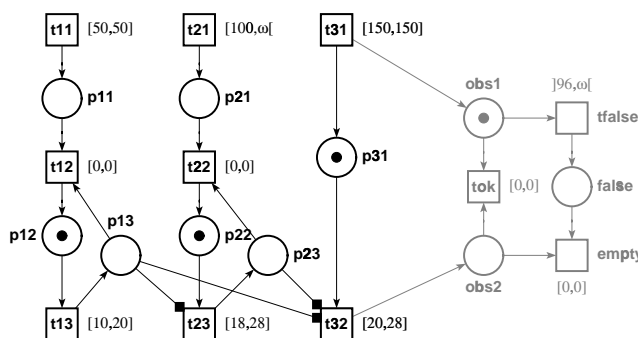
$$\begin{aligned} \text{marquage} &= p_0 p_3 \\ \text{domaine de tir} &= \{\underline{\phi}'_{t_4} = 1, 0 \leq \underline{\phi}_{t_3}, \underline{\phi}_{t_3} \leq \underline{\phi}_{t_1} \leq (n+1)/(n+2)\} \end{aligned}$$

Lorsque l'algorithme 1 termine, il produit une abstraction finie de l'espace d'états qui préserve les marquages et les propriétés *LTL* du réseau. Les graphes de classes d'états "fortes" et "atomiques" introduits dans (Berthomieu *et al.*, 2003) pourraient aussi être facilement adaptés aux SwTPN ; le premier préserve les états et les propriétés *LTL*, et le second préserve les états et les propriétés *CTL*.

### 3. Un exemple simple

Nous décrivons dans cette section quelques expériences faites avec une implémentation de l'algorithme 1 pour les SwTPN intégrée dans une extension de l'outil *Tina* (Berthomieu *et al.*, 2004b). Pour les opérations sur les polyèdres, l'implémentation repose sur la bibliothèque *NewPolka* (Jeannet, 2002).

Nombre de SwTPN modélisant des problèmes classiques admettent par l'algorithme 1 des graphes de classes finis. Comme illustration, considérons l'exemple proposé dans (Bucci *et al.*, 2004). Cet exemple modélise trois tâches indépendantes : deux tâches périodiques (de période 50 et 150 unités de temps) et une tâche sporadique avec un intervalle minimum d'interarrivée de 100. La tâche 1 (de période 50) a une priorité supérieure à celle des 2 autres tâches et la tâche sporadique a une priorité supérieure à celle de la troisième. Traduit en SwTPN, cet exemple est montré figure 2 (en noir).



**Figure 2.** Deux tâches périodiques, une sporadique, et observateur

Les propriétés typiquement intéressantes pour ce type d'application sont l'ordonnancabilité, et des propriétés quantitatives telles que le pire temps de réponse (WCRT).

L'ordonnancabilité est satisfaite si le réseau est sauf. L'espace d'états surapproximé par DBM de (Roux *et al.*, 2002, Bucci *et al.*, 2004) produit un graphe de 608 classes, toutes avec un marquage sauf. Pour vérifier l'existence de certaine exécutions, (Bucci *et al.*, 2004) propose de calculer par la programmation linéaire les échéanciers réalisables à partir de la séquence non temporisée fournie par la surapproximation. Des méthodes spécifiques sont proposées pour la vérification de propriétés quantitatives.

Pour cet exemple l'algorithme 1 construit un graphe de 323 classes et 477 transitions. Tous les marquages sont saufs ce qui implique l'ordonnancabilité. Les propriétés quantitatives peuvent être vérifiées par l'utilisation d'observateurs. La figure 2 montre un observateur non-intrusif (en gris) pour la propriété : "La tâche 3 est toujours exécutée en moins de 96 unités de temps". La propriété est satisfaite si la place `false` n'est jamais marquée. Le calcul exact de l'espace d'états confirme que la propriété est vraie et qu'elle devient fausse si l'intervalle de tir de `false` inclut 96. Le WCRT de cette tâche est donc de 96 unités de temps.

En raisonnant uniquement avec la surapproximation par DBM, nous obtiendrions un WCRT de 144 unités de temps. Enfin si le temps d'exécution de la tâche 3 ( $t_{32}$ ) est augmenté à  $[20,35]$ , alors le graphe des classes obtenu avec la surapproximation DBM devient non borné alors que le calcul exact reste fini. Dans ce cas, la méthode de (Bucci *et al.*, 2004) ne peut pas être appliquée.

#### 4. Décidabilité des SwTPN

Les problèmes de l'accessibilité d'un marquage ou d'un état, et le caractère borné, sont indécidables pour les TPN (Jones *et al.*, 1977). Il s'ensuit que ces problèmes sont également indécidables pour les SwTPN. Toutefois, ces problèmes sont décidables pour les TPN bornés. L'accessibilité d'un marquage peut alors être décidée en utilisant le graphe des classes d'états de (Berthomieu *et al.*, 1983), et l'accessibilité d'un état ainsi que la vivacité sont décidées par les constructions de (Berthomieu *et al.*, 2003). La question posée dans cette section est de savoir si ces problèmes sont décidables ou non pour les SwTPN *bornés*.

La réponse est malheureusement négative. Il est prouvé dans la suite que le problème de l'accessibilité pour les SwTPN peut être réduit à celui de l'arrêt d'une machine à deux compteurs. Après quelques rappels sur ces machines, un codage en SwTPN est proposé, et le résultat d'indécidabilité en est déduit. L'encodage utilisé est apparentée à celui utilisé dans (Henzinger *et al.*, 1995) pour démontrer l'indécidabilité de l'accessibilité pour une sous-classe d'automates hybrides, mais il est évidemment très différent, les SwTPN ne manipulant pas d'horloges explicitement.

##### 4.1. Machine à deux compteurs

Une *machine à deux compteurs* est un  $n$ -uplet  $\mathcal{M} = \langle Q, q_0, q_F, \mathcal{I}, C_1, C_2 \rangle$  tel que :

- $Q$  est un ensemble fini d'états,
- $q_0 \in Q$  est l'état initial,
- $q_F \in Q$  est l'état final ou l'état d'arrêt,
- $C_1$  et  $C_2$  sont des compteurs, chacun contenant un entier naturel, initialement 0,
- $\mathcal{I}$  est un ensemble fini d'instructions avec la forme et le sens suivants ( $i \in \{1, 2\}$ ):
  - $(p, dec_i, q)$  : dans l'état  $p$ , décrémenter  $C_i$  et aller dans l'état  $q$ ,
  - $(p, inc_i, q)$  : dans l'état  $p$ , incrémenter  $C_i$  et aller dans l'état  $q$ ,
  - $(p, test_i, q, r)$  : dans l'état  $p$ , aller dans l'état  $q$  si  $C_i = 0$ , sinon aller en  $r$ .

Une *configuration* de  $\mathcal{M}$  est un triplet  $(q, x_1, x_2)$ , où  $q \in Q$ , et  $x_1, x_2 \in \mathbf{N}$  sont les valeurs des compteurs  $C_1$  et  $C_2$ . La configuration initiale est  $c_0 = (q_0, 0, 0)$ .

Le problème de l'arrêt d'une machine à deux compteurs est indécidable (Minsky, 1961).

## 4.2. Encodage d'une machine à deux compteurs en SwTPN

### 4.2.1. Principes d'encodage et notations

Pour deux événements donnés  $e_1$  et  $e_2$ , de même période  $\rho$ , la *différence de phase* entre  $e_1$  et  $e_2$  est le temps écoulé entre une occurrence de  $e_1$  et l'occurrence suivante de  $e_2$ . Une valeur  $k$  du compteur  $C_i$  ( $i \in \{1, 2\}$ ) sera encodée par une différence de phase de  $\rho/2^{k+1}$  entre un événement  $i.clock$ , associé au compteur  $C_i$ , et un événement de référence  $tick$  (c.-à-d. les phases  $\rho/2, \rho/4, \rho/8, \dots$  encodent les valeurs  $0, 1, 2, \dots$ ). De tels encodages, d'un espace discret infini dans un espace dense borné, sont aussi utilisés dans (Čerāns, 1992) et (Henzinger *et al.*, 1995).

Observé à l'instant où une certaine place  $p$  devient marquée, la différence de phase entre  $i.clock$  et  $tick$  sera notée  $\phi_i^p$ . Les réseaux encodant les instructions sont construits à partir de cinq blocs élémentaires décrits dans la suite. La plupart des preuves sont omises, mais beaucoup paraphrasent simplement le comportement des blocs et sont très facilement obtenues. Les preuves peuvent être trouvées dans le rapport technique (Berthomieu *et al.*, 2004a).

### 4.2.2. Événements de base et bloc d'initialisation

Les événements de base que nous utilisons sont donc des tirs périodiques de transitions, avec la période  $\rho$ . Nous en avons une pour chaque compteur,  $1.clock$  et  $2.clock$ , et une pour l'événement de référence  $tick$ . La période 8 de ces tirs de transitions est découpée en  $1 + 7$  afin de créer des points de synchronisation explicites. Nous utiliserons par ailleurs deux autres événements,  $3.clock$  et  $guess$  de façon temporaire, pour l'instruction d'incrémement.

Les motifs correspondants (appelés registres) sont donnés figure 3(a). La transition *init* initialise les phases des deux compteurs à 4 (ce qui encode une valeur de comptage de 0). Les places *in* et *out* (en gris) ne font pas partie du bloc, elles matérialisent simplement le contexte d'utilisation.

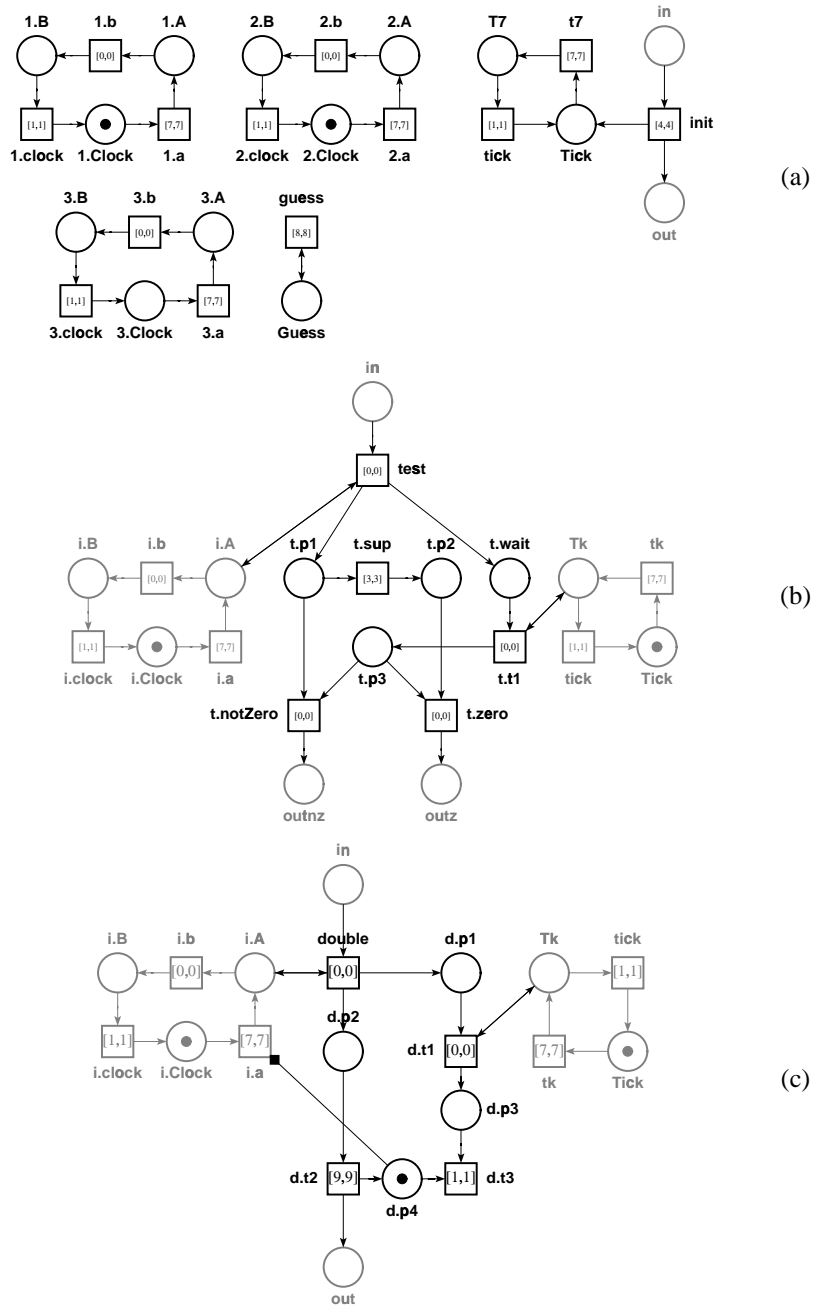
### 4.2.3. Instruction de test du zéro

Ce bloc, figure 3(b), teste si la phase d'un registre  $i$  est égale à la phase initiale 4. Comme précédemment, les noeuds et arcs en gris représentent le contexte. Un jeton dans la place *in* est propagé soit à la place *outz* (si  $\phi_i^{in} = 4$ ) soit à *outnz* (si  $\phi_i^{in} \leq 2$ ). Aucun arc activateur n'est nécessaire.

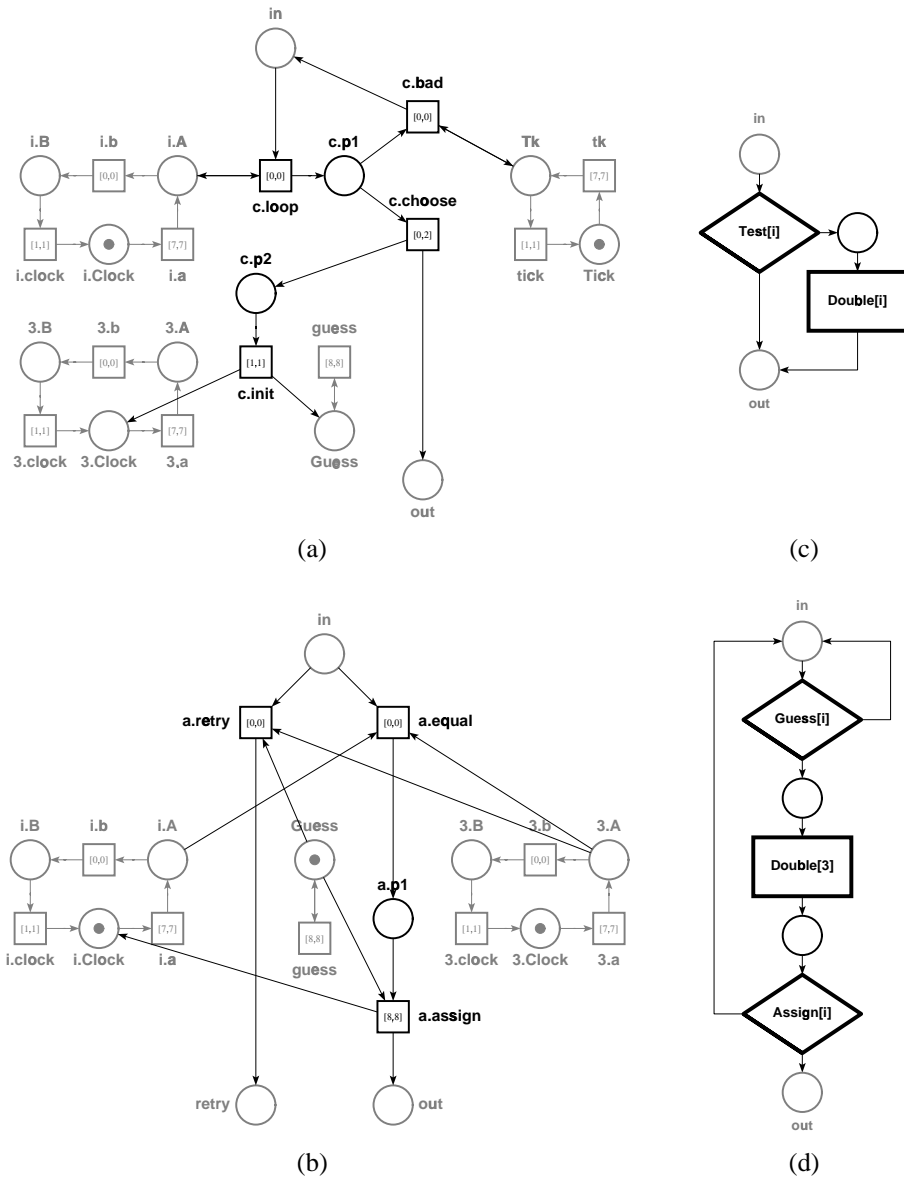
### 4.2.4. Doublement de phase

Ce bloc, figure 3(c), double la phase de  $i.clock$ , en supposant qu'elle est inférieure à 4. L'arc de la place  $d.p4$  vers  $i.a$  est un arc activateur. Supposons qu'un jeton est présent dans la place *in*, et  $\phi_i^{in} = k$ , avec  $0 < k \leq 2$ . Alors, le jeton est propagé à la place *out*, et  $\phi_i^{out} = 2 * k$ . Comme schéma de preuve, supposons que *double* tire à la date  $t$ , alors la transition  $i.a$  devient sensibilisée à la date  $t + 1$ , suspendue à la date





**Figure 3.** Registres et bloc d'initialisation (a), instruction de test à zero (b), et bloc de doublement (c)



**Figure 4.** Bloc de sélection (a), bloc de comparaison et affectation (b), instruction de décrémentation (c), et instruction d'incrémentation (d)

$t + k + 1$ , et redémarrée à  $t + 9$ . Alors,  $i.clock$  tirera de nouveau à  $t + 17 - k$ , et le  $tick$  suivant à  $t + k + 17$ , établissant  $\phi_i^{out} = 2 * k$ .

#### 4.2.5. Bloc de sélection

Ce bloc, figure 4(a), initialise les événements  $guess$  et  $3.clock$ , leur donnant une phase dans l'intervalle  $[0, \phi_i^{in}]$ .

#### 4.2.6. Comparaison et affectation

Ce bloc, figure 4(b), compare les phases de  $3.clock$  et  $i.clock$ . Un jeton dans la place  $in$  se propage soit dans  $out$  (si  $3.clock$  et  $i.clock$  sont en phase), établissant au passage la phase de  $i.clock$  à celle de  $guess$ , soit dans la place  $retry$  pour des phases quelconques de  $3.clock$  et  $i.clock$  (donc de manière non déterministe si  $3.clock$  et  $i.clock$  sont en phase).

#### 4.2.7. Instruction de décrémentation

L'instruction de décrémentation, figure 4(c), est implémentée par une copie du bloc de doublement précédée d'une copie du bloc de test. Le bloc de test empêche le doublement d'une phase plus grande que 2 (donc la décrémentation d'un compteur nul).

#### 4.2.8. Instruction d'incrémentement

Pour incrémenter le compteur  $C_i$ , il faut diviser la phase de  $i.clock$  par deux. L'instruction d'incrémentement est organisée en trois blocs, comme représenté figure 4(d). Une phase est d'abord choisie pour les compteurs temporaires  $C_3$  et  $guess$  grâce au motif de sélection. La phase de  $C_3$  est ensuite doublée en utilisant le motif de doublement, et comparée à la phase du compteur  $C_i$  en utilisant le motif de comparaison et affectation. Si les deux valeurs sont égales, cela signifie que  $guess$  a la phase de  $C_i$  divisée par 2 et nous pouvons donc affecter la valeur de  $guess$  à  $C_i$ . Dans le cas contraire, le processus peut reprendre et un autre choix initial être fait.

#### 4.2.9. Encodage de la machine à 2 compteurs

Le SwTPN  $\mathcal{N}$  encodant la machine  $\mathcal{M}$  est construit comme suit :

- Si  $\mathcal{M}$  a  $n + 1$  états  $q_0, q_1, \dots, q_n$ , alors créer  $\mathcal{N}$  avec  $n + 2$  places nommées  $start, p_0, p_1, \dots, p_n$ . La place  $start$  est la seule place marquée.
- $q_0$  étant l'état initial de la machine, ajouter à  $\mathcal{N}$  les registres et le bloc  $Init$  connecté aux places  $start$  (en entrée) et  $p_0$  (en sortie).
- Pour chaque instruction  $(q_a, dec_i, q_b)$  (resp.  $(q_a, inc_i, q_b)$ ) ajouter à  $\mathcal{N}$  une copie du bloc  $Dec_i$  (resp.  $Inc_i$ ), connectée aux places  $p_a$  (en entrée) et  $p_b$  (en sortie).
- Pour chaque instruction  $(q_a, test_i, q_b, q_c)$ , ajouter à  $\mathcal{N}$  une copie du bloc  $Test_i$ , connectée aux places  $p_a$  (en entrée),  $p_b$  (comme sortie  $outz$ ), and  $p_c$  (sortie  $outnz$ ).

### 4.3. Résultats d'indécidabilité

Soit  $\mathcal{N}$ , avec  $s_0$  pour état initial, le SwTPN encodant la machine à deux compteurs  $\mathcal{M}$ , obtenu comme indiqué ci-dessus. Pour toute configuration  $c = (q_i, x_1, x_2)$  de  $\mathcal{M}$ , et tout état  $s = (m, I)$  de  $\mathcal{N}$ , nous écrirons  $c \cong s$  ssi :

- $m$  marque  $p_i$ ,  $1.Clock$ ,  $2.Clock$ , et  $Tick$ ,
- Pour  $t \in \{1.clock, 2.clock, tick\}$ ,  $I(t)$  est un point, avec :  

$$I(tick) - I(1.clock) = 1/2^{x_1+1}$$
 et  $I(tick) - I(2.clock) = 1/2^{x_2+1}$

**Théorème 1** *Une configuration  $c$  est accessible à partir de  $c_0$  dans la machine  $\mathcal{M}$  (noté  $c_0 \rightarrow c$ ) ssi un état  $s$  est accessible à partir de  $s_0$  dans le réseau  $\mathcal{N}$  (noté  $s_0 \rightarrow s$ ) tel que  $s \cong c$ . C'est à dire :*

- (i)  $(\forall c)(c_0 \rightarrow c \Rightarrow (\exists s)(s_0 \rightarrow s \wedge s \cong c))$
- (ii)  $(\forall c)(\forall s)(s_0 \rightarrow s \wedge s \cong c \Rightarrow c_0 \rightarrow c)$

**Preuve 1** *Induction sur les instructions de  $\mathcal{M}$ , en utilisant les propriétés des blocs.*

**Théorème 2** *L'accessibilité d'état est indécidable pour les TPN à chronomètres, même bornés.*

**Preuve 2** *Le réseau  $\mathcal{N}$  est clairement borné, il est même sauf (1-borné). D'après le théorème 1, une configuration  $c$  comprenant l'état final  $q_F$  est accessible dans  $\mathcal{M}$ , (c.-à-d.  $\mathcal{M}$  s'arrête), ssi un état  $s \cong c$  est accessible dans  $\mathcal{N}$ .*

Comme corollaires, nous avons que l'accessibilité d'un marquage, le caractère k-borné, et la vivacité, sont indécidables pour les SwTPN bornés. Concernant les autres extensions des TPN modélisant la préemption, le bloc de doublement proposé peut facilement être adapté pour ces modèles (Preemptive-TPN, IHTPN et Scheduling-TPN). Le théorème 2 se généralise donc à toutes ces extensions, ce qui résout un problème ouvert.

## 5. Espace d'états approximé

Le théorème 2 met définitivement fin à tout espoir d'un algorithme de calcul de l'espace d'états abstrait pour tout SwTPN borné. En général, on se contentera de calculer des surapproximations de ces espaces, capturant tous les états accessibles, mais éventuellement plus. De telles surapproximations fournissent des conditions suffisantes pour les propriétés de sûreté.

Des surapproximations pour les Preemptive-TPN, Scheduling-TPN and IHTPN ont été proposées dans (Bucci *et al.*, 2004), (Lime *et al.*, 2003), et (Roux *et al.*, 2004), approximant les domaines temporels des classes d'états par la plus petite DBM enveloppante. La méthode est efficace, mais les approximations sont souvent trop *grossières*. D'autre part, les approximations proposées pour les automates hybrides linéaires (Alur *et al.*, 1995) sont inutilement riches pour nos objectifs.

Nous proposons ici une autre technique, basée sur une quantification des polyèdres capturés par les classes d'états, selon une discrétisation de l'espace, les polyèdres étant toujours calculés en considérant un temps dense. La technique produit des approximations plus précises que les techniques précédentes, et avec une précision ajustable.

Rappelons d'abord le théorème de Motzkin sur la décomposition des polyèdres (voir par ex. (Schrijver, 1986)) : tout polyèdre  $P$  peut être décomposé de façon unique en un polytope (un polyèdre borné, obtenu par la combinaison convexe des sommets extrémaux de  $P$ , en nombre fini) et un cône polyédral. Dans notre cas, puisque tous les polyèdres se situent dans le quadrant non négatif, les cônes sont des cônes pointés (produits par des combinaisons linéaires positives des rayons extrémaux de  $P$ ).

La méthode utilisera deux propriétés des polyèdres obtenus par l'algorithme 1 :

**Théorème 3** *Pour tout SwTPN  $\mathcal{N}$  :*

- (i) *Il existe un entier  $b$  tel que toutes les coordonnées de tous les sommets extrémaux des polyèdres calculés par l'algorithme 1 pour  $\mathcal{N}$  sont plus petites que  $b$  ;*
- (ii) *Tous les rayons extrémaux, de tous les polyèdres calculés par l'algorithme 1 pour  $\mathcal{N}$ , appartiennent à la base canonique de  $\mathbb{R}^n$ .*

**Preuve 3** *Par induction. (i) et (ii) sont vérifiés pour la classe d'état initiale et sont préservées par des dérivations de classes de l'algorithme 1. Pour (i),  $b$  est n'importe quel nombre entier plus grand que la plus grande des bornes apparaissant dans les intervalles statiques des transitions. Tout sommet extrémal est une borne finie d'un intervalle de tir, et ceux-ci peuvent seulement se déplacer vers 0 par le tir des transitions. Pour (ii), la seule étape dans l'algorithme 1 qui peut introduire des rayons obliques est (1), mais ceux-ci disparaissent après l'étape de projection (3).*

Les polyèdres caractérisant les informations temporelles dans les classes d'états (obtenues par l'algorithme 1) seront approximés de la manière suivante :

**Définition 3 (Approximations des polyèdres)** *Étant donné  $k > 0$  ( $k \in \mathbb{Q}$ ). Considérons  $\mathbb{R}_+^n$  "discrétisé" par des hypercubes de taille  $k$ , et appelons  $\mathcal{H}_k$  l'ensemble de ces hypercubes. Soit  $P \subseteq \mathbb{R}_+^n$  un polyèdre. Relativement à  $k$  :*

- *Un point  $x \in \mathbb{R}_+^n$  est approximé par l'intersection de tous les hypercubes de  $\mathcal{H}_k$  contenant  $x$ .*
- *Un polytope  $Q \subseteq \mathbb{R}_+^n$  est approximé par l'enveloppe convexe des approximations de ses sommets extrémaux.*
- *Le polyèdre  $P$  est approximé par le polyèdre  $h_k(P)$  construit à partir du cône de  $P$  et de l'approximation du polytope de  $P$ .*

Clairement, pour tous  $P$  et  $k$ ,  $h_k(P)$  contient  $P$ . De plus, en ajustant la taille de la grille  $k$ ,  $h_k(P)$  peut être choisi aussi près de  $P$  que désiré. En effet, quand  $P$  inclut ses limites alors il existe  $k$  tel que  $h_k(P) = P$ .

**Algorithme 2 (Graphe des classes d'états approximé pour une taille de grille  $k$ )**

*Le graphe des classes d'états approximé pour une taille de grille  $k$  est construit comme par l'algorithme 1, excepté que  $D'$  est approximé par  $h_k(D')$  après l'étape 3.*

Nous dirons qu'un SwTPN est intrinsèquement borné si le réseau de Petri sous-jacent (obtenu en enlevant les arcs activateurs et les contraintes temporelles) est borné.

**Théorème 4** *Pour tout  $k$  et tout SwTPN intrinsèquement borné, l'algo. 2 termine.*

**Preuve 4** *Les polyèdres approximés obéissent aussi au théorème 3, le nombre de rayons possibles est donc fini. D'autre part, le nombre d'hypercubes de côté  $k$  dans tout sous-espace borné de  $\mathbf{R}^n$  est fini, ainsi le nombre de sommets extrémaux possibles est fini, et seul un nombre fini de polyèdres peut être construit à partir de ces derniers et d'un ensemble fini de rayons. Finalement, le réseau étant intrinsèquement borné, il reste borné par la relaxation des contraintes temporelles.*

Puisque tout polyèdre peut être approximé aussi étroitement que voulu en ajustant la taille de grille  $k$ , le graphe exact des classes d'états peut lui aussi être approximé aussi étroitement que désiré, mais il n'y a pas en général de limite sur  $k$  telle que le graphe approximé coïncide avec le graphe exact (ceci contredirait le théorème 2).

Les algorithmes 1 et 2 ont été implantés dans une extension de l'outil *Tina* (Berthomieu *et al.*, 2004b). L'approximation ne nécessite que des opérations classiques sur les polyèdres, fournies par les bibliothèques existantes. Le tableau suivant donne les tailles des graphes de classes d'états du réseau de la Figure 1, pour plusieurs tailles de grille et pour plusieurs méthodes de regroupement de classes ( $\cong$ , comme utilisé par l'algorithme 1, et la variante utilisant l'inclusion de classes  $\subseteq$ , non discutée ici). Le comportement approximé devient non borné quand la taille de grille est choisie plus grande que 1 (ce réseau n'est pas intrinsèquement borné). Pour l'exemple de la figure 2, le comportement exact est obtenu pour une taille de grille de 1.

grille ( $k$ )	classes/transitions (règle $\cong$ )	classes/transitions (règle $\subseteq$ )
2	non borné	non borné
1	57/137	12/32
1/4	920/2060	18/47
1/16	29704/64436	18/47

L'approximation par la "plus petite DBM englobante" définie dans (Lime *et al.*, 2003) ou (Bucci *et al.*, 2004) est souvent trop grossière. Elle peut être améliorée en rendant la méthode paramétrable, comme la nôtre : en définissant une grille sur les espaces des polyèdres et en approximant les polyèdres  $P$  par la plus petite DBM incluant  $P$  dont les sommets extrémaux sont sur la grille plutôt qu'à coordonnées entières. Cependant, même avec cette amélioration, l'approximation par DBM englobante ne peut pas être aussi fine que l'approximation par quantification proposée.

Par exemple, la figure 5 montre les résultats des approximations avec une taille de grille  $k = 1$  de deux polyèdres simples de dimensions 2, choisis parmi ceux obtenus par l'algorithme 1 pour le réseau de la Figure 1. Pour le polyèdre de gauche, défini par  $\{0 \leq y, y \leq x \leq 3/4\}$  et dessiné en noir, les deux approximations donnent le même résultat (le triangle externe). Mais pour le polyèdre de droite, défini par  $\{0 \leq x \leq 1, x + y = 1\}$  (la diagonale noire) la méthode "par DBM englobante" conduit au carré de côté 1 (quelle que soit la taille de grille  $k \leq 1$  choisie), alors que la méthode par quantification fournit le polyèdre exact.



**Figure 5.** Polyèdre exact et surapproximations

Enfin, notons que la technique de quantification est aussi applicable aux TPN (sans chronomètres), elle permet là de réduire la taille des abstractions de l'espace d'états.

## 6. Conclusion

Cet article introduit d'abord les SwTPN, une extension simple des réseaux de Petri temporels, capable d'exprimer la suspension et la reprise de l'écoulement du temps pour les horloges associées aux transitions en fonction de conditions ne dépendant que du marquage. Le modèle peut être vu comme une simplification des IHTPN (Roux *et al.*, 2004). Ces deux modèles pourraient être fusionnés pour constituer un modèle expressivement riche pour un éventail de problèmes impliquant la préemption.

Le résultat principal de l'article est la preuve que l'accessibilité d'état est indécidable pour les SwTPN, même lorsqu'ils sont bornés. Ce résultat implique l'indécidabilité du problème de l'accessibilité d'un marquage, du caractère  $k$ -borné, et de la vivacité pour les SwTPN, ainsi que l'indécidabilité de tous ces problèmes pour toutes les autres extensions semblables connues des TPN, même lorsque les réseaux sont bornés. Ces problèmes étaient restés ouverts jusqu'ici.

Un semi-algorithme a été présenté pour calculer des représentations exactes des espaces d'états. Les expériences avec cette implémentation montrent que le calcul exact de l'espace d'états termine pour nombre d'applications pratiques.

Enfin, nous avons proposé une méthode originale pour calculer des surapproximations finies des graphes de classes d'états pour une famille de SwTPN bornés, basées sur la quantification des polyèdres représentant l'information temporelle. Elle est plus coûteuse que les méthodes disponibles, mais conduit à des approximations plus précises, et sa précision peut être paramétrée. Elle est également applicable aux TPN.

## 7. Bibliographie

- Alur R., Courcoubetis C., Halbwachs N., Henzinger T., Ho P.-H., Nicollin X., Olivero A., Sifakis J., Yovine S., « The Algorithmic Analysis of Hybrid Systems », *Theoretical Computer Science*, vol. 138, p. 3-34, 1995.
- Berthomieu B., Diaz M., « Modeling and Verification of Time Dependent Systems Using Time Petri Nets. », *IEEE Trans. on Software Engineering*, vol. 17, n° 3, p. 259-273, March, 1991.
- Berthomieu B., Lime D., Roux O., Vernadat F., Reachability problems and abstract state spaces for time Petri nets with stopwatches, Technical Report n° 04483, Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS), Toulouse, France, October, 2004a.
- Berthomieu B., Menasche M., « An Enumerative Approach for Analyzing Time Petri Nets. », *IFIP Congress Series*, vol. 9, p. 41-46, 1983.
- Berthomieu B., Ribet P.-O., Vernadat F., « The tool TINA – Construction of Abstract State Spaces for Petri Nets and Time Petri Nets », *International Journal of Production Research*, vol. 42, n° 14, p. 2741-2756, 15 July, 2004b.
- Berthomieu B., Vernadat F., « State Class Constructions for Branching Analysis of Time Petri Nets », *Proc. Tools and Algorithms for the Construction and Analysis of Systems, Springer LNCS 2619*, 2003.
- Bucci G., Fedeli A., Sassoli L., Vicario E., « Time state space analysis of real-time preemptive systems », *IEEE Trans. on Software Engineering*, vol. 30, n° 2, p. 97-111, February, 2004.
- Cassez F., Larsen K. G., « The Impressive Power of Stopwatches », *11th Int. Conf. on Concurrency Theory, University Park, P.A., USA*, Springer LNCS 1877, p. 138-152, 2000.
- Čerāns K., *Algorithmic problems in analysis of real time system specifications*, University of Latvia, Dr.sc.comp. Thesis, 1992.
- Henzinger T. A., Kopke P. W., Puri A., Varaiya P., « What's decidable about hybrid automata ? », *Proc. of the 27th Annual Symp. on Theory of Computing, ACM Press*, p. 373-382, 1995.
- Jeannet B., The Polka Convex Polyhedra library, Edition 2.0.1, <http://www.irisa.fr/prive/bjeannet/newpolka.html>, IRISA, Rennes, 2002.
- Jones N. D., Landweber L. H., Lien Y. E., « Complexity of Some Problems in Petri Nets. », *Theoretical Computer Science* 4p. 277-299, 1977.
- Lime D., Roux O. H., « Expressiveness and analysis of scheduling extended time Petri nets », *5th IFAC International Conference on Fieldbus Systems and their Applications, (FET'03)*, Elsevier Science, July, 2003.
- Merlin P. M., *A Study of the Recoverability of Computing Systems.*, PhD Thesis, Irvive, 1974.
- Minsky M., « Recursive Unsolvability of Post's problem », *Ann. of Math.*, vol. 74, p. 437-454, 1961.
- Roux O. H., Déplanche A.-M., « A t-time Petri net extension for real time task scheduling modeling », *Eur. Journal of Automation (JESA)*, 2002.
- Roux O. H., Lime D., « Time Petri Nets with Inhibitor Hyperarcs. Formal Semantics and State Space Computation », *Proc. Int. Conf. on Applications and Theory of Petri Nets (ICATPN'04)*, Bologna, Italy, 2004.
- Schrijver A., *Theory of Linear and Integer Programming*, John Wiley and Sons, NY, 1986.