Codage de CDEVS et de PDEVS en réseau de Petri temporisé

Vincent Albert, Sangeeth Saagar Ponnusamy

Encoding CDEVS and PDEVS into Timed Petri Net

Codage de CDEVS et de PDEVS en réseau de Petri temporisé

V. Albert [1,2] S. S. Ponnusamy [1,2]
[1] CNRS, LAAS, 7 avenue du Colonel Roche, F-31400 Toulouse, France
[2] Université de Toulouse, LAAS, F-31400 Toulouse, France

vincent.albert@laas.fr
sangeeth.saagar.ponnusamy@laas.fr

Résumé :

Cet article présente un codage des sémantiques CDEVS (Classic Discrete-EVent Specification) et PDEVS (Parallel Discrete-EVent Specification) en TPN (réseaux de Petri temporisés - Timed PetriNet) avec priorités et gestion des données. Ce codage constitue une spécification formelle de la sémantique d'exécution des modèles CDEVS et PDEVS pour notre outil de simulation ProDEVS [1]. A partir d'un modèle à base de réseau de Petri résultant d'une transformation automatique d'un modèle DEVS saisi dans ProDEVS, nous exécutons une exploration exhaustive du modèle. Nous montrons comment faire une vérification formelle de propriétés de simulation liées à la correction du modèle : est-il bien construit et légitime pour toutes ses éventuelles utilisations ? Sera-t-il correctement exécuté par le simulateur.

Mots-clés :

Réseau de Petri temporisé, CDEVS, PDEVS, V&V, Méthodes formelles.

Abstract:

This paper presents an encoding of CDEVS (Classic Discrete-EVent Specification) and PDEVS (Parallel Discrete-EVent Specification) semantics in TPN (Timed Petri Nets) with priorities and data handling. This encoding is a formal specification of the execution semantics for our simulation tool. From a Timed Petri Net based model resulting from an automatic transformation of a DEVS model designed in ProDEVS, we perform an exhaustive exploration of the model. We show how to check formal verification simulation properties related to the correctness of the model : is it well built and legitimate for all its potential uses ? Will be properly executed by the simulator.

Keywords:

Timed Petri Net, CDEVS, PDEVS, V&V, Formal Methods.

1 Introduction

Les réseaux de Petri (TPN) sont un formalisme de modélisation très utilisés pour l'analyse et l'ordonnancement des systèmes multi-taches et la conception de contrôleurs de chaînes de production ou de systèmes automatisés. Ils permettent de très bien décrire le parallélisme et la concurrence (partage de ressources, synchronisation) entre des taches ou des processus. Les réseaux de Petri sont constitués de places et de transitions reliées par des arcs et des jetons qui circulent dans les places à travers les transitions. Une place dans un réseau de Petri peut décrire une condition, une activité, une attente, une ressource. Les transitions représentent des événements, le passage d'une activité à une autre, la prise ou la libération d'une ressource. Ils sont formels puisqu'ils reposent sur une définition mathématique. En l'occurrence il suffit des deux matrices que sont `Pre` et `Post` (matrice d'incidence amont et matrice d'incidence aval respectivement) pour définir un réseau de Petri. De ces matrices nous pouvons tirer des graphes ou des arbres puis faire des analyses sur ces graphes basées sur des invariants linéaires ou semi-flots pour vérifier des propriétés de conservation, de vivacité/blocage, et d'atteignabilité.

Contrairement à la simulation, l'exécution qui consiste à construire un graphe ou un arbre à partir d'un modèle à base de réseau de Petri ne nécessite pas de scénario d'entrée. Elle va étudier les traces pour toutes les entrées possibles et tous les chemins éventuels de la simulation modulo la précision des calculs de l'ordinateur. De cette différence fondamentale entre les méthodes d'exploration exhaustive et la simulation, nous souhaitons

transférer les résultats issus des méthodes formelles en général et des réseaux de Petri en particulier vers la dynamique d'une simulation DEVS afin de mieux appréhender nos modèles et nos simulations qui en outre disposeront d'une spécification formelle de leur sémantique d'exécution.

D'autres travaux [2] et [5] s'intéressent au couplage de la simulation DEVS (Discrete-EVent Specification) et des méthodes formelles. Dans [2], les auteurs ont donné des exemples de propriétés non vérifiables tantôt par les méthodes formelles, tantôt par simulation. Ils ont défini un codage de DEVS pour Promela. Dans [5], les auteurs utilisent un codage de DEVS en automates temporisés et d'une transformation vers Uppaal pour vérifier leurs modèles avant implantation sur cible matérielle ou logicielle. Il faut également citer [6] qui a défini une sémantique opérationnelle pour CDEVS (Classic DEVS).

Dans la section suivante nous présentons les réseaux de Petri. Dans la section 3 nous donnons le codage de DEVS en un réseau de Petri temporisé TPN pour un composant atomique et le codage des sémantiques CDEVS (Classic DEVS) et PDEVS (Parallel DEVS) en un réseau de Petri temporisé (PrTPN) avec priorité. Dans la section 4 nous appliquons ces codages à un exemple, nous analysons leur graphe d'état et nous montrons comment utiliser les logiques temporelles pour vérifier des propriétés d'absence de famine et de légitimité sur la dynamique de la simulation. Nous conclurons et présenterons les perspectives de ce travail dans la section 5.

2 Réseau de Petri temporisé

Les définitions suivantes sont tirées de [4].

Un réseau de Petri temporisé avec priorité (PrTPN) est un tuple $\langle P, T, Pre, Post, \succ, m_0, I_s \rangle$ dans lequel :

— $P, T, Pre, Post, m_0$ est un réseau de Petri. P est l'ensemble des places, T est l'ensemble des transitions, m_0 est le marquage initial et $Pre, Post : T \times P \rightarrow \mathbb{R}^+_{0,\infty}$ sont les matrices d'incidences amont et aval respectivement.

— $I_s : T \rightarrow I^+$ est une fonction d'intervalle statique avec I^+ l'ensemble non vide des intervalles de réels tel que ses bornes sont des rationnels non négatifs.

— $\succ$ est une relation de priorité, supposée antiréflexive, asymétrique et transitive. Les priorités sont représentées par des arcs orientés entre deux transitions, la transition source étant la plus prioritaire.

Un état du réseau de Petri est une paire $s = (m, I)$ dans laquelle m est un marquage et I est une fonction appelée la fonction intervalle. La fonction $I : T \rightarrow I^+$ est une fonction qui associe à chaque transition sensibilisée au marquage m un intervalle temporel.

Le graphe d'état d'un PrTPN est le système de transition temporisé $SG = \langle S, s_0, \rightsquigarrow \rangle$. La sémantique d'un PrTPN $\langle P, T, Pre, Post, \succ, m_0, I_s \rangle$ est le système de transition temporisé $\langle S, s_0, \rightsquigarrow \rangle$ où :

— S est l'ensemble des états (m, I) du PrTPN

— $s_0 = (m_0, I_0)$ est l'état initial, où m_0 est le marquage initial et I_0 est la fonction d'intervalle statique I_s restreinte aux transitions sensibilisées à m_0.

— $\rightsquigarrow \subseteq S \times T \cup \mathbb{R}^+ \times S$ est la transition d'état, définie par $((s, a, s') \in \rightsquigarrow$ et notée $(s \overset{a}{\rightsquigarrow} s')$.

— nous avons $(m, I) \overset{t}{\rightsquigarrow} (m', I')$ ssi $t \in T$ et :

1. $m \geq Pre(t)$, t est sensibilisée à partir d'un état m
2. $0 \in I(t)$, t est tirable instantanément
3. $(\forall t' \in T)$ alors $(m \geq Pre(t')$ et $(t' \succ t) \Rightarrow 0 \notin I(t'))$, il n'y a pas de transition avec une plus forte priorité qui satisfait les conditions 1 et 2
4. $(\forall k \in T)(m' \geq Pre(k) \Rightarrow I'(k) =$

si $k \neq t \wedge m - Pre(t) \geq Pre(k)$ alors $I(k)$ sinon $I_s(k)$). Après le tir de t alors $m' = m - Pre(t) + Post(t)$, les transitions qui restent sensibilisées (t exclue) conservent leur intervalle d'avant le tir, les autres sont associées à leur intervalle statique

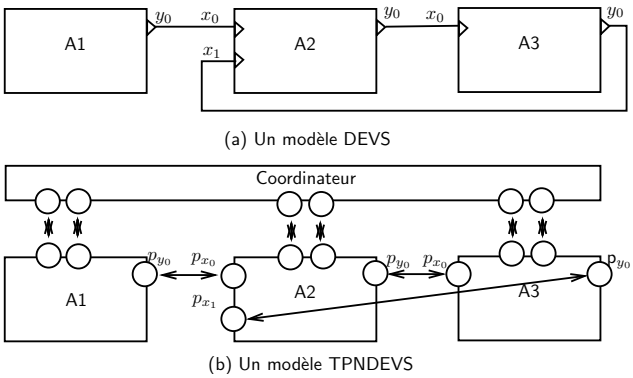
— nous avons $(m, I) \overset{\theta}{\rightsquigarrow} (m', I')$ ssi $\theta \in \mathbb{R}^+$ et :

5. $(\forall k \in T)(m \geq Pre(k) \Rightarrow \theta \leq \uparrow I(k))$, une transition temporelle θ est possible si θ n'est pas plus grand que la borne gauche de l'intervalle de chaque transition sensibilisée.
6. $(\forall k \in T)(m \geq Pre(k) \Rightarrow I'(k) = I(k) - \theta$, θ est retiré de l'intervalle de toutes les transitions qui étaient sensibilisées avant le tir de la transition temporelle.

Toute transition sensibilisée doit être tirée dans l'intervalle de temps qui lui est associé. Dans le codage ci-après, tous les intervalles sont sous la forme $[\theta; \theta]$, dit ponctuels. De plus nous utilisons différents type d'arcs. L'arc classique noté $p \rightarrow t$ avec $p \in P$ et $t \in T$ donne $Pre(t, p) = 1$ ou $t \rightarrow p$ avec $p \in P$ et $t \in T$ donne $Post(t, p) = 1$. L'arc inhibiteur, noté $p \multimap t$ désensibilise t si il y a un jeton dans p. L'arc reset noté $p \overset{*}{\rightarrow} t$ vide le contenu de p lors du tir de t et n'est pas bloquant, i.e. $Pre(t, p) = 0$. L'arc de lecture noté $p \bullet\!\!- t$ est bloquant, i.e. $Pre(t, p) = 1$, mais ne modifie pas le marquage de p après le tir de t.

3 Codage de DEVS

Un modèle DEVS est une composition de N composants atomiques qui communiquent par des messages. Un TPNDEVS est un ensemble de $N + 1$ réseaux de Petri qui partagent des places communes. Pour chaque composant atomique du modèle DEVS nous avons un réseau de Petri avec des places qui correspondent aux ports d'entrée/sortie du composant atomique. Une connexion d'un port de sortie vers un port d'entrée dans un modèle couplé résulte en une fusion des places correspondantes. Un réseau de Petri supplémentaire, appelé le coordinateur, permet de coder la sémantique choisie, classique ou parallèle. Les composants atomiques et le coordinateur communiquent par des places. La figure 1 illustre la structure du codage d'un modèle DEVS. Les flèches représentent une fusion de place.



(a) Un modèle DEVS

(b) Un modèle TPNDEVS

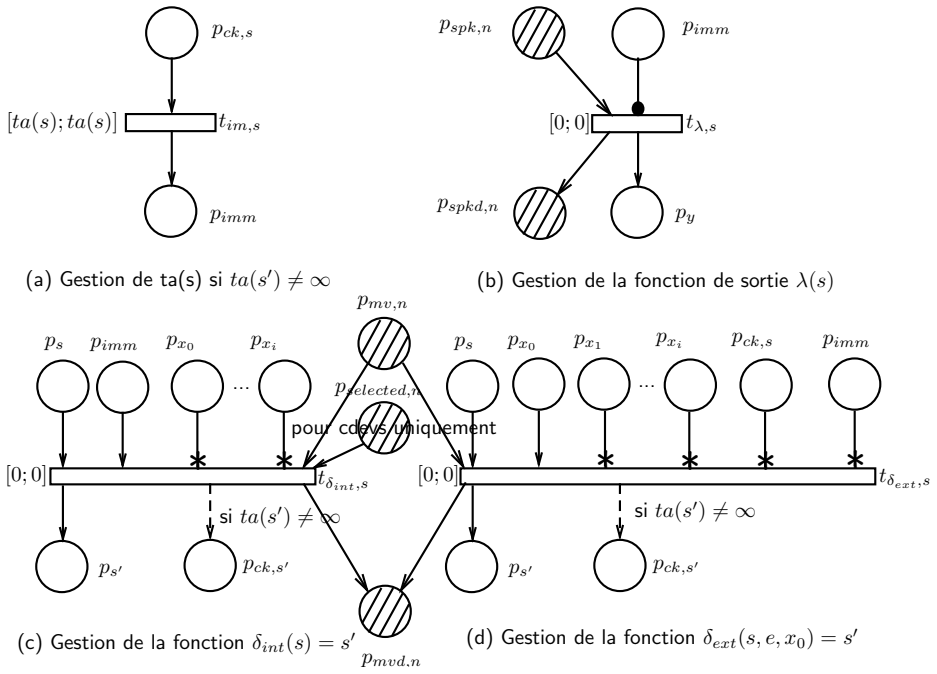
Figure 1 – Structure du codage en TPNDEVS

Pour la suite du papier nous considérons que chaque composant peut recevoir un événement externe à tout moment. Nous considérons les communications asynchrones, c'est à dire non bloquantes avec possibilité de perte de message. Si lors de la réception d'un message sur le port d'entrée x le composant est dans un état s qui écoute sur x i.e. $\delta_{ext}(s, e, x) \in \delta_{ext}$, alors le message sera traité, sinon il sera perdu. De plus nous considérons les communications de type un-vers-plusieurs. Une entrée ne peut être associée qu'à un connecteur et une sortie peut être associée à plusieurs connecteurs. Et nous considérons qu'il n'y a pas de sortie d'un composant connectée avec une entrée du même composant.

3.1 Codage du composant atomique

Un composant atomique est codé à partir de 4 blocs élémentaires pour la gestion des horloges, des fonctions de sorties, de transitions internes et externes.

La représentation graphique de ces 4 blocs est donnée sur la figure 2. Pour un composant



(a) Gestion de ta(s) si $ta(s') \neq \infty$

(b) Gestion de la fonction de sortie $\lambda(s)$

(c) Gestion de la fonction $\delta_{int}(s) = s'$

(d) Gestion de la fonction $\delta_{ext}(s, e, x_0) = s'$

Figure 2 – Règles de codage du composant atomique

atomique $n = \langle X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta \rangle$ il existe un TPN équivalent $\langle P_n, T_n, Pre, Post, m_0, I_s \rangle$ tel que :

1. pour chaque état $s \in S$ nous avons une place $p_s \in P_n$
2. pour chaque port d'entrée $x \in X$ nous avons une place $p_x \in P_n$
3. pour chaque port de sortie $y \in Y$ nous avons une place $p_y \in P_n$
4. nous avons les places $p_{spk,n}, p_{spkd,n}, p_{mv,n}, p_{mvd,n} \in P_n$ (et $p_{selected,n}$ pour CDEVS) qui servent de communications avec le coordinateur
5. nous avons la place $p_{imm} \in P_n$, qui lorsqu'elle est marquée, indique que le composant est imminent
6. pour chaque état $s \in S$ tel que $ta(s) \neq \infty$ nous avons la place $p_{ck,s} \in P_n$ et la transition $t_{im,s} \in T_n$, $p_{ck,s}$ est une place qui permet de déclencher l'horloge pour $ta(s)$. Nous avons $p_{ck,s} \rightarrow t_{im,s}$, $t_{im,s} \rightarrow p_{imm}$ et $Is(t_{im,s}) = [ta(s); ta(s)]$
7. pour chaque fonction de sortie $\lambda(s) \in \lambda$ nous avons une transition $t_{\lambda,s} \in T_n$ tel que $Is(t_{\lambda,s}) = [0;0]$, i.e. le tir est immédiat, $p_{imm} \multimap t_{\lambda,s}$ et $t_{\lambda,s} \rightarrow p_y$, le marquage de $p_y \in P$ indique la présence d'une donnée dans le port de sortie $y \in Y$ et $t_{\lambda,s}$ indique le tir de la fonction de sortie à partir de s
8. pour chaque fonction de transition interne $\delta_{int}(s) \in \delta_{int}$, nous avons une transition $t_{\delta_{int},s} \in T_n$ tel que $Is(t_{\delta_{int},s}) = [0;0]$ et $p_{imm} \rightarrow t_{\delta_{int},s}$. Le marquage de p_s indique que l'état courant est s. Pour une fonction de transition interne $\delta_{int}(s) = s'$, nous avons $p_s \rightarrow t_{\delta_{int},s}$ et $t_{\delta_{int},s} \rightarrow p_{s'}$, et pour tout les ports d'entrée $x \in X$, nous avons $p_x \overset{*}{\rightarrow} t_{\delta_{int},s}$. Enfin si $ta(s') \neq \infty$ nous avons $t_{\delta_{int},s} \rightarrow p_{ck,s'}$
9. pour chaque fonction de transition externe $\delta_{ext}(s, e, x) \in \delta_{ext}$, nous avons une transition $t_{\delta_{ext},s} \in T_n$ tel que $Is(t_{\delta_{ext},s}) = [0;0]$. Pour une fonction de transition externe $\delta_{ext}(s, e, x) = s'$, nous avons $p_s \rightarrow t_{\delta_{ext},s}$, $t_{\delta_{ext},s} \rightarrow p_{s'}$ et $p_x \rightarrow t_{\delta_{ext},s}$, et pour tout les ports d'entrée $x' \in X$ (excluant x),nous avons $p_{x'} \overset{*}{\rightarrow} t_{\delta_{ext},s}$. Enfin nous avons $p_{imm} \overset{*}{\rightarrow} t_{\delta_{ext},s}$, $p_{ck,s} \overset{*}{\rightarrow} t_{\delta_{ext},s}$ et si $ta(s') \neq \infty$ nous avons $t_{\delta_{ext},s} \rightarrow p_{ck,s'}$.

Soit un composant n avec comme état initial $s \in S$, nous avons un jeton dans $p_{ck,s}$ et un jeton dans p_s. La seule condition pour tirer $t_{im,s}$ est le marquage de $p_{ck,s}$. Si $p_{ck,s}$ reste marquée pendant $ta(s)$ car aucun événement externe n'est survenu, alors $t_{im,s}$ est tirée ce qui entraîne le marquage de p_{imm} qui signifie que le composant est imminent. Le composant déclenche alors la fonction de sortie par le tir de $t_{\lambda,s}$ qui met un jeton dans le place du port de sortie p_y. Puis il déclenche la fonction de transition interne par le tir de $t_{\delta_{int},s}$, qui aura pour effet de vider les places des ports d'entrée et d'aller vers l'état s' indiqué par le marquage de $p_{s'}$. Si un événement est survenu sur l'entrée $x \in X$ avant l'expiration de ta(s), indiqué par le marquage de la place du port d'entrée p_x, toutes les conditions sont réunies pour déclencher la fonction de transition externe par le tir de $t_{\delta_{ext},s}$ qui va vider les places de tous les ports d'entrée du composant et la place de l'horloge de l'état s, $p_{ck,s}$. Le composant se retrouve dans un nouvel état s' indiqué par le marquage de $p_{s'}$ et le cycle recommence. Si un composant est imminent, $m(p_{imm}) = 1$, et qu'il reçoit une entrée sur x, $m(p_x) = 1$ alors $t_{\delta_{int},s}$ et $t_{\delta_{ext},s}$ sont en conflit que l'on peut résoudre en ajoutant une priorité entre ces transitions.

3.2 Codage du coordinateur

L'exécution d'un TPNDEVS est un jeu qui se déroule en deux temps. Dans ce jeu, les joueurs sont les composants atomiques. Dans un premier temps ils peuvent parler en exécutant une fonction de sortie et dans un deuxième temps ils peuvent bouger en exécutant une fonction de transition interne ou externe. Ils peuvent aussi ne rien faire. Pour le premier temps, la sémantique PDEVS est codée par une structure parallèle des réseaux de Petri et la sémantique CDEVS par une alternative. Le codage du deuxième temps est le même pour PDEVS et CDEVS.

Un coordinateur parallèle est un PrTPN $\langle P, T, Pre, Post, \succ, m_0, I_s \rangle$ tel que :

1. nous avons une place $p_{1immN} \in P$ qui indique le nombre de composants imminents à un cycle de simulation et nous avons les transitions $t_{spk}, t_{mv} \in T$ qui lorsqu'elles sont franchies indiquent qu'il est temps de parler et de bouger respectivement
2. pour chaque composant atomique $n \in N$ nous avons :
   — les places $p_{spk,n}, p_{spkd,n}, p_{mv,n}, p_{mvd,n} \in P$ et les transitions $t_{spkpass,n}, t_{mvpass,n} \in T$. $p_{spk,n}$ et $p_{mv,n}$ indiquent au composant n qu'il peut parler et bouger, respectivement. $p_{spkd,n}$ et $p_{mvd,n}$ indiquent que n a parlé et bougé respectivement. $t_{spkpass,n}$ et $t_{mvpass,n}$ sont franchies si n passe son tour dans l'étape parler et l'étape bouger, respectivement
   — $p_{imm,n} \multimap t_{spkpass,n}$, un composant ne peut passer son tour que s'il n'est pas imminent. Et pour chaque $t_{im,s} \in T_n$ nous avons $t_{im,s} \succ t_{spkpass,n}$
   — pour chaque $t_{\lambda,s} \in T_n$, nous avons $p_{spk,n} \rightarrow t_{\lambda,s}$, $t_{\lambda,s} \rightarrow p_{spkd,n}$ et $t_{\lambda,s} \rightarrow p_{1immN}$
   — pour chaque $t_{\delta_{int}} \in T_n$, nous avons $p_{mv,n} \rightarrow t_{\delta_{int}}$, $t_{\delta_{int}} \rightarrow p_{mvd,n}$, $t_{\delta_{int}} \succ t_{mvpass,n}$. Même chose pour chaque $t_{\delta_{ext}} \in T_n$
   — pour chaque $p_x \in P$ nous avons $p_x \overset{*}{\rightarrow} t_{mvpass,n}$.

La représentation graphique d'un coordinateur pour PDEVS est donnée sur la figure 3.

p_{1immN} nous permet de mettre le réseau de Petri dans un état de blocage si aucun des composants n'est imminent. Elle indique qu'il faut au moins un composant imminent parmi les N pour continuer le jeu. C'est la seule place du TPNDEVS qui est N-bornée. Toutes les autres sont 1-bornées.

À l'état initial nous avons un jeton dans $p_{spk,1}, ..., p_{spk,n}$ qui indique aux joueurs qu'ils peuvent parler. $p_{spk,n}$ est consommé soit par le tir d'une fonction de sortie soit par $t_{spkpass,n}$ si $p_{imm,n}$ n'est pas marquée. Chaque composant a l'opportunité de déclencher une fonction de sor-

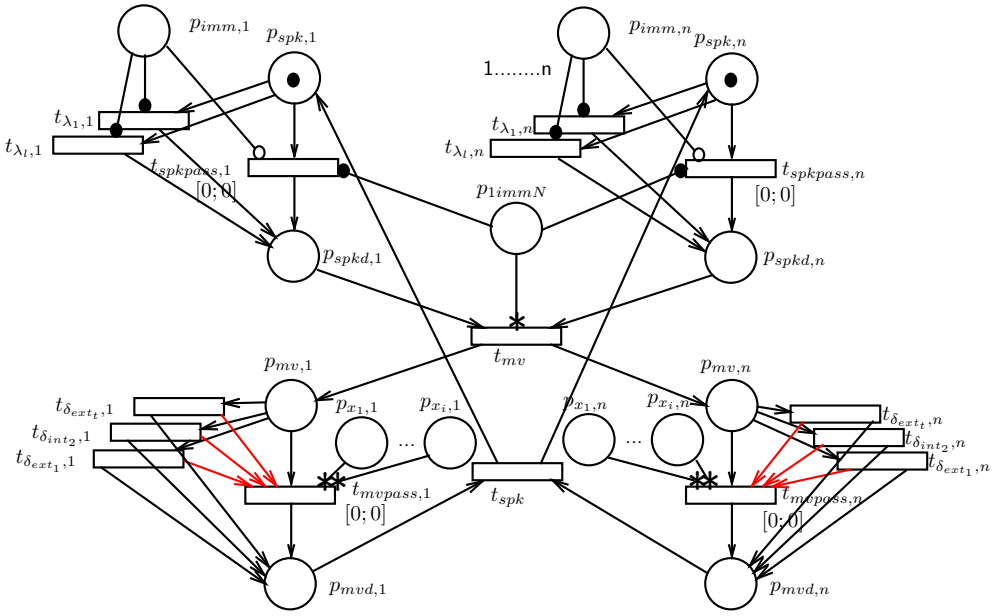


Figure 3 – Coordinateur de la sémantique PDEVS

tie si il est imminent, sinon il passe son tour. À la fin du tour, t_{mv} est franchissable. Les composants peuvent alors bouger par le marquage de $p_{mv,1}, ..., p_{mv,n}$. Un composant n a la possibilité de tirer une transition externe ou interne ou sinon de ne rien faire en passant par $t_{mvpass,n}$. Notons que les places d'entrées sont vidées par le franchissement de $t_{mvpass,n}$. En effet il est possible qu'un composant non imminent ait reçu des entrées qui ne sont pas écoutées dans l'état courant du composant, ce que nous considérons comme passer son tour. Lorsque tout le monde a bougé, le coordinateur revient à l'état initial pour un nouveau cycle.

Pour CDEVS, si plusieurs composants sont imminents, c'est le coordinateur qui sélectionne sur la base de la fonction select. La représentation graphique d'un coordinateur pour CDEVS est donnée figure 4.

Un coordinateur classique est un PrTPN $\langle P, T, Pre, Post, \succ, m_0, I_s \rangle$ tel que :

1. nous avons les places $p_c, p_{spkd} \in P$, p_c est la place qui indique que nous sommes au début d'un cycle et p_{spkd} indique que le composant imminent sélectionné a parlé. Nous avons les transitions t_{spk}, t_{mv} qui lorsqu'elles sont franchies indiquent qu'il est temps de parler et bouger respectivement.
2. pour chaque composant atomique $n \in N$, nous avons
   - les places $p_{spk,n}, p_{selected,n}, p_{mv,n}, p_{mvd,n} \in P$ et les transitions $t_{select,n}, t_{mvpass,n} \in T$. $p_{spk,n}$ et $p_{mv,n}$ indiquent au composant n qu'il peut parler et bouger respectivement. $p_{mvd,n}$ indiquent que le composant a bougé. $p_{selected,n}$ sert de pré-condition au tir d'une transition interne. $t_{select,n}$ est franchie si un composant imminent est sélectionné.
   - $p_{imm,n} \multimap t_{select,n}$, seul les composants imminents peuvent être sélectionnés. Pour chaque couple $(t_{im,s}, t_{select,n}) \in T_n \times T$, nous avons $t_{im,s} \succ t_{select,n}$.
   - pour chaque $t_{\lambda,s} \in T_n$, nous avons $p_{spk,n} \rightarrow t_{\lambda,s}, t_{\lambda,s} \rightarrow p_{spkd}$
   - pour chaque $t_{\delta_{int}}, \in T_n$, nous avons $p_{selected,n} \rightarrow t_{\delta_{int}}$
   - pour chaque $t_{\delta_{int}} \in T_n$, nous avons $p_{mv,n} \rightarrow t_{\delta_{int}}$, $t_{\delta_{int}} \rightarrow p_{mvd,n}$, $t_{\delta_{int}} \succ t_{mvpass,n}$. Même chose pour chaque $t_{\delta_{ext}} \in T_n$
   - pour chaque $p_x \in P$ nous avons $p_x \overset{*}{\rightarrow} t_{mvpass,n}$.

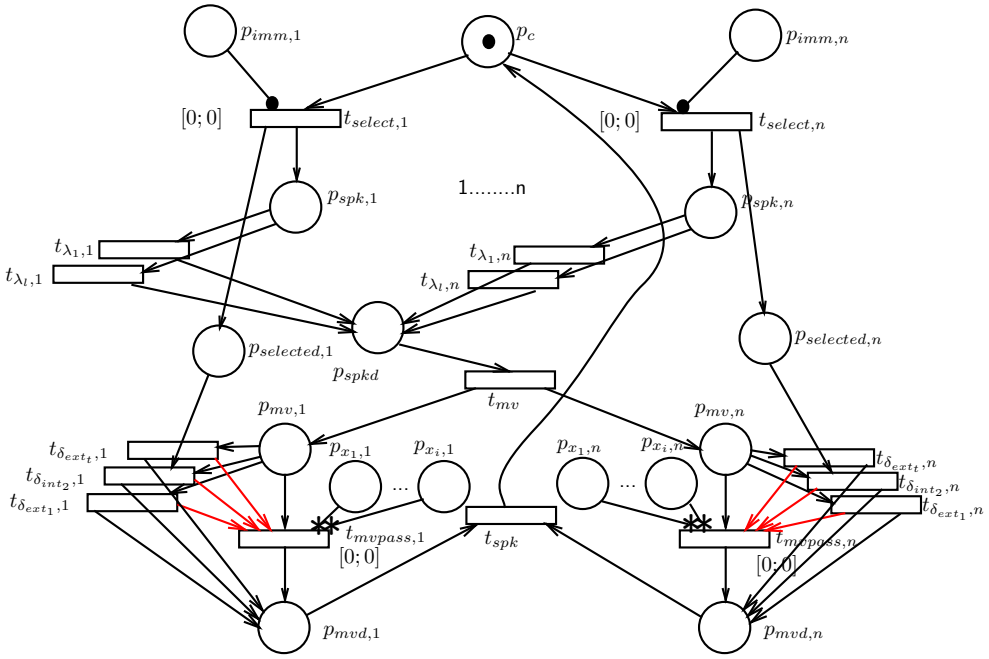


Figure 4 – Coordinateur de la sémantique CDEVS

4 Exploration exhaustive et analyse

Nous appliquons le codage ci-dessus au modèle DEVS composé de trois composants atomiques générateur (G), buffer (B) et processeur (P) figure 5. Cet exemple est tiré de [3]. G génère un événement sur la sortie *job* tout les θ_g. Lorsque un événement sur l'entrée *job* de B est reçu, celui-ci incrémente une variable entière positive q et, si P est libre, B génère un événement sur la sortie *req*, sinon il attend l'événement sur l'entrée *done* qui indique que P a fini de traiter la tache. Lorsque P reçoit un événement sur l'entrée *req* il passe dans un état occupé pour θ_p. Lorsque ce temps est écoulé il génère un événement sur la sortie *done* pour informer B qu'il a fini de traiter la tache. B écoute sur job dans tout état.

Le graphe d'état du TPNDEVS qui code le modèle GBP avec $t_g = 2, t_p = 3$ pour CDEVS est donnée figure 6. Nous utilisons des techniques de réduction pour abstraire le graphe d'état obtenu par exploration exhaustive en un graphe d'état bisimilaire qui considère uniquement l'état du TPN entre deux pas de simulation. Par exemple à $t = 0$ le modèle est à l'état

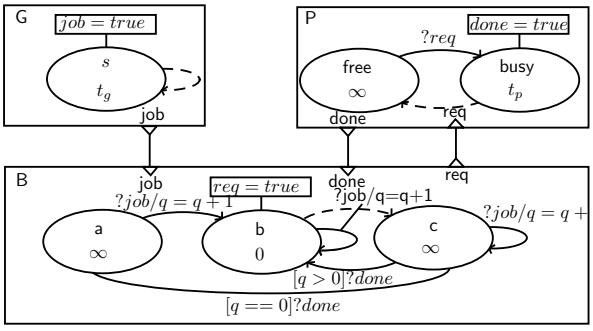


Figure 5 – Composant couplé GBP

0 qui représente l'état du modèle avant l'envoie de job par G et l'état 1 représente l'état du modèle après la transition interne de G et la transition externe de B. Nous pouvons constater que l'exécution est linéaire (pas de branchements) jusqu'à t=6 où nous avons G et P qui sont imminents en même temps. La simulation utilisera la fonction `select` pour déterminer le composant à exécuter en premier. Dans le graphe d'état les deux branches sont exécutées. Ce qu'il est intéressant de constater ici, c'est que quel que soit le chemin emprunté par la simulation et quelle que soit la spécification de la fonction `select`, l'exécution mènera au même résultat puisqu'il y a convergence des branches

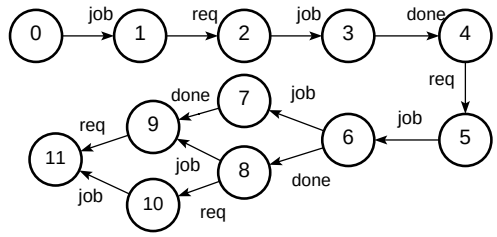


Figure 6 – Graphe d'état du modèle GBP, CDEVS, $t_g = 2, t_p = 3$

à l'état 11. Notons que l' état 11 est équivalent à l'état 2 modulo la valeur de q, qui dans ce modèle n'est pas bornée. Il n'y a pas de perte de message, tous les job sont traités par B et un req est envoyé seulement quand P est disponible.

Le graphe d'état pour PDEVS est donné figure 7. On peut constater qu'à la classe 6, suivant si B choisit job ou done l'exécution diverge. Si done est choisi, job est perdu, q ne s'incrémente pas et le modèle revient à l'état 2. Si job est choisi, done est perdu, B est pour toujours dans l'état (b), et une boucle infinie de job suivra.

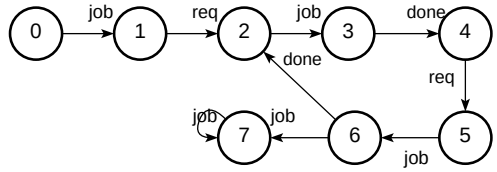


Figure 7 – Graphe d'état du modèle GBP, PDEVS, $t_g = 2, t_p = 3$

L'analyse du graphe d'état peut vite devenir impossible alors on utilise le model checking pour vérifier des spécifications sur une sémantique. Les spécifications s'écrivent sous forme de logiques temporelles. Il y a des logiques qui permettent de raisonner sur les enchaînements d'états comme LTL (Linear Temporal Logic) et d'autres sur les branchements et les chemins comme CTL (Combinatory Temporal Logic). L'utilisation des priorités dans les TPN ne préserve pas CTL mais les propriétés ci-dessous sont exprimables en LTL.

Les propriétés de non blocage sont liées au bon avancement de la simulation. Nous avons identifié deux causes de blocage de l'avancement de la simulation. La première est que tous les composants sont dans un état $s \in S$ avec $ta(s) = \infty$. Cela peut survenir par exemple si un événement est attendu pour sortir d'un état passif et que cet événement n'arrive jamais. La formule $\Box\Diamond p$ énonce que dans tous les futurs à tout moment p sera fatalement marquée. L'absence de famine est vérifiée si pour tout $n \in N$, $\Box\Diamond p_{imm,n}$.

La deuxième cause de blocage est l'illégitimité. Un modèle est illégitime si un nombre infini de transitions internes peut être tiré en temps nul. La légitimité du modèle est vérifiée pour toutes les traces possibles en ajoutant une place p_{tick} et une transition t_{tick} au TPNDEVS tel que $Is(t_{tick}) = [\theta; \theta]$ et que $\theta < ta(s)_{min}$ (le plus petit de tout le modèle) , $p_{tick} \rightarrow t_{tick}$ et $t_{tick} \rightarrow p_{tick}$. La formule $\Box\Diamond t_{tick}$ énonce que dans tous les futurs à tout moment t_{tick} sera fatalement tirée. Si elle est vraie, c'est qu'il n'y a pas d'exécution où le simulateur ne peut plus faire avancer le temps de simulation.

5 Conclusion et perspectives

Ce papier défini un codage des sémantiques CDEVS et PDEVS qui préserve LTL. Nous avons montré comment vérifier des propriétés de correction liées à la dynamique du modèle. Des propriétés de correction liées à la statique peuvent aussi être écrites comme s'assurer qu'un état s avec $ta(s) = \infty$ n'est pas associé à une fonction de transition interne ou de sortie. Ce codage peut également servir pour vérifier des propriétés de correction du codage lui-même comme par exemple s'assurer qu'un composant qui n'est pas imminent ne peut pas être sélectionné par le coordinateur. Nous souhaitons trouver un codage qui n'utilise pas les priorités pour pouvoir nous intéresser à l'organisation en arbre des exécutions dans le but de confronter le modèle à une utilisation particulière.

Références

[1] Le Hung H. Vu, D.Foures, V.Albert, *ProDEVS : An Event-driven Modeling and Simulation Tool for Hybrid Systems Using State Diagrams*, Acte de la 8th International Conference on Simulation Tools and Techniques (SIMUTOOL), Athènes, Grèce, pp. 29-37, 2015

[2] A. Yacoub, M. Hamri, C. Frydman, C. Seo, B. P. Zeigler, *Towards an Extension of PROMELA for the Modeling, Simulation and Verification of Discrete-Event Systems*, Acte du 27th European Modelling and Simulation Symposium (EMSS), pp. 340-348, 2015

[3] B. P. Zeigler, H. Praehofer, et T. G. Kim, *.Theory of Modeling and Simulation*. Academic Press, 2nd edition, 2000.

[4] B. Berthomieu, F. Peres, F. Vernadat, *Model-checking Bounded Prioriterized Time Petri Nets*, Actes du 5th Automated Technology for Verification and Analysis Symposium (ATVA), 2007.

[5] F. Madlener. *A Model of Computation for Reconfigurable Systems*, Thèse de la Technische Universitat Darmstadt, Germany, 2013

[6] E. Posse, *Modelling and simulation of dynamic structure discrete-event systems*, Thèse de la School of Computer Science McGill University, 2008.