Problèmes d'Accessibilité et Espaces d'États Abstraits des Réseaux de Petri Temporels à Chronomètres

Bernard Berthomieu[1], **Didier Lime**[2,3] , **Olivier H. Roux**[3] et **François Vernadat**[1]

[1] *LAAS-CNRS, 7, Avenue du Colonel Roche, 31077 Toulouse Cedex, France*
[2] *Aalborg University - CISS, Fredrik Bajers Vej 7B, 9220 Aalborg East, Denmark*
[3] *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 indecidable, 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 basée sur une quantification des polyèdres représentant le domaine temporel d'une classe d'états. En ajustant un paramètre, le comportement exact peut être approché aussi précisément que voulu.*

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 one based on a quantization of the polyhedra representing temporal information. By adjusting a parameter, the exact behavior can be approximated as closely as desired. The methods have been implemented, experiments are reported.*

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.*

1re soumission à *MSR 05*, le 4 mai 2005.

1. Introduction

Modéliser certains systèmes temps réel nécessite d'exprimer la suspension et la reprise d'actions. Pour répondre à cette nécessité, plusieurs modèles basés sur la notion de chronomètre (stopwatch) ont été proposés. Une extension des automates temporisés (TA), les automates à chronomètres (SWA), peut être définie comme une sous-classe des automates hybrides linéaires (LHA) pour laquelle les dérivées par rapport au temps des variables 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 [CAS 00]. Ce problème a été démontré indécidable pour pour LHA [ALU 95] et l'est donc pour SWA. 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 gros. Ces surapproximations fournissent des conditions suffisantes pour les propriétés de sûreté.

Les réseaux de Petri Temporels (TPN) [MER 74] 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 [BER 83] [BER 91] [BER 03]. Ces classes représentent des ensembles d'états par un marquage et un polyèdre saisissant l'information temporelle. Le problème de l'accessibilité d'états est indécidable pour les TPN, mais devient décidable pour les TPN bornés, ce qui est suffisant pour la grande majorités des cas rencontrés en pratiques.

Plusieurs extensions des TPN ont été proposées pour répondre au problème de la modélisation de la suspension et de la reprise d'action : les Scheduling-TPN [ROU 02] [LIM 03], les Preemptive-TPN [BUC 04] et les TPN à hyperarcs inhibiteurs (IHTPN) [ROU 04]. Les deux premiers ajoutent des ressources et des priorités au modèle TPN, les IHTPN introduisent des arcs inhibiteurs qui contrôlent la progression des transitions. Puisque tous étendent les TPN, le problème de l'accessibilité d'états est indécidable, mais l'accessibilité d'états lorsque le réseau est borné (qui est d'un intérêt pratique élevé), reste un problème ouvert.

Pour toutes ces extensions, des semi-algorithmes calculant une abstraction de l'espace d'états en termes de classe d'états sont disponibles. Mais, comme pour les SWA, aucune sous classe décidable suffisamment expressive n'a été identifiée. Une méthode a été proposée, permettant une surapproximation du polyèdre caractérisant le domaine temporel dans une classe d'états par le plus petit polyèdre représentable par une DBM incluant le polyèdre exact. La méthode est efficace, mais les surapproximations obtenues sont souvent trop grossières.

Dans cet article, nous introduisons un modèle simple de TPN à chronomètres (SwTPN). Les SwTPN étendent les TPN avec des *arcs activateurs* qui contrôlent la progression des transitions. Ils peuvent être vus comme une simplification des IHTPN [ROU 04].

Nous démontrons ensuite que le problème de l'accessibilité d'états est indécidable pour les SwTPN, 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 de TPN discutées 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 des classes d'états pour les TPN s'adaptent facilement au SwTPN. Ils conduisent à des abstractions exactes de l'espace d'état, mais, par suite du résultat d'indécidabilité ci-dessus, le caractère borné du SwTPN n'implique pas le caractère fini des graphes des classes d'états. Pour assurer l'arrêt sur une classe d'un SwTPN borné, nous proposons une nouvelle méthode de surapproximation basée sur la quantification des polyèdres représentant l'information temporelle dans les classes d'états. En ajustant un paramètre, le comportement exact du SwTPN peut être approché aussi étroitement que souhaité. Les méthodes de calcul exact et surapproximé ont été appliquées dans une extension de l'outil *Tina* [BER 04b].

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 ∞. Pour tout $\theta \in \mathbf{R}^+$, $i \dot{-} \theta$ représente l'intervalle $\{x-\theta | x \in i \wedge x \geq \theta\}$.

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* $\mathbf{Sw} : T \rightarrow P \rightarrow \mathbf{N}$ *est une fonction appelée* fonction d'incidence des arcs activateurs*.*

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 $(p, t) \in P \times T$. Les valeurs supérieures à 0 sont représentées par un arc particulier (éventuellement valué) appelé *arc activateur*, 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 la relation de transition temporisée $\xrightarrow{t@\theta}$, sont définis comme suit :

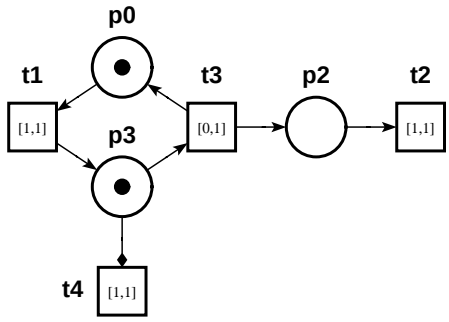

Figure 1. *Un SwTPN*

Définition 2 *Un* état *d'un SwTPN est un couple $s = (m, I)$ tel que m est un marquage et I, est la fonction* Intervalle*, qui associe un intervalle temporel dans $\mathbf{I}^+$ à chaque transition sensibilisée par m. Nous notons $(m, I) \xrightarrow{t@\theta} (m', I')$ 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) $(\forall k \in T)(m' \geq \mathbf{Pre}(k) \Rightarrow$
$I'(k) =$ **si** $k \neq t \ \wedge \ m - \mathbf{Pre}(t) \geq \mathbf{Pre}(k)$
alors si $m \geq \mathbf{Sw}(k)$ **alors** $I(k) \dot{-} \theta$ **sinon** $I(k)$
sinon $I_s(k)$

Nous avons $s \xrightarrow{t@\theta} s'$ si le tir de t à partir de s à la date (relative) θ 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 (qui restent sensibilisées) ont leur intervalle inchangé si elles étaient suspendues, ou décalé de θ (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 par la relation $\xrightarrow{t@\theta}$, à partir de son état initial $s_0 = (m_0, I_0)$, avec $I_0(t) = I_s(t)$. Une *exécution* est une séquence de tirs successifs de transitions à leurs dates relatives respectives.

2.2. *Classes d'états d'un SwTPN*

Comme pour les TPN, le nombre d'états d'un SwTPN est potentiellement infini. Les différentes constructions du graphe des classes d'états, qui fournissent des abstrac-

tions de l'espace d'états pour les TPN [BER 83] [BER 03] sont facilement adaptables aux SwTPN. Nous adoptons ici la construction proposée dans [BER 83] qui préserve les propriétés LTL.

Une classe d'états est définie par un couple (m, D), où m est un marquage et D un domaine de tir décrit pas un système d'inéquations $A\underline{\phi} \leq \underline{b}$. La variable ϕ_i représente la date à laquelle la i^{me} transition sensibilisée par m peut être tirée. Notons $(m, D = \{A\underline{\phi} \leq \underline{b}\}) \cong (m', D' = \{A'\underline{\phi} \leq \underline{b'}\})$ 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 :

Algorithm 1 (Calcul des classes d'états)
Pour toute séquence de tir σ, un couple C_σ peut être calculé comme indiqué ci-dessous. L'ensemble des classes d'états est le plus petit ensemble C incluant C_ϵ 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_σ 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 σ 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 distincts D que l'on peut obtenir par l'algorithme 1 est fini [BER 83], que le TPN soit borné ou non. Ainsi, les TPN bornés admettent des graphes des 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. Décrire les domaines de tir exige des systèmes d'inéquations plus riches, et le caractère borné du SwTPN n'implique pas le caractère fini de son ensemble de classes d'états.

Considérons comme exemple le réseau de la figure1. 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, ce qui conduit par l'algorithme 1 à une suite infinie de classe d'état, toutes ayant le même marquage. Le tir de σ_n conduit à la classe suivante :

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

Ajuster l'intervalle de t_1 à $[2, 2]$, par exemple, conduit par contre à un graphe des classes d'états fini avec 25 classes et 38 transitions.

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 des classes d'états "fortes" et "atomiques" introduits dans [BER 03] pourraient 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 l'implémentation (disponible à `http ://www.laas.fr/tina/next`) de l'algorithme 1 pour les SwTPN inclue dans une extension de l'outil *Tina* [BER 04b]. Pour des opérations sur les polyèdres, l'implémentation repose sur la bibliothèque $NewPolka$[JEA 02].

Les expériences montrent que l'espace d'états abstrait du SwTPN obtenu par l'algorithme 1 est fini dans beaucoup de cas pratiques. Pour illustrer cela, considérons l'exemple simple proposé dans [BUC 04]. Cet exemple est le modèle de 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. Cette exemple est facilement traduit des Preemptive-TPN vers les SwTPN. Le SwTPN est donné Fig. 2 (en noir).

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 (DBM) de [ROU 02, BUC 04] produit un graphe de 608 classes, toutes avec un marquage sauf. Pour vérifier si certaine exécutions sont effectivement possibles, [BUC 04] propose une méthode de calcul (par la résolution d'un problème de programmation linéaire) des séquences de tir réalisables à partir de la séquence non temporisée fournie par la surapproximation. Des méthodes spécifiques pour vérifier des propriétés quantitative sont alors proposées.

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

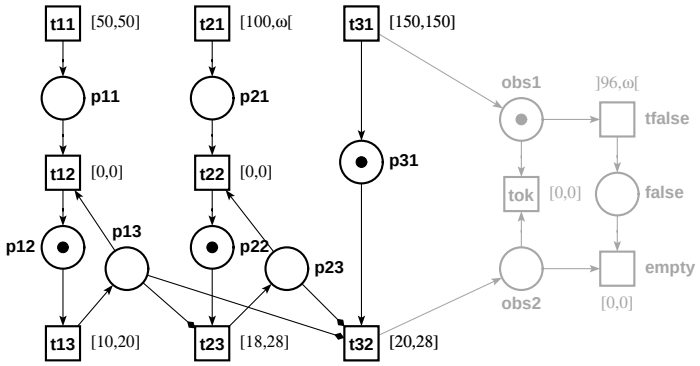

Figure 2. *SwTPN de deux tâches périodiques, une sporadique et un observateur.*

quantitatives peuvent être vérifiées par l'utilisation d'observateurs. La Fig. 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 tâches 3 (`t32`) 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 [BUC 04] 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 [JON 77]. Il s'ensuit que ces problèmes sont également indécidables pour les SwTPN.

Bien qu'indécidables dans le cas général, ces problèmes deviennent décidables pour les TPN bornés. En effet, pour les TPN bornés, l'accessibilité de marquage peut être décidée en utilisant le graphe des classes d'états de [BER 83], et l'accessibilité d'état ainsi que la vivacité sont décidées par les constructions alternatives de [BER 03]. La question posée dans cette section est de savoir si ces problèmes sont décidables pour les SwTPN *bornés*.

Malheureusement, la réponse est négative. Il est prouvé dans la suite que l'accessibilité d'état pour les SwTPN peut être réduite au problème de l'arrêt d'une machine à deux compteurs. Après un rappel de la structure de telles machines, un codage en SwTPN est proposé, et le résultat d'indecidabilité en est déduit. L'encodage utilisé est

apparentée à celui utilisé dans [HEN 95] pour démontrer l'indécidabilité de l'accessibilité pour une sous-classe d'automates hybrides, mais il est évidemment très différent, notamment parce que les SwTPN ne manipulent pas les horloges explicitement.

4.1. *Machine à deux compteurs*

Une *machine à deux compteurs (avec entrées libres)* 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, décrémente C_i et va dans l'état q,
- (p, inc_i, q) : dans l'état p, incrémente C_i et va dans l'état q,
- $(p, test_i, q, r)$: dans l'état p, teste C_i ; va dans l'état q si $C_i = 0$, sinon 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 [MIN 61].

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 ρ, 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 $\rho/2, \rho/4, \rho/8, \ldots$ encodent les valeurs $0, 1, 2, \ldots$). De tels encodages, d'un espace discret infini dans un espace dense borné, sont aussi utilisés dans [ČER 92] et [HEN 95].

Observé à l'instant ou une certaine place p devient marquée, la différence de phase entre $i.clock$ et $tick$ sera notée ϕ_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 [BER 04a].

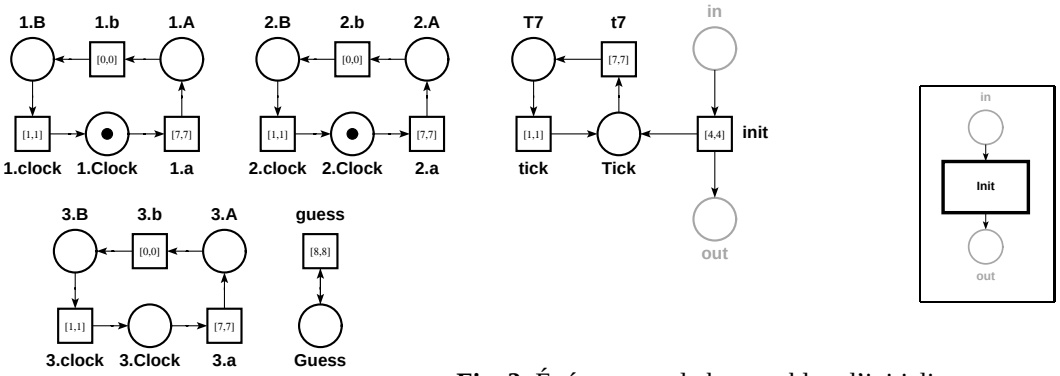

Fig. 3. Événements de base et bloc d'initialisation

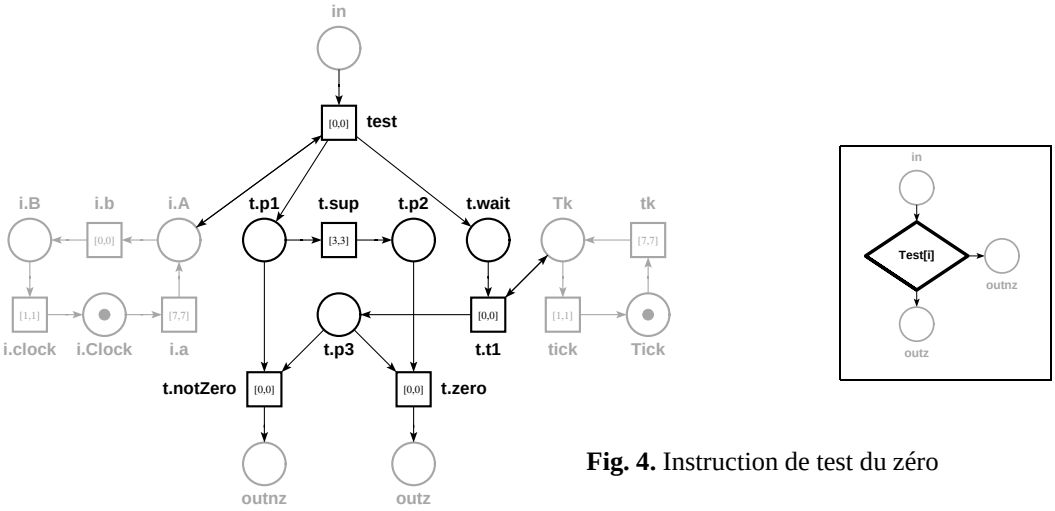

Fig. 4. Instruction de test du zéro

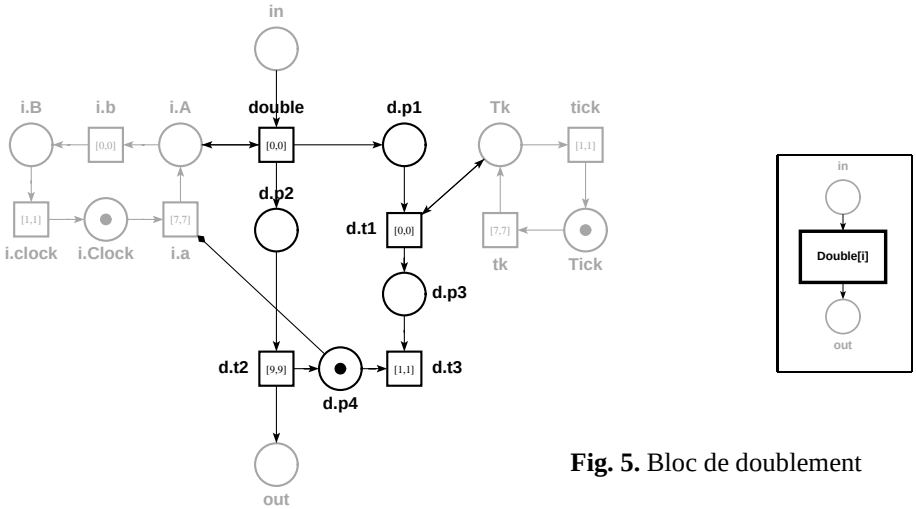

Fig. 5. Bloc de doublement

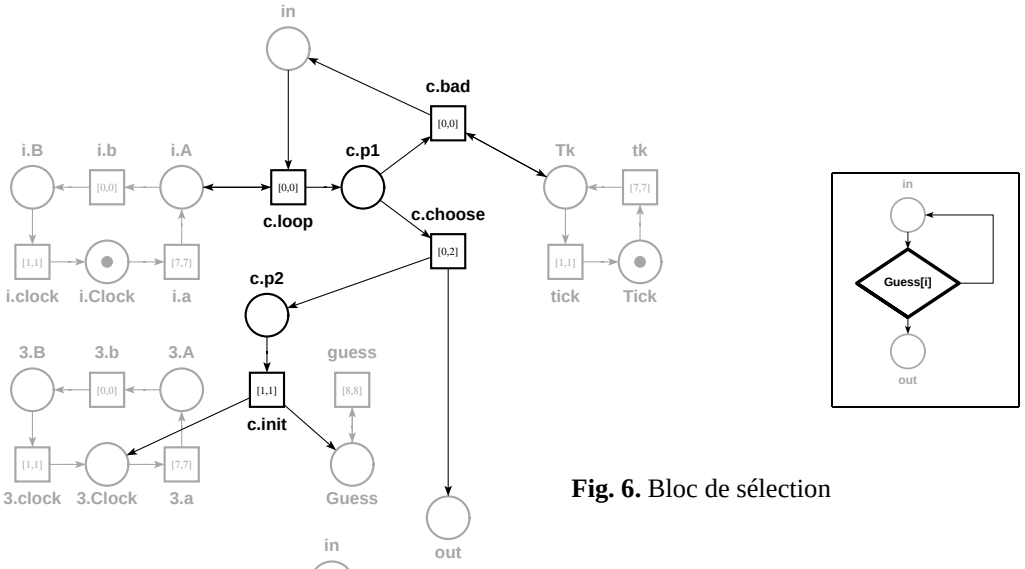

Fig. 6. Bloc de sélection

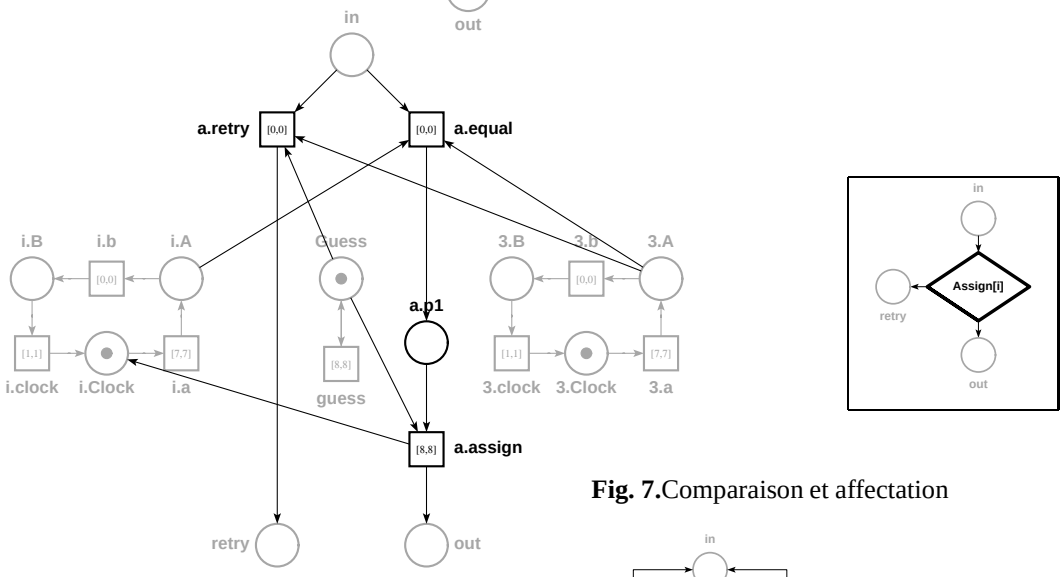

Fig. 7. Comparaison et affectation

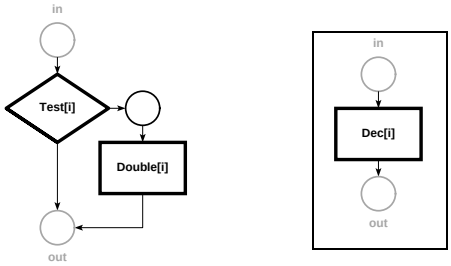

Fig. 8. Décrémentation

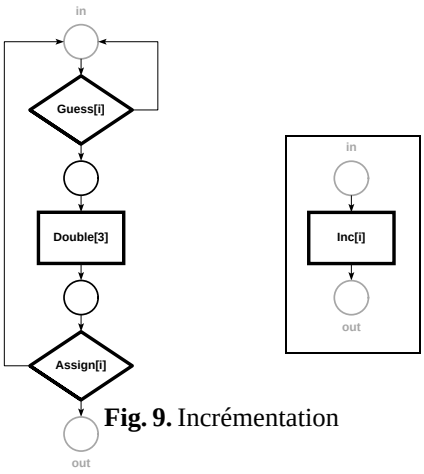

Fig. 9. Incrémentation

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 ρ. 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'incrémentation.

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

4.2.3. *Instruction de test du zéro :*

Ce bloc (Figure 4) 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 5) 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 $t + k + 1$, 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 6) initialise les registres $guess$ et 3, leur donnant une phase dans l'intervalle $[0, \phi_i^{in}]$.

4.2.6. *Comparaison et affectation :*

Ce bloc (Figure 7) compare les phases de $3.clock$ and $i.clock$. Un jeton dans la place in se propage soit dans out (si $3.clock$ et $i.clock$ sont en phase) 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 8, 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'incrémentation :*

Pour incrémenter le compteur C_i, il faut diviser la phase correspondante par deux. Le bloc d'incrémentation est organisé en trois blocs (Figure 9). Ceci est fait en choisissant une valeur au hasard pour le compteur temporaire C_3 grâce au motif de la Figure 6. Cette valeur est ensuite multipliée par deux en utilisant le motif de la Figure 5 et comparée à la valeur du compteur C_i en utilisant le motif de la Figure 7. Si les deux valeurs sont égales, cela signifie que C_3 contient la valeur de C_i divisée par 2 et nous pouvons donc affecter la valeur de C_3 à C_i. Dans le cas contraire, le processus peut reprendre.

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 $q_0, q_1, \ldots, q_n$, alors créer $\mathcal{N}$ avec $n+2$ places nommées $start, p_0, p_1, \ldots, p_n$. 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) and 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'indecidabilité*

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 *Par induction sur les transitions de la machine, 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 que nous proposons, 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 probablement plus. De tels surapproximations fournissent des conditions suffisantes pour des propriétés de sûreté.

Des surapproximations pour les Preemptive-TPN, Scheduling-TPN and IHTPN ont été proposées dans [BUC 04], [LIM 03], et [ROU 04], 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 [ALU 95] 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 exemple. [SCH 86]) : 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 polyèdral. Dans notre cas, puisque tous les polyèdres se situent dans le quadrant non négatif, les cônes sont des cônes aigus (produits par des combinaisons linéaires positives des rayons extrémaux de P).

La methode 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 petits 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 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 borne des intervalles statiques des transitions. N'importe quel sommet extrémal est une borne finie d'un intervalle de tir, et ceux-ci peuvent seulement se déplacer vers 0. Pour (ii), la seule étape dans l'algorithme 1 qui peut présenter les 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 $k > 0$ ($k \in \mathbf{Q}$). Considérons $\mathbf{R}_+^\mathbf{n}$ "discrétisé" par des hypercubes de taille k, et appelons $\mathcal{H}_k$ l'ensemble de ces hypercubes. Soit $P \subseteq \mathbf{R}_+^\mathbf{n}$ un polyèdre. Relativement à k :*

– Un point $x \in \mathbf{R}_+^\mathbf{n}$ est approximé par l'intersection de tous les hypercubes de $\mathcal{H}_k$ contenant x.

– Un polytope $Q \subseteq \mathbf{R}_+^\mathbf{n}$ 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 $h_k(P) = P$.

Le graphe des classes d'états approximé pour une taille de grille k est construit ainsi :

Algorithm 2 (Graphe des classes d'états approximé pour une taille de grille k) *De façon identique à 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 son réseau de Petri sous-jacent (obtenu en enlevant les arcs activateurs et les contraintes temporelles) est borné.

Théorème 4 *Pour tout k, l'algorithme 2, appliqué à un SwTPN intrinsèquement borné, termine.*

Preuve 4 *Les polyèdres approximés obéissent aussi au théorème 3. De plus, le nombre d'hypercubes de côté k dans n'importe quel sous-espace borné de $\mathbf{R}^\mathbf{n}$ est fini, ainsi le nombre de sommets extrémaux pour les polyèdres est fini aussi, et seul un nombre fini de polyèdres peut être construit à partir de ces derniers et d'un ensemble fini de rayons. La raison de la restriction aux SwTPN intrinsèquement bornés est que la relaxation de contraintes temporelles peut rendre non borné le comportement approximé d'un SwTPN borné mais non intrinsèquement borné.*

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 de manière générale une limite sur k tel que le graphe approximé coïncide avec le graphe exact (ceci contredirait le théorème 2).

L'algorithme 2 a été implémenté. L'approximation ne nécessite que des opérations sur les polyèdres fournies par les bibliothèques. Le tableau suivant donne les tailles des graphes des 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 içi). 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 [LIM 03] ou [BUC 04] est souvent trop grossière. Elle peut être amélioré 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 au lieu de la plus petite DBM (incluant P) avec des sommets extrémaux entier.

Cependant, même avec cette amélioration, l'approximation "plus petite DBM" englobante" ne peut pas approximer les polyèdres aussi finement que l'approximation par quantification. La raison est simple : le polyèdre peut être défini par des contraintes non-DBM non redondantes, ce qui se produit fréquemment en pratique.

La figure 10 montre les résultats des approximations avec la 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 (la triangle externe). Mais pour le polyèdre de droite, défini par $\{0 \leq x \leq 1,\ x + y = 1\}$ (la diagonale noire) la méthode "plus petite DBM englobante" conduit au carré de coté 1 (quelle que soit la taille de grille $k \leq 1$ choisie), alors que la méthode par quantification fournit le polyèdre exact.

Enfin, précisons que la méthode par quantification est applicable aussi aux TPN (sans chronomètres) ce qui peut être utile dans les cas où les abstractions de l'espace d'état sont de tailles importantes.

Figure 10. *Polyèdre exact et surapproximations.*

6. Conclusion

Cet article présente d'abord une extension simple des réseaux de Petri temporels avec des arcs activateurs, capables d'exprimer la suspension et la reprise de l'écoulement des temps pour les horloges associées aux transitions en fonction de conditions ne dépendant que des marquages. Le modèle peut être vu comme simplification des IHTPN de [ROU 04]. Les dispositifs des IHTPN (hyperarcs inhibiteurs) pourraient être fusionnés avec ceux des SwTPN 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 de l'accessibilité de 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 se termine sur beaucoup d'applications pratiques.

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

Bien que conceptuellement proches, les techniques proposées sur les SwTPN demandent plus de calcul que celles, similaires, utilisées pour les TPN. Un travail futur s'intéressera aux aspects algorithmiques pour une implémentation efficace.

7. Bibliographie

[ALU 95] 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, 1995, p. 3-34.

[BER 83] BERTHOMIEU B., MENASCHE M., « An Enumerative Approach for Analyzing Time Petri Nets. », *IFIP Congress Series*, vol. 9, 1983, p. 41–46, Elsevier Science Publ. Comp. (North Holland).

[BER 91] BERTHOMIEU B., DIAZ M., « Modeling and Verification of Time Dependent Systems Using Time Petri Nets. », *IEEE Transactions on Software Engineering*, vol. 17, n° 3, 1991, p. 259–273.

[BER 03] 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 (TACAS'2003), Springer LNCS 2619*, 2003.

[BER 04a] BERTHOMIEU B., LIME D., ROUX O., VERNADAT F., « Reachability problems and abstract state spaces for time Petri nets with stopwatches », rapport n° 04483, octobre 2004, Laboratoire d'Analyse et d'Architecture des Systèmes (LAAS), Toulouse, France.

[BER 04b] 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, 2004, p. 2741–2756.

[BUC 04] BUCCI G., FEDELI A., SASSOLI L., VICARIO E., « Time state space analysis of real-time preemptive systems », *IEEE transactions on software engineering*, vol. 30, n° 2, 2004, p. 97–111.

[CAS 00] CASSEZ F., LARSEN K. G., « The Impressive Power of Stopwatches », *11th Int. Conf. on Concurrency Theory, (CONCUR'2000), University Park, P.A., USA*, Springer LNCS 1877, 2000, p. 138–152.

[ČER 92] ČERĀNS K., *Algorithmic problems in analysis of real time system specifications*, University of Latvia, Dr.sc.comp. Thesis, 1992.

[HEN 95] HENZINGER T. A., KOPKE P. W., PURI A., VARAIYA P., « What's decidable about hybrid automata ? », *Proceedings of the 27th Annual Symposium on Theory of Computing (STOC), ACM Press*, 1995, p. 373–382.

[JEA 02] JEANNET B., « The Polka Convex Polyhedra library, Edition 2.0.1 », http ://www.irisa.fr/prive/bjeannet/newpolka.html, 2002, IRISA, Rennes.

[JON 77] JONES N. D., LANDWEBER L. H., LIEN Y. E., « Complexity of Some Problems in Petri Nets. », *Theoretical Computer Science 4*, , 1977, p. 277–299.

[LIM 03] 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, juillet 2003.

[MER 74] MERLIN P. M., *A Study of the Recoverability of Computing Systems.*, Irvine : Univ. California, PhD Thesis, 1974.

[MIN 61] MINSKY M., « Recursive Unsolvability of Post's problem », *Ann. of Math*, vol. 74, 1961, p. 437–454.

[ROU 02] ROUX O. H., DÉPLANCHE A.-M., « A t-time Petri net extension for real time task scheduling modeling », *Eur. Journal of Automation (JESA)*, vol. 36, n° 7, 2002.

[ROU 04] 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, (to appear).

[SCH 86] SCHRIJVER A., *Theory of Linear and Integer Programming*, John Wiley and Sons, NY, 1986.