Synthèse de contrôleur pour les réseaux de Petri temporels basée sur les classes d'états*

Loriane Leclercq, Didier Lime, and Olivier H. Roux

Nantes Université, École Centrale Nantes, CNRS, LS2N, UMR 6004, F-44000 Nantes
{loriane.leclercq,didier.lime,olivier-h.roux}@ec-nantes.fr

Résumé

Nous proposons un nouvel algorithme pour la synthèse de contrôleur garantissant l'accessibilité dans les réseaux de Petri temporels (TPN) ainsi que dans leur extension paramétrée (PTPN). Nous considérons une sémantique atypique pour les réseaux de Petri, dans laquelle la date de tir d'une transition est choisie dans son intervalle statique au moment où elle devient sensibilisée. Cette sémantique est motivée i) *par une considération pratique :* elle tend à approcher l'implémentation d'un contrôleur temps-réel ; ii) *par une préoccupation théorique :* elle garantit que dans le graphe de classes d'états (SCG) [6], tout état d'une classe d'états est un état atteignable dans le PTPN, ce qui n'est pas le cas dans la sémantique habituelle basée sur les intervalles. Nous définissons un nouveau type de jeu temporisé à deux joueurs sur le SCG afin de calculer efficacement et symboliquement les états gagnants et les paramètres à l'aide des classes d'états. Cette approche est implémentée dans le logiciel Roméo [18]. Nous l'illustrons par un petit cas d'étude.

1 Introduction

Les systèmes réactifs permettent à de multiples composantes d'interagir ensemble et avec l'environnement. Afin de garantir leur fonctionnement correct, des contrôleurs sont utilisés pour limiter leur comportement. Il s'agit alors de modéliser l'interaction entre un contrôleur et son environnement à l'aide d'actions contrôlables et non contrôlables (des transitions dans notre cas). La théorie du contrôle a d'abord été définie pour les systèmes à évènements discrets dans [20], puis étendue à divers modèles. Le problème de contrôle consiste à concevoir un contrôleur qui garantit qu'une spécification est valide, quelle que soit l'action de l'environnement, c'est-à-dire que le système se comporte correctement par rapport à des propriétés. Parmi les propriétés de base, on trouve l'atteignabilité et son dual, la sûreté. Étant donné un état du système, le *problème d'atteignabilité* consiste à décider si cet état est atteignable à chaque exécution.

Jeux temporisés et réseaux de Petri temporels paramétrés pour le contrôle Les jeux sur les graphes [21] constituent un cadre classique et efficace pour la synthèse de contrôleurs dans les systèmes réactifs. Cependant, nombre de ces systèmes ont de fortes exigences temporelles et doivent être modélisés à l'aide de formalismes temporels tels que les automates temporisés (TAs) [1] ou encore les réseaux de Petri temporels (TPNs) [6, 8]. Cela conduit à l'étude théorique des jeux temporisés [19]. Des algorithmes efficaces basés sur des horloges et utilisant ce que l'on appelle les zones pour représenter l'espace d'états ont été élaborés. Ils sont implémentés, par exemple dans le logiciel de référence Uppaal-Tiga [4] ou le logiciel Roméo [18], où les TPNs y sont même étendus pour prendre en compte les paramètres temporels [16]. L'algorithme basé

*Ce travail a été partiellement financé par les projets ANR ProMiS ANR-19-CE25-0015 et BisoUS ANR-22-CE48-0012

sur les zones d'horloges est cependant confronté à un problème de terminaison qui peut être résolu par une opération d'extrapolation/approximation amenant de nouvelles problématiques.

La sémantique classique des TPNs n'est pas celle des « horloges explicites » de [12], mais plutôt la sémantique basée sur les intervalles de [6, 8]. Cette dernière sémantique conduit naturellement à un type d'abstraction différent appelé classes d'états, qui présente certains avantages. En particulier, aucune utilisation d'opération d'extrapolation/approximation n'est nécessaire pour assurer la terminaison. Le problème de contrôle des TPNs a ainsi été étudié à l'aide de différentes approches dans [13, 3, 17]. Les preuves, omises ici pour des raisons de place, peuvent être trouvées dans [17] que nous étendons au cas paramétré.

En effet, une connaissance complète des données temporelles du système n'est pas toujours possible dans les premières phases de conception et de synthèse du contrôleur. Les automates temporisés paramétrés [2] et les réseaux de Petri temporels paramétrés [22] étendent respectivement les TAs et les TPNs pour s'affranchir de ces limites. La notion de jeu temporisé est aussi étendue aux paramètres [15] pour résoudre le problème suivant : « Existe-t-il des valeurs des paramètres telles qu'il existe un contrôleur garantissant un objectif d'accessibilité quelle que soit l'évolution de l'environnement ? Si oui, synthétiser ces valeurs et le contrôleur associé. » L'approche est basée sur des zones paramétrées et un algorithme en arrière.

Implémentation du contrôleur Implémenter un contrôleur temporisé sur une cible matérielle telle qu'un microcontrôleur n'est pas une opération triviale. Si une action contrôlable doit être exécutée après une durée d'attente comprise dans un intervalle de temps $[a, b]$, il est nécessaire de choisir d'abord une durée d dans cet intervalle et ensuite d'entrer en attente non active, ce qui est généralement réalisé en utilisant un *timer* de durée d qui déclenchera une interruption. Le programme associé à cette interruption exécute alors l'action contrôlable. Si une action de l'environnement se produit entre temps et nécessite une modification de la durée d, alors le contrôleur doit avoir la capacité de changer la valeur du timer. C'est ce qui est implicitement considéré dans des travaux sur la synthèse de contrôleurs temporisés pour les TPNs et automates temporisés dont le calcul est basé sur un algorithme « en arrière » et sur les horloges [19, 9, 11]. Cette réévaluation implicite des durées d'attente rend l'implémentation du contrôleur difficile (à moins d'utiliser l'attente active, le polling, ce qui n'est pas acceptable dans un contexte temps réel) En effet, on ne sait pas a priori quelles actions doivent entraîner la réévaluation des durées.

Contribution Nous proposons une sémantique atypique des réseaux de Petri temporels dans laquelle la date de tir d'une transition est choisie lorsqu'elle est sensibilisée. Cette sémantique est motivée par un aspect pratique : approcher l'implémentation d'un contrôleur temps réel. Le choix de la valeur du *timer* doit être fait dès que la transition contrôlable est sensibilisée. Si la valeur doit être réévaluée, le réseau de Petri doit le modéliser explicitement. Une autre motivation est théorique : une correspondance étroite entre cette sémantique et la construction du graphe de classes d'états (SCG) [6, 8]. Elle garantit que dans le SCG, chaque état de chaque classe d'états est réellement un état accessible du réseau de Petri, ce qui n'est pas le cas avec la sémantique usuelle par intervalles. Cette sémantique a déjà été étudiée dans [7], motivée par l'utilisation de dates de tir dynamiques, qui peuvent être réévaluées à chaque tir de transition.

Nous étendons cette sémantique avec des paramètres temporels pour définir la classe des PTPNs. Un nouveau type de jeu temporisé basé sur les PTPNs est défini et nous montrons comment calculer efficacement et symboliquement les états gagnants à l'aide des classes d'états. Notre méthode calcule en arrière les états gagnant sur le SCG en utilisant des opérations de prédécesseurs pour diviser les classes d'états. Cette approche, implémentée dans le logiciel

Roméo [18], est appliquée à un petit cas d'étude.

L'article est organisé comme suit : la Section 2 présente notre nouvelle sémantique des PTPNs et fournit les définitions nécessaires pour la construction du jeu à deux joueurs sur le SCG, la Section 3 décrit le calcul des états gagnants pour le contrôleur, conduisant à la stratégie et la Section 4 applique notre approche à un cas d'étude. Nous concluons en Section 5.

2 Définitions

2.1 Préliminaires

On désigne l'ensemble des entiers naturels (y compris 0) par $\mathbb{N}$ et l'ensemble des nombres réels par $\mathbb{R}$. On note $\mathbb{R}_{\geq 0}$ l'ensemble des nombres réels positifs ou nuls. Pour $n \in \mathbb{N}$, on désigne par $[\![0, n]\!]$ l'ensemble $\{i \in \mathbb{N} \mid i \leq n\}$. Soit un ensemble fini X, on note sa taille $|X|$. Étant donné un ensemble X, nous désignons par $\mathcal{I}(X)$, l'ensemble des intervalles réels non vides et non nécessairement bornés, dont les extrémités se trouvent dans X. On dit qu'un intervalle I est positif ou nul si $I \subseteq \mathbb{R}_{\geq 0}$.

Soient V et X des ensembles, une V-valuation (ou simplement valuation quand V est clair par le contexte) de X est une fonction de X dans V. On note par V^X l'ensemble des V-valuations de X. Lorsque X est fini, étant donné un ordre arbitraire fixé sur X, nous considérerons souvent les V-valuations comme des vecteurs de $V^{|X|}$ de manière équivalente.

2.2 Réseaux de Petri temporels

Un réseau de Petri temporel (TPN) est un réseau de Petri avec des intervalles de temps associés à chacune de ses transitions. Nous proposons une sémantique légèrement différente de celle communément utilisée, dans laquelle les dates de tirs sont décidées au moment où la transition est nouvellement sensibilisée. Les PTPNs en sont une extension dans lesquels les bornes des intervalles des transitions peuvent être des paramètres.

Définition 1 (Réseau de Petri temporel paramétré)**.** *Un* réseau de Petri temporel paramétré (PTPN) *est un tuple* $\mathcal{N} = (P, T, P_a, F, I_s)$ *dans lequel :* P *est un ensemble fini non vide de* places*,* T *est un ensemble fini de* transitions*, tel que* $T \cap P = \emptyset$*,* P_a *est un ensemble fini de* paramètres *tels que* $\forall q \in P_a, q \in \mathbb{R}$*,* $F : (P \times T) \cup (T \times P)$ *est une* relation de flot*,* $I_s : T \to \mathcal{I}(\mathbb{N} \cup P_a)$ *est la fonction d'*intervalle statique*.*

Les places d'un réseau de Petri peuvent contenir des *jetons*. Un *marquage* est alors généralement un $\mathbb{N}$-valuation de P donnant le nombre de jetons dans chaque place.

On suppose que T contient au moins une transition t_{init} et P au moins une place p_0 tel que $(p_0, t_{\mathsf{init}}) \in F$, $I_s(t_{\mathsf{init}}) = [0, 0]$, pour tout $p \in P \setminus \{p_0\}$, $(p, t_{\mathsf{init}}) \notin F$ et pour tout $t \in T \setminus \{t_{\mathsf{init}}\}$, $(p_0, t) \notin F$. On suppose aussi que pour tout $t \in T$, il existe $p \in P$ telle que $(p, t) \in F$.

Remarque 1. *Dans la suite, nous ne considérons que des réseaux* saufs*, c'est-à-dire des réseaux dans lesquels il y a toujours au plus* 1 *jeton dans chaque place et où tous les arcs ont un poids de* 1*. Tous les développements suivants peuvent être généralisés sans difficulté à des dynamiques discrètes plus complexes, à condition que le réseau reste borné, c.-à-d. qu'il existe une constante* K *telle que toutes les places ne contiennent jamais plus de* K *jetons. Cette restriction est appropriée puisque le problème de contrôle est indécidable pour les TPN non bornés.*

Nous définissons donc un marquage comme l'ensemble de places de P contenant un jeton On dit que ces places sont *marquées*.

Généralement, on définit un marquage initial pour le réseau. Sans perte de généralité, nous considérons ici que toutes les places sont initialement vides sauf p_0 qui est marquée. Une transition immédiate t_{init} met en place le marquage initial. Par construction, tant qu'elle n'a pas été tirée, aucune autre transition ne peut être tirée.

Étant donné une transition t, nous définissons les ensembles de ses places d'entrée $\mathsf{Pre}(t) = \{p \mid (p,t) \in F\}$ et de ses places de sortie $\mathsf{Post}(t) = \{p \mid (t,p) \in F\}$.

Définition 2 (Transitions sensibilisées et persistantes)**.** *Une transition t est dite* sensibilisée *par un marquage m si toutes ses places d'entrée sont marquées : $\mathsf{Pre}(t) \subseteq m$. Une transition t est dite* persistante *pendant le tir d'une transition t' à partir d'un marquage m si elle n'est pas tirée et toujours sensibilisée quand on retire les jetons des places d'entrée de t' : $t \neq t'$ et $\mathsf{Pre}(t) \subseteq m \setminus \mathsf{Pre}(t')$. On dit que t est* nouvellement sensibilisée *en tirant t' depuis m, si t est sensibilisée après le tir de t', mais non persistante. Nous désignons par $\mathsf{en}(m)$, $\mathsf{pers}(m,t)$ et $\mathsf{newen}(m,t)$ resp. les ensembles de transitions sensibilisées, persistantes et nouvellement sensibilisées.*

Remarque 2. *La définition des transitions nouvellement sensibilisées utilise une politique de réinitialisation dans laquelle chaque transition dont un jeton d'entrée est pris par $\mathsf{Pre}(t')$ est considérée nouvellement sensibilisée, même si elle est sensibilisée à nouveau après avoir remis en place les jetons de $\mathsf{Post}(t')$. Et en particulier la transition tirée elle-même est toujours considérée comme nouvellement sensibilisée. Il s'agit de la politique de mémoire classique appelée sémantique intermédiaire, voir [5] pour des détails et une comparaison entre sémantiques.*

Définition 3 (États et sémantique d'un PTPN)**.** *Un* état *d'un PTPN est un triplet $s = (m, \theta, v_p)$ avec $m \subseteq P$ un marquage, v_p une valuation des paramètres et $\theta : T \to \mathbb{R}_{\geq 0} \cup \{\bot\}$ une fonction qui associe une date de tir à chaque transition t sensibilisé par le marquage m ($t \in \mathsf{en}(m)$) et $\bot$ à toutes les autres transitions. Pour chaque valuation θ sur les transitions, on note $\mathsf{tr}(\theta)$ l'ensemble des transitions t telles que $\theta(t) \neq \bot$. On utilisera θ_i pour désigner $\theta(t_i)$.*

La sémantique d'un PTPN est un système de transitions temporisé *$(S, s_0, \Sigma, \to)$ avec : S l'ensemble des états possibles ; dont un état initial $s_0 = (\{p_0\}, \theta_0, v_p) \in S$ avec $\theta_0(t_{\mathsf{init}}) = 0$, et $\forall t \neq t_{\mathsf{init}}, \theta_0(t) = \bot$; un alphabet d'étiquetage Σ divisé en deux types de lettres : $t_f \in T$ et $d \in \mathbb{R}_{\geq 0}$, la relation de transition entre états $\to \subseteq S \times \Sigma \times S$ et, $(s, a, s') \in \to$ est noté $s \xrightarrow{a} s'$:*

- *soit $(m, \theta, v_p) \xrightarrow{t_f} (m', \theta', v'_p)$ pour $t_f \in T$ si :*
  1. *$t_f \in \mathsf{en}(m)$, $\theta_f = 0$ et $v_p = v'_p$*
  2. *$m' = (m \setminus \mathsf{Pre}(t_f)) \cup \mathsf{Post}(t_f)$*
  3. *$\forall t_k \in T, \theta'_k \in I_s(t_k)$ si $t_k \in \mathsf{newen}(m, t_f)$, $\theta'_k = \theta_k$ si $t_k \in \mathsf{pers}(m, t_f)$, et $\theta'_k = \bot$ sinon*
- *ou $(m, \theta, v_p) \xrightarrow{d} (m, \theta', v'_p)$ si : $d \in \mathbb{R}_{\geq 0} \setminus \{0\}$, $\forall t_k \notin \mathsf{en}(m), \theta'_k = \bot$, et $\forall t_k \in \mathsf{en}(m), \theta_k - d \geq 0$, $\theta'_k = \theta_k - d$ et $v_p = v'_p$.*

Remarque 3. *Si nous n'avions pas considéré qu'une unique transition t_{init} était sensibilisée avec un intervalle de temps réduit à $[0, 0]$, on aurait eu en général une infinité de fonctions θ_0 correspondant à tous les choix possibles des dates de tirs dans les intervalles statiques des transitions sensibilisées. Ce n'est pas un problème, mais un petit inconvénient pour la construction du jeu à deux joueurs qui va suivre. Nous aurions eu besoin d'un premier demi tour pour atteindre un état correct avant même de commencer. Une infinité de valuations pour les paramètres ne pose pas ce souci car il est seulement nécessaire qu'elle soit identique à tous les états qui se succèdent et ne prend pas part aux tours du jeu comme nous le verront par la suite.*

Une *exécution* dans la sémantique d'un PTPN est une séquence possiblement infinie $s_0 a_0 s_1 a_1 s_2 a_2 \cdots$ telle que pour tout i, $s_i \xrightarrow{a_i} s_{i+1}$. On désigne par $\mathsf{seq}(\rho)$ la sous-séquence de ρ contenant exactement les transitions $a_0 a_1 a_2 \cdots$.

Dans cette sémantique le choix de date de tir intervient directement quand la transition est nouvellement sensibilisée, tandis que ce choix est reporté au moment du tir de la transition dans la sémantique classique de [6]. Cette approche est déjà présente dans [14] mais avec des choix probabilistes au lieu de choix non déterministes. Nous avons choisi cette sémantique car elle nous permet de relier plus précisément les états du réseau et les classes d'états comme définies dans la prochaine section. Une relation aussi étroite n'a jamais été obtenue avec la sémantique par intervalle de [6], ce qui a conduit à des raffinements supplémentaires des classes.

2.3 Classes d'états paramétrées

Le nombre d'états d'un TPN est infini en général, en raison de la densité des intervalles statiques. Il existe plusieurs représentations finies abstrayant l'espace d'états d'un TPN par diverses méthodes et l'une d'elles est le graphe de classes d'états (SCG). Un de ses avantages est d'être fini tant que le TPN est borné, c'est-à-dire que le nombre de jetons dans chaque place est borné (par 1 dans le cas d'un réseau sauf). La construction classique s'étend au cas paramétré.

Définition 4 (Classe d'états paramétrée)**.** *Soit $\sigma = t_1 ... t_n$ une séquence de transitions. La* classe d'états paramétrée *K_σ est l'ensemble de tous les états obtenus en tirant σ dans l'ordre, avec tous les délais possibles* avant *chaque transition tirée, ainsi que toutes les valuations possibles pour les paramètres. Clairement, tous les états dans K_σ partagent le même marquage m. Donc on écrit $K_\sigma = (m, D)$ où D, appelé le* domaine de tir*, est l'union de toutes les fonctions de dates de tir et paramètres possibles pour ces états. Un point dans ce domaine est (θ, v_p) avec v_p une valuation des paramètres dans P_a et θ les dates de tirs des transitions sensibilisées.*

Le domaine de tir D est un ensemble de valuations des transitions et paramètres. Avec un ordre arbitraire sur les transitions et paramètres, et en ignorant les valeurs $\perp$, une telle valuation peut être vue comme un point dans $\mathbb{R}_{\geq 0}^{|\mathsf{en}(m)|+|P_a|}$. Nous considérerons donc de tels ensembles de valuations comme des sous-ensembles de $\mathbb{R}_{\geq 0}^{|\mathsf{en}(m)|+|P_a|}$. Et comme nous le verrons, dans cet espace les domaines de tir sont en fait un type particulier de polyèdres convexes.

Une conséquence directe de la Définition 4 est une correspondance entre les états du PTPN atteignables par le tir d'une séquence de transitions σ et les états dans la classe K_σ.

Remarque 4. *Avec la sémantique classique de [6], la partie temps des états assigne des intervalles aux transitions sensibilisées et un intervalle arbitraire pris dans D ne correspond pas nécessairement à un état atteignable. Un état peut contenir un intervalle chevauchant deux intervalles adjacents regroupés dans une classe, mais qui n'est pas accessible.*

Nous pouvons maintenant naturellement étendre les notions de sensibilisées, persistantes, et nouvellement sensibilisées des transitions aux classes d'états : $\mathsf{en}((m, D)) = \mathsf{en}(m)$, $\mathsf{newen}((m, D), t) = \mathsf{newen}(m, t)$, et $\mathsf{pers}((m, D), t) = \mathsf{pers}(m, t)$.

L'algorithme classique de calcul du graphe de classes d'états de [6] permet de calculer $K' = (m', D')$ depuis $K = (m, D)$ par le tir de la transition tirable t_f. Dans cette construction, D' n'est pas vide si et seulement si il existe θ dans D tel que pour tout $i \in \mathsf{en}(m)$, $\theta_i \geq \theta_f$. Dans ce cas, on dit que t_f est *tirable* depuis (m, D).

Remarque 5. *L'intégration des paramètres dans le domaine de tir permet de réduire automatiquement le domaine des paramètres aux évaluations qui rendent la classe atteignable, bien qu'ils ne participent pas à la construction des classes de successeurs.*

La classe d'états associée avec la séquence vide ϵ contient l'ensemble des états initiaux : $K_\epsilon = (m_0, (\{\theta_0\}, \mathbb{Q}_{\geq 0}^{|P_a|}))$. Il est bien connu que ces classes d'états peuvent être représentées et calculées à l'aide d'un type particulier de polyèdres convexes encodés dans la structure de données efficace appelée *matrice de différence bornée* (DBM) [6, 10], mais cela ne s'étend malheureusement pas au cas paramétré.

Définition 5. *En partant de K_ϵ, on peut construire un arbre infini dirigé (étiqueté par les transitions) en calculant inductivement les successeurs par les transitions tirables. Le* graphe de classes d'états paramétrées (PSCG) *$\mathcal{G}$ est le graphe obtenu en quotientant cet arbre avec une relation d'égalité sur les classes d'états (même marquage, domaine de tir et de paramètres).*

2.4 Jeu à deux joueurs sur le graphe de classes d'états

Comme nous nous intéressons à la synthèse de contrôleur, à partir de maintenant, l'ensemble des transitions est partitionné en deux ensembles T_c et T_u qui contiennent respectivement les transitions contrôlables et incontrôlables. Les transitions contrôlables sont contrôlées par un contrôleur, en ce sens qu'il peut choisir leurs dates de tir, et l'ordre de déclenchement, mais des transitions incontrôlables peuvent être tirées entre elles.

On définit $\mathsf{newen_u}(m,t) = T_u \cap \mathsf{newen}(m,t)$, $\mathsf{newen_c}(m,t) = T_c \cap \mathsf{newen}(m,t)$, $\mathsf{en_u}(m) = T_u \cap \mathsf{en}(m)$ et $\mathsf{en_c}(m) = T_c \cap \mathsf{en}(m)$. Ces notations sont étendues aux classes d'états.

On définit maintenant un jeu sur le PTPN $\mathcal{N}$ qui simule le comportement des transitions contrôlables et incontrôlables afin de décider si un ensemble d'états est toujours atteignable en choisissant les bons paramètres et dates de tir contrôlables ou pas. Et si c'est le cas, une stratégie sera construite pour le contrôleur.

Avant le début du jeu, le contrôleur va choisir une valuation pour les paramètres. Et à partir de ce moment, le jeu se déroulera sur un TPN sans paramètres.

Un tour de jeu se déroule en trois étapes : i) le contrôleur choisit une transition tirable $t_c \in T_c$ qu'il veut tirer en premier ; ii) l'environnement choisit soit de tirer une transition tirable $t_u \in T_u$, soit de laisser le contrôleur tirer t_c ; iii) les deux choisissent indépendamment les dates de tir de leurs transitions nouvellement sensibilisées.

Définition 6. *Soit $\mathcal{N} = (P, T, F, I_s)$, un PTPN avec $T = T_c \cup T_u$ et $T_c \cap T_u = \emptyset$ et $(S, s_0, \Sigma, \to)$, sa sémantique. Une* arène *est un tuple $\mathcal{A} = (S, \to, Pl, (\mathsf{Movt}_i)_{i\in Pl}, (\mathsf{Movf}_i)_{i\in Pl}, \mathsf{Trans})$ avec :*

- *$Pl = (Pl_u, Pl_c)$ les deux joueurs du jeu : l'environnement (Pl_u) et le contrôleur (Pl_c). Le contrôleur joue sur des transitions contrôlables, tandis que l'environnement joue sur des transitions incontrôlables.*
- *$\mathsf{Movt}_u : S \times T_c \to 2^T$ et $\mathsf{Movt}_c : S \to 2^T$ déterminent les choix de transitions :*

$$\mathsf{Movt}_c(m,\theta) = \{t_i \mid t_i \in \mathsf{en_c}(m) \wedge \theta_i = \min_{t_k \in \mathsf{en}(m)} \theta_k\}$$
$$\mathsf{Movt}_u((m,\theta),t_c) = \{t_i \mid t_i \in \mathsf{en_u}(m) \wedge \theta_i = \min_{t_k \in \mathsf{en}(m)} \theta_k\} \cup \{t_c\}$$

- *$\mathsf{Movf}_u : S \times T \to 2^{\mathbb{R}_{\geq 0}^{T_u}}$ et $\mathsf{Movf}_c : S \times T \to 2^{\mathbb{R}_{\geq 0}^{T_c}}$ déterminent les choix de dates de tirs :*

$$\mathsf{Movf}_c((m,\theta),t_i) = \left\{ \theta^c \in \mathbb{R}_{\geq 0}^{T_c} \;\middle|\; \begin{array}{l} \theta_k^c \in I_s(t_k) \text{ si } t_k \in \mathsf{newen}(m,t_i) \\ \theta_k^c = \theta_k - \theta_i \text{ si } t_k \in \mathsf{pers}(m,t_i) \\ \theta_k^c = \perp \text{ sinon} \end{array} \right\}$$

On obtient $\mathsf{Movf}_u((m,\theta),t_i)$ *en changeant toutes les occurrences de* θ^c *par* θ^u *et* $\mathbb{R}_{\geq 0}^{T_c}$ *par* $\mathbb{R}_{\geq 0}^{T_u}$ *dans la précédente définition.*

- *enfin,* $\mathsf{Trans} : S \times T \times T \times \mathbb{R}_{\geq 0}^{T_c} \times \mathbb{R}_{\geq 0}^{T_u} \to S$ *combine tous ces choix des joueurs et donne l'état qui en résulte :*

$$\mathsf{Trans}(s,t_c,t_u,\theta^c,\theta^u) = ((m \setminus \mathsf{Pre}(t_u)) \cup \mathsf{Post}(t_u), \theta^c \cup \theta^u)$$

si $t_c \in \mathsf{Movt}_c(s)$, $t_u \in \mathsf{Movt}_u(s,t_c)$, $\theta(t_c) = \theta(t_u) = \min_k(\theta(t_k))$, $t_u \in T_u \vee t_u = t_c$, $\theta^c \in \mathsf{Movf}_c(s,t_u)$ *et* $\theta^u \in \mathsf{Movf}_u(s,t_u)$.

L'union $\theta^u \cup \theta^c$ *est disjointe et dans* $\mathbb{R}_{\geq 0}^{T}$ *car* $\mathbb{R}_{\geq 0}^{T_u}$ *et* $\mathbb{R}_{\geq 0}^{T_c}$ *sont disjoints et* $\mathbb{R}_{\geq 0}^{T_u} \cup \mathbb{R}_{\geq 0}^{T_c} \subseteq \mathbb{R}_{\geq 0}^{T}$.

Un jeu d'atteignabilité $\mathcal{R} = (\mathcal{A}, \mathsf{Goal})$ *est composé d'une arène et d'un ensemble* $\mathsf{Goal} \subseteq S$ *d'états cibles. L'objectif de* Pl_c*, le contrôleur, est d'atteindre un état de* Goal *et l'objectif de* Pl_u*, l'environnement, est d'éviter ces états.*

Définition 7. *Une* exécution *dans une arène est un mot fini ou infini* $s_0 s_1 ... s_n$ *sur l'alphabet* S *tel que*

$$s_0 \xrightarrow{t_1} s_1 \xrightarrow{t_2} s_2 .... \xrightarrow{t_n} s_n$$

avec $\forall i, \exists \theta_i^u, \theta_i^c, t_{ci}, t_{ui}.\mathsf{Trans}(s_i, t_{ci}, t_{ui}, \theta_i^c, \theta_i^u) = s_{i+1}$ *et* $t_{ci} \in \mathsf{Movt}_c(s_i)$, $t_{ui} \in \mathsf{Movt}_u(s_i, t_{ci})$, $\theta_i^u \in \mathsf{Movf}_u(s_i, t_{ui})$ *et* $\theta_i^c \in \mathsf{Movf}_c(s_i, t_{ui})$.

Définition 8. *Une* stratégie *pour l'environnement* Pl_u *(resp. le contrôleur* Pl_c*) est une fonction* $\sigma_u : S \times T_c \to T \times \mathbb{R}_{\geq 0}^{T_u}$ *(resp.* $\sigma_c : S \to T_c \times \mathbb{R}_{\geq 0}^{T_c}$*).*[1]

Définition 9. *Une exécution* $s_0 s_1 ...$ est conforme à *une stratégie* σ_c *(resp.* σ_u*) si à chaque position* i *(sauf la dernière dans le cas d'une exécution finie) :* $\exists t_{ci}, t_{ui}, \theta_i^c, \theta_i^u$ *t. q.* $\sigma_c(s_i) = (t_{ci}, \theta_i^c)$ *(resp.* $\sigma_u(s_i, t_{ci}) = (t_{ui}, \theta_i^u)$*) et* $Trans(s_i, t_{ci}, t_{ui}, \theta_i^c, \theta_i^u) = s_{i+1}$.

Définition 10. *Une* exécution maximale *est une exécution qui est soit infinie, soit finie et telle que de son dernier état, aucun marquage n'est accessible.*

Définition 11. *Une exécution maximale est* gagnante *pour le contrôleur s'il existe une position* n *telle que* $s_n \in \mathsf{Goal}$*. Sinon, l'exécution est gagnante pour l'environnement.*

Une stratégie est gagnante *pour un joueur si et seulement si toutes les exécutions conformes à cette stratégie sont gagnantes pour ce joueur.*

Remarque 6. *Ce jeu est un* jeu concurrent déterminé à deux joueurs *et on aura toujours un unique gagnant entre* Pl_c *et* Pl_u.

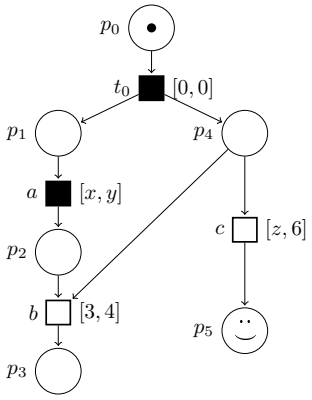
3 Calculer les états gagnants

La construction d'une stratégie pour le contrôleur est basée sur le PSCG $\mathcal{G}$. Pour construire une telle stratégie gagnante, on va utiliser un procédé en arrière pour calculer récursivement les prédécesseurs contrôlables des états cibles, et ceci jusqu'à atteindre un point fixe.

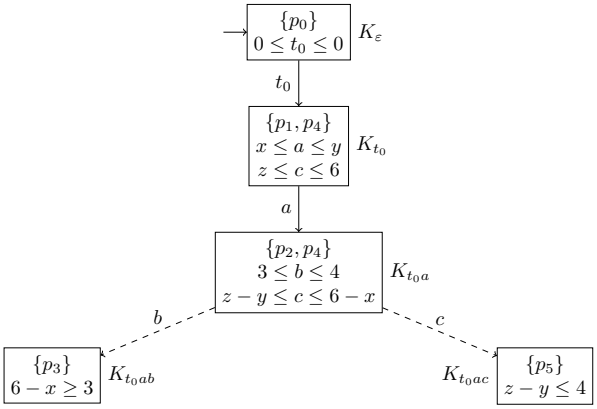
Un PTPN $\mathcal{N}$ et son graphe de classes d'états $\mathcal{G}$ sont présentés en figures 1a et 1b. On utilise des carrés noirs pour représenter les transitions contrôlables et des blancs pour les transitions

[1] Habituellement, les stratégies sont définies avec la trace entière en mémoire, mais nous verrons en sous-section 3.3 que par construction seule une stratégie sans mémoire est nécessaire.

incontrôlables. Dans $\mathcal{G}$, des flèches en pointillés sont utilisées pour les transitions incontrôlables. Les états ciblés sont ceux avec un jeton dans la place p_5. Pour atteindre un tel état depuis les états de K_{t_0a}, on doit tirer la transition c avant b. C'est la condition que l'on va chercher à propager durant le procédé en arrière sur cet exemple.



(1a) Réseau de Petri temporel paramétré $\mathcal{N}$



(1b) Graphe de classes d'états paramétrées $\mathcal{G}$

3.1 Ensembles de prédécesseurs et opérateurs sur les valuations

Définition 12. *Soit $C \xrightarrow{t_f} C'$ une transition dans $\mathcal{G}$ et B un sous-ensemble d'une classe C'.*

On définit l'ensemble des prédécesseurs $\mathsf{Pred}_{C \xrightarrow{t_f} C'}(B)$ *de $B \subseteq C'$ dans C par la transition t_f:* $\mathsf{Pred}_{C \xrightarrow{t_f} C'}(B) = \{s \in C \mid \exists s'. s \xrightarrow{t_f} s' \in B\}$.

Nous définissons deux ensembles $\mathsf{cPred}_{C \xrightarrow{t_f} C'}(B)$ et $\mathsf{uPred}_{C \xrightarrow{t_f} C'}(B)$ pour les *prédécesseurs contrôlables* et *incontrôlables* d'un sous-ensemble B de C' dans une classe C et par le tir de la transition t_f. $\mathsf{cPred}_{C \xrightarrow{t_f} C'}(B)$ correspond aux états à partir desquels le contrôleur peut forcer à atteindre B et $\mathsf{uPred}_{C \xrightarrow{t_f} C'}(B)$ à ceux depuis lesquels le contrôleur ne peut pas éviter B.

Définition 13. *On suppose sans perte de généralité que* $\{t_1, ..., t_n\} = \mathsf{newen_c}(C, t_f)$ *et* $\{t_{n+1}, ..., t_{n+k}\} = \mathsf{newen_u}(C, t_f)$. *On définit :*

$$\mathsf{cPred}_{C \xrightarrow{t_f} C'}(B) = \left\{ (m, \theta, v_p) \in C \;\middle|\; \begin{array}{ll} \forall t_i \in \mathsf{newen_c}(C, t_f), \exists \theta'_i \in I_s(t_i) \text{ t. q.} & (1) \\ \forall t_{n+j} \in \mathsf{newen_u}(C, t_f), \forall \theta'_{n+j} \in I_s(t_{n+j}), & (2) \\ s \xrightarrow{t_f} s' = (m', \theta', v_p) \in B & \\ \text{où } \forall i \in [\![1, n+k]\!], \theta'(t_i) = \theta'_i & \\ \quad \text{et } \forall i \in [\![1, l]\!], \theta'(t_{n+k+i}) = \theta(t_{n+k+i}) - \theta(t_f) & \end{array} \right\}$$

La définition de $\mathsf{uPred}_{C \xrightarrow{t_f} C'}(B)$ *est obtenue en remplaçant les lignes (1) et (2) par :* $\forall t_i \in \mathsf{newen_c}(C, t_f), \forall \theta'_i \in I_s(t_i)$ *t. q.* $\forall t_{n+j} \in \mathsf{newen_u}(C, t_f), \exists \theta'_{n+j} \in I_s(t_{n+j})$

Afin de calculer symboliquement ces ensembles d'états, nous aurons besoin de quelques opérateurs sur les ensembles de valuations. Dans la suite, D et D' sont des ensembles de valuations et nous désignons les valuations par les séquences de leurs valeurs non $\bot$.

On définit d'abord la classique *projection existentielle* :

Définition 14. *Quel que soit l'ensemble de valuations D t. q. $\forall \theta \in D, \mathsf{tr}(\theta) = \{t_1, \ldots, t_{n+k}\}$,*

$$\pi^{\exists}_{\{t_1,\ldots,t_n\}}(D) = \{(\theta_1 ... \theta_n, v_p) \mid \exists \theta_{n+1}, ..., \theta_{n+k}, (\theta_1 ... \theta_{n+k}, v_p) \in D\}$$

Nous définissons aussi une *projection universelle* moins usuelle de D' à l'intérieur de D :

Définition 15. *Quels que soient deux ensembles de valuations D et D' tels que $D' \subseteq D$ et $\forall \theta \in D, \mathsf{tr}(\theta) = \{t_1, \ldots, t_{n+k}\}$,*

$$\pi^{\forall}_{\{t_1,\ldots,t_n\}}(D, D') = \left\{ (\theta_1 ... \theta_n, v_p) \middle| \begin{array}{l} \exists \theta_{n+1}, ..., \theta_{n+k}, (\theta_1 ... \theta_{n+k}, v_p) \in D \\ \wedge \forall \theta_{n+1}, ..., \theta_{n+k}, (\theta_1 ... \theta_{n+k}, v_p) \in D \\ \qquad \Longrightarrow \ (\theta_1 ... \theta_{n+k}, v_p) \in D' \end{array} \right\}$$

Nous avons aussi besoin d'une *opération d'extension* :

Définition 16. *Pour tout ensemble de valuations D t. q. $\forall \theta \in D, \mathsf{tr}(\theta) = \{t_1, \ldots, t_n\}$,*

$$\pi^{-1}_{\{t_1,\ldots,t_{n+k}\}}(D) = \{(\theta_1 ... \theta_{n+k}, v_p) \mid (\theta_1 ... \theta_n, v_p) \in D \text{ et } \forall i, \theta_{n+i} \geq 0\}$$

Enfin, nous définissons un opérateur de *retour dans le temps* :

Définition 17. *Pour tout ensemble de valuations D t. q. $\forall \theta \in D, \mathsf{tr}(\theta) = \{t_1, \ldots, t_n\}$ et pour $t_f \neq t_i$ pour tout $i \in [\![1, n]\!]$,*

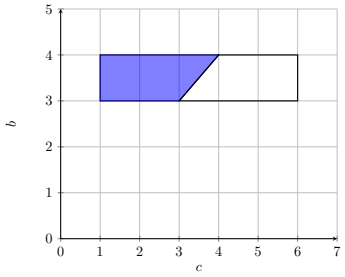
$$D + t_f = \{(\theta'_1 \ldots \theta'_n \theta'_f, v_p) \mid (\theta_1 \ldots \theta_n, v_p) \in D, \theta'_f \geq 0 \text{ et } \forall i, \theta'_i = \theta_i + \theta'_f\}$$

Remarque 7. *La projection universelle est paramétrée par deux ensembles de valuations contrairement à la projection existentielle, car nous voulons uniquement des états qui, après extension, sont corrects du point de vue de la sémantique du TPN. Et puisque dans notre construction, nous utiliserons cette projection avec des transitions nouvellement sensibilisées, nous pouvons aisément justifier ce choix, car choisir ces dates de tir en dehors de leur intervalle statique n'est pas pertinent. Il est donc naturel de restreindre les projections à des valuations qui peuvent être étendues en des états corrects atteignables, à savoir des états qui font partie d'une classe du PSCG. Notez que l'opération d'extension nous donne ce type de valuations non pertinentes, de sorte qu'il faut les intersecter avec le domaine d'une classe d'états du graphe au préalable.*

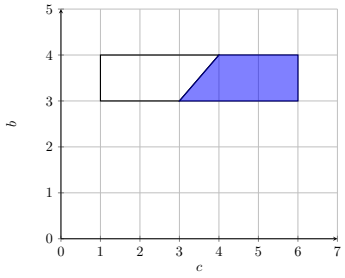
La projection universelle est exprimable en utilisant uniquement des compléments d'ensembles et des projections existentielles : $\forall \tau \subseteq T, \pi^{\forall}_{\tau}(D, D') = \pi^{\exists}_{\tau}(D) \cap \overline{\pi^{\exists}_{\tau}(\overline{D'} \cap D)}$.

Exemple 1. *Une illustration graphique de l'intuition derrière la projection universelle est donnée en figure 2 en deux dimensions. On suppose pour des questions graphiques que $z - y = 1$ et on prend D' une partie du domaine de la classe d'états K_σ pour le tir de la séquence $\sigma = t_0.a$, qui permet de mettre un jeton en p_5. Les autres choix de paramètres causeront un décalage de l'arête gauche du rectangle sur l'axe horizontal.*

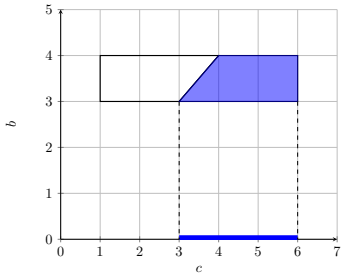
L'importance de la sélection des paramètres lors de la synthèse de contrôleur est illustrée en figure 3 où la projection $\pi^{\forall}_{\{c\}}(K_{t_0 a}, D')$ donne $1 \leq c < 3$ en figure 3a mais un ensemble vide pour les deux autres choix de paramètres en figures 3b et 3c.



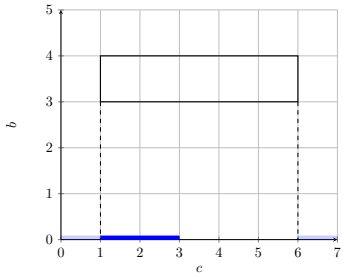
(a) D' coloré dans le rectangle $K_{t_0 a}$



(b) $K_{t_0 a} \cap \overline{D'}$

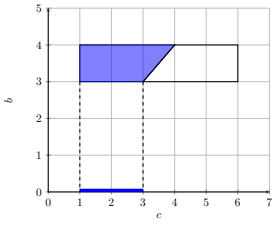


(c) $\pi^{\exists}_{\{c\}}(K_{t_0 a} \cap \overline{D'})$

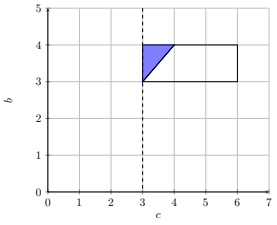


(d) $\pi^{\exists}_{\{c\}}(K_{t_0 a}) \cap \overline{\pi^{\exists}_{\{c\}}(K_{t_0 a} \cap \overline{D'})} = \pi^{\forall}_{\{c\}}(K_{t_0 a}, D')$

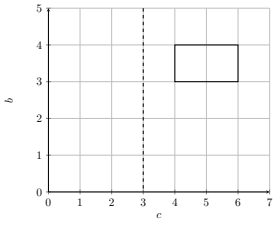
Figure 2: Exemple de projection universelle avec $z - y = 1$



(a) $z - y = 1$



(b) $z - y = 3$



(c) $z - y = 4$

Figure 3: L'influence de différents choix de paramètres sur la projection universelle

3.2 Calcul symbolique des prédécesseurs

Nous avons maintenant toutes les opérations sur les ensembles de valuations nécessaires pour calculer symboliquement les prédécesseurs d'un sous-ensemble d'une classe. Dans ce qui suit, on supposera toujours que $B \subseteq C'$ et en particulier, on aura $\mathsf{en}(C') = \mathsf{en}(B)$.

Les opérateurs de projection sont les éléments de base pour les opérateurs prédécesseurs.

Proposition 1 (Calcul de $\mathsf{cPred}()$ et $\mathsf{uPred}()$)**.** *Soit* $C = (m, D)$, $C' = (m', D')$ *et* $B = (m', D'') \subseteq C'$. *Notons* $\mathsf{cPred}_{C \xrightarrow{t_f} C'}(B) = (m, D^c_p)$ *et* $\mathsf{uPred}_{C \xrightarrow{t_f} C'}(B) = (m, D^u_p)$.

$$D^c_p = D \cap \pi^{-1}_{\mathsf{en}(C)}\left(\pi^{\exists}_{\mathsf{pers}(C,t_f)}\left(\pi^{\forall}_{\substack{\mathsf{newen_c}(C,t_f) \\ \cup \mathsf{pers}(C,t_f)}}(D', D'')\right) + t_f\right)$$

$$D_p^u = D \cap \pi^{-1}_{\mathsf{en}(C)}\Big(\pi^{\forall}_{\mathsf{pers}(C,t_f)}\big(\pi^{\exists}_{\substack{\mathsf{newen_u}(C,t_f)\\ \cup\mathsf{pers}(C,t_f)}}(D'), \pi^{\exists}_{\substack{\mathsf{newen_u}(C,t_f)\\ \cup\mathsf{pers}(C,t_f)}}(D'')\big) + t_f\Big)$$

Les formules données pour $\mathsf{cPred}()$ et $\mathsf{uPred}()$ peuvent être implémentées avec des opérations sur les polyèdres convexes. En effet, projection existentielle et intersection sont des opérations classiques. La projection universelle nécessite un complément qui produit une union finie de polyèdres convexes. L'opération d'extension consiste juste à redimensionner le polyèdre et initialiser les nouvelles variables de façon non contrainte. Toutes ces opérations s'étendent directement aux unions finies de polyèdres convexes.

3.3 États gagnants

L'objectif de cette partie est de définir l'ensemble Win des états gagnants pour le contrôleur. Nous définirons d'abord inductivement Win_n pour des stratégies en moins de n étapes et nous montrerons ensuite qu'il admet un point fixe qui correspond à l'ensemble des états gagnants.

Définition 18. *Nous définissons d'abord les ensembles d'états suivants, dont nous aurons besoin pour construire* Win_{k+1} *à partir de* Win_k *:*

$$\mathsf{uGood}_k(C) = \bigcup_{\substack{(C \xrightarrow{t_f} C') \in \mathcal{G},\\ t_f \in \mathsf{en_u}(C)}} \Big(\mathsf{cPred}_{C \xrightarrow{t_f} C'}(\mathsf{Win}_k \cap C')\Big)$$

$$\mathsf{uBad}_k(C) = \bigcup_{\substack{(C \xrightarrow{t_f} C') \in \mathcal{G},\\ t_f \in \mathsf{en_u}(C)}} \Big(\mathsf{uPred}_{C \xrightarrow{t_f} C'}(\overline{\mathsf{Win}_k} \cap C')\Big)$$

On définit de même $\mathsf{cGood}_k(C)$ *et* $\mathsf{cBad}_k(C)$ *en remplaçant* $t_f \in \mathsf{en_u}(C)$ *par* $t_f \in \mathsf{en_c}(C)$*.*

Intuitivement, un état est dans $\mathsf{cGood}_k(C)$ s'il existe une transition contrôlable pouvant être tirée et, en arrivant dans C', on peut choisir des dates de tirs pour les transitions contrôlables nouvellement sensibilisées de sorte que, quoi que choisisse l'environnement comme dates de tirs pour ses transitions incontrôlables, on arrive dans Win_k. L'ensemble $\mathsf{uGood}_k(C)$ est le même, sauf que la transition tirée est incontrôlable. À l'inverse, un état est dans $\mathsf{uBad}_k(C)$ s'il existe une transition incontrôlable qui peut être tirée et pour laquelle, en arrivant dans C', quel que soit le choix de dates de tirs des transitions contrôlables, nous ne pouvons pas assurer d'atteindre Win_k. L'ensemble $\mathsf{cBad}_k(C)$ est identique, sauf que la transition tirée est contrôlable.

À partir de ces ensembles, on définit inductivement l'ensemble Win_n correspondant exactement aux états depuis lesquels le contrôleur a une stratégie gagnante en au plus n étapes :

$$\mathsf{Win}_0 = \mathsf{Goal} \quad \text{et} \quad \mathsf{Win}_{k+1} = \mathsf{Win}_k \cup \bigcup_{C \in \mathcal{G}} \Big(\Big[\big(\mathsf{uGood}_k(C) \setminus \mathsf{cBad}_k(C)\big) \cup \mathsf{cGood}_k(C)\Big] \setminus \mathsf{uBad}_k(C)\Big)$$

Lemme 1. *Pour tout état* s *de* $\mathcal{N}$*,* $s \in \mathsf{Win}_n$ *si et seulement si depuis* s *le contrôleur a une stratégie pour atteindre* Goal *en au plus* n *étapes.*

La preuve est omise pour des raisons d'espace mais le cas non paramétré est dans [17].

Nous obtenons l'ensemble des états gagnants pour le contrôleur : $\mathsf{Win} = \mathsf{Win}_n$ pour le plus petit n tel que l'on a atteint un point fixe dans la construction de Win_n (c.-à-d. $\mathsf{Win}_n = \mathsf{Win}_{n+1}$). Ce point fixe existe toujours quand le graphe $\mathcal{G}$ est fini, c.-à-d. si $\mathcal{N}$ est borné. Le contrôleur

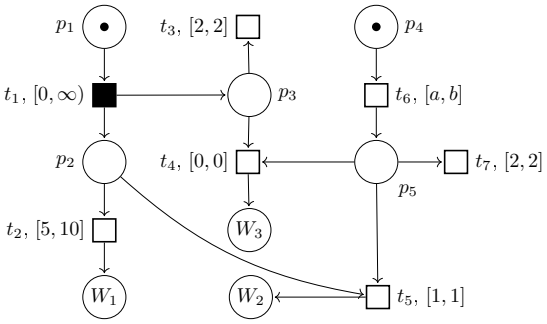
a une stratégie gagnante si et seulement si un des états initiaux est dans Win. Une stratégie consiste à choisir une des valuations de paramètres gagnantes, et ensuite d'opter pour des tirs et dates de tirs permettant de rester dans Win. Ainsi, si l'état courant s est dans Win_{i+1} pour $i \geq 0$, le contrôleur choisira un successeur de s qui est dans $\mathsf{Win}_i \backslash \mathsf{Win}_{i+1}$, afin d'éviter les boucles infinies. Tant que $s \notin \mathsf{Goal}$, ceci est toujours possible par construction de Win. Pour rendre ce choix déterministe, on supposera que les états sont ordonnées, p. ex. par ordre lexicographique, et que le contrôleur choisit le plus petit d'abord. Les successeurs par une transition contrôlable t_c auront la priorité, car le contrôleur doit proposer une transition en début de tour. Ensuite, la nouvelle partie de valuation θ^c sera choisie selon la transition t_u choisie par l'environnement. L'état courant est la seule information utilisée pour faire ce choix. Aucune information sur les tours précédents n'est nécessaire, la stratégie ne fait pas appel à l'utilisation de mémoire.

4 Étude de cas

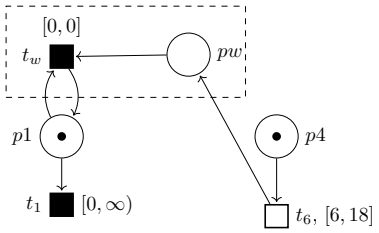
Par souci de lisibilité, nous omettons la transition d'initialisation immédiate et plaçons le réseau directement dans le marquage initial.

Nous considérons le modèle de la Figure 4 de deux lignes de production commençant respectivement en p_1 et p_4, associées à une cellule de tri et d'assemblage. Les deux lignes commencent par amener les produits en $p_2 + p_3$ et en p_5, respectivement avec les transitions t_1 et t_6. Les produits de p_5 sont soit déchargés par la transition t_7, soit assemblés avec les produits de p_2 ou p_3. Les produits de p_5 assemblés avec les produits de p_3 sont déchargés par la transition t_4. Les produits de p_3 qui ne sont pas assemblés avec p_5 sont fournis par la transition t_3 à une autre ligne W_3. Les produits de p_5 assemblés avec les produits de p_2 sont fournis par la transition t_5 à une autre ligne W_2. Les produits de p_2 qui ne sont pas assemblés avec p_5 sont fournis à une autre ligne W_1 par la transition t_2. La transition t_1 est la seule transition contrôlable.

Nous souhaitons synthétiser un contrôleur qui garantira aux produits d'atteindre les places W_1, W_2 ou W_3, selon le cas considéré à l'aide de l'implémentation dans l'outil Roméo [18].



4.a Le modèle réseau de Petri



4.b Réinitialisation de la date de tir de t_1 quand t_6 est tirée

Figure 4: Lignes de productions

Synthèse de la stratégie sans paramètre Nous fixons d'abord la valeur des paramètres temporels de la transition t_6 à 12 : $a = b = 12$.

On cherche un contrôleur pour atteindre l'un des états objectifs W_1, W_2 et W_3. On obtient trois stratégies gagnantes, qui consistent à initialiser la date de tir de t_1 dans l'état initial.

Les résultats sont les suivants : i) Si l'objectif est W_1, initialiser t_1 tel que : $\theta_1 \in [0,3)$ ou $\theta_1 \in (10,+\infty)$; ii) Si l'objectif est W_2, initialiser t_1 tel que : $\theta_1 \in (0,3)$; iii) Si l'objectif est W_3, initialiser t_1 tel que : $\theta_1 \in (10,12)$ ou $\theta_1 \in (12,14)$.

Synthèse des paramètres et de la stratégie Nous cherchons maintenant à synthétiser les contraintes sur les paramètres a et b pour obtenir une stratégie gagnante pour l'objectif W_3.

Nous obtenons alors $b-a<4$ et $a \leq b$ et la stratégie gagnante est « Initialiser t_1 tel que : $0 \leq a \leq b \leq \theta_1 \leq a+2$ ou $\theta_1 < a+2$ et $\theta_1 \in (b-2,b]$ ».

Implémentation de la stratégie Si nous fixons les paramètres a et b respectivement à 6 et 18, il n'existe pas de stratégie pour atteindre W_3 avec notre méthode et sémantique, car le choix de la date de t_1 doit être réévalué en fonction de ce que l'environnement fait. En effet, nous recherchons des contrôleurs qui peuvent être mis en œuvre avec des méthodes classiques de temps réel : des interruptions déclenchées par des timers. Nous devons donc spécifier explicitement quelle action de l'environnement doit amener le contrôleur à réévaluer la date de t_1. Cela peut se faire facilement avec un gadget qui permet de désactiver puis de réactiver une transition contrôlable donnée (ici t_1) lorsqu'une transition donnée de l'environnement (ici t_6) est tirée, comme le montre la Figure 4.b.

Nous obtenons alors une stratégie gagnante pour atteindre W_3 comme suit :

« Dans l'état initial, initialiser t_1 tel que : $t_1 \in (16,+\infty)$.

Après le tir de t_6 (et donc de t_w), initialiser t_1 tel que : $t_1 \in [0,2)$. »

5 Conclusion

Nous avons défini un nouveau type de jeu temporisé d'accessibilité à deux joueurs sur le graphe de classes d'états de réseaux de Petri temporels paramétrés. Cela permet de synthétiser un contrôleur qui choisit, dès qu'une nouvelle transition est sensibilisée, la date à laquelle cette transition sera tirée. L'intérêt de ce type de contrôleur est qu'il peut être implémenté dans un contexte temps réel avec des interruptions déclenchées par des timers dont les durées sont fixées dès que les actions associées sont planifiées.

Dans nos travaux futurs, nous étudierons comment cette sémantique s'adapte au problème de l'observation partielle. Nous nous efforcerons également de calculer efficacement le PSCG. De plus, nous prévoyons d'étudier le problème de la synthèse des contrôleurs pour la sûreté et les propriétés ω-régulières.

References

[1] R. Alur and D. Dill. A Theory of Timed Automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[2] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In *ACM Symposium on Theory of Computing*, pages 592–601, 1993.

[3] Francesco Basile, Roberto Cordone, and Luigi Piroddi. Supervisory control of timed discrete-event systems with logical and temporal specifications. *IEEE Transactions on Automatic Control*, 67(6):2800–2815, 2022.

[4] Gerd Behrmann, Agnès Cougnard, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. Uppaal-tiga: Time for playing games! In *19th International Conference on Computer Aided Verification (CAV 2007)*, volume 4590 of *LNCS*, pages 121–125, Berlin, Germany, 2007. Springer.

[5] B. Bérard, F. Cassez, S. Haddad, Didier Lime, and O. H. Roux. Comparison of different semantics for time petri nets. In Doron A. Peled and Yih-Kuen Tsay, editors, *Automated Technology for Verification and Analysis*, pages 293–307, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[6] B. Berthomieu and M. Diaz. Modeling and verification of time dependent systems using time Petri nets. *IEEE trans. on soft. eng.*, 17(3):259–273, 1991.

[7] Bernard Berthomieu, Silvano Dal Zilio, Łukasz Fronc, and François Vernadat. Time petri nets with dynamic firing dates: Semantics and applications. In Axel Legay and Marius Bozga, editors, *Formal Modeling and Analysis of Timed Systems*, pages 85–99, Cham, 2014. Springer.

[8] Bernard Berthomieu and Miguel Menasche. An enumerative approach for analyzing time petri nets. In *Proceedings IFIP*, pages 41–46. Elsevier Science Publishers, 1983.

[9] Franck Cassez, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *CONCUR'05*, volume 3653 of *LNCS*, pages 66–80, San Fransisco, CA, USA, August 2005. Springer.

[10] D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Proc. Int. Workshop Automatic Verification Methods for Finite State Systems (CAV'89)*, volume 407 of *LNCS*, pages 197–212. Springer, 1989.

[11] Guillaume Gardey, Olivier (F.) Roux, and Olivier H. Roux. Safety control synthesis for time Petri nets. In *8th International Workshop on Discrete Event Systems (WODES'06)*, pages 222–228, Ann Arbor, USA, July 2006. IEEE Computer Society Press.

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

[13] Parisa Heidari and Hanifa Boucheneb. Maximally permissive controller synthesis for time petri nets. *International Journal of Control*, 86, 03 2013.

[14] András Horváth, Marco Paolieri, Lorenzo Ridi, and Enrico Vicario. Transient analysis of non-markovian models using stochastic state classes. *Performance Evaluation*, 69(7):315–335, 2012. Selected papers from QEST 2010.

[15] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. A Game Approach to the Parametric Control of Real-time Systems. *International Journal of Control*, 92(9):2025–2036, 2019.

[16] Aleksandra Jovanović, Didier Lime, and Olivier H. Roux. Control of real-time systems with integer parameters. *IEEE Transactions on Automatic Control*, 67(1):75–88, 2022.

[17] Loriane Leclercq, Didier Lime, and Olivier H. Roux. A state class based controller synthesis approach for Time Petri Nets. In *The 44th International Conference on Application and Theory of Petri Nets and Concurrency (Petri Nets 2023)*, LNCS, Lisbon, Portugal, June 2023. Springer.

[18] Didier Lime, Olivier H. Roux, Charlotte Seidner, and Louis-Marie Traonouez. Romeo: A parametric model-checker for Petri nets with stopwatches. In Stefan Kowalewski and Anna Philippou, editors, *15th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2009)*, volume 5505 of *LNCS*, pages 54–57, York, United Kingdom, March 2009. Springer.

[19] Oded Maler, Amir Pnueli, and Joseph Sifakis. On the synthesis of discrete controllers for timed systems. In *STACS'95*, volume 900 of *LNCS*, pages 229–242. Springer, 1995.

[20] P. J. Ramadge and W. M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal on Control and Optimization*, 25(1):206–230, 1987.

[21] Wolfgang Thomas. On the synthesis of strategies in infinite games. In *Proc. 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95)*, volume 900, pages 1–13. Springer, 1995.

[22] Louis-Marie Traonouez, Didier Lime, and Olivier H. Roux. Parametric model-checking of time Petri nets with stopwatches using the state-class graph. In *6th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS 2008)*, volume 5215 of *LNCS*, pages 280–294, Saint-Malo, France, September 2008. Springer.