TP COO
Conception et mise en oeuvre d'un outil de simulation à événements discrets de systèmes hybrides

1 Introduction

Les systèmes hybrides sont des systèmes qui mélangent des dynamiques continues et des dynamiques discrètes. Une dynamique continue est typiquement modélisée par un ensemble d'équations différentielles. Une dynamique discrète est une discontinuité qui change le système d'équation à résoudre. Elle est typiquement modélisée par des événements d'état ou temporels. La simulation consiste à exécuter le modèle et gérer l'avancement du temps de simulation. Une simulation à événements discrets est une simulation dans laquelle l'avancement du temps est dirigé par l'ordonnancement temporel des événements étiquetés du modèle. Le but de ce TP est de fournir une conception et une mise en oeuvre d'un simulateur à événements discrets de systèmes hybrides.

2 Le modèle

Définition du composant atomique

Un composant atomique est une structure :

$$M = \langle X, Y, S, \delta_{ext}, \delta_{int}, \lambda, ta \rangle \text{ où} \tag{1}$$

— $X = \{p, v\} \, | p \in IPorts, v \in X_p$, est l'ensemble des ports d'entrée et leurs valeurs,
— $Y = \{p, v\} \, | p \in OPorts, v \in Y_p$, est l'ensemble des ports de sortie et leurs valeurs,
— S est l'ensemble des états séquentiels et s_0 est l'état initial
— $\delta_{ext} : Q \times X \to S$, est la fonction de transition externe, où
  — $Q = \{(s, e) | s \in S, 0 \leq e \leq ta(s)\}$ est l'état total
  — e est le temps écoulé depuis la dernière transition
— $\delta_{int} : S \to S$, est la fonction de transition interne,
— $\lambda : S \to Y$, est la fonction de sortie,
— $ta : S \to \mathbb{R}^+_{0,\infty}$, est la fonction d'avancement du temps,

Un composant atomique **doit obligatoirement** fournir une implémentation
— d'une fonction de transition externe,
— d'une fonction de transition interne,
— d'une fonction de sortie,
— d'une fonction d'avancement du temps et
— d'une fonction d'initialisation pour remettre le composant dans son état initial.

De plus un composant atomique **s'engage** à retourner et à se faire modifier les 4 variables temps e, tl, tn et tr à tout moment. e est le temps écoulé dans un état depuis le dernier changement d'état du composant. tl est la date du dernier changement d'état du composant. tn est la date du prochain changement d'état du composant et tr est le temps restant avant le prochain événement du composant.

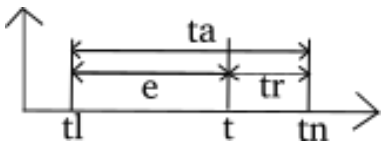

FIGURE 1 – Les 4 variables temps

3 Exemple du gen-buf-proc (10/20)

La figure ci-dessous donne une représentation graphique du modèle gen-buf-proc en utilisant le formalisme défini ci-dessus.

Comme nous pouvons le voir dans cette figure un modèle de simulation est constitué de composants atomiques (en l'occurrence 3 dans cet exemple), et de connecteurs orientés qui relient les sorties des composants aux entrées d'autres composants. Il est interdit de relier la sortie d'un composant avec l'entrée de ce même composant. Une sortie peut être reliée à plusieurs entrées mais plusieurs sorties ne peuvent être reliées à une même entrée. Les ports d'entrées et de sorties peuvent véhiculer des booléens, entiers ou des réels.

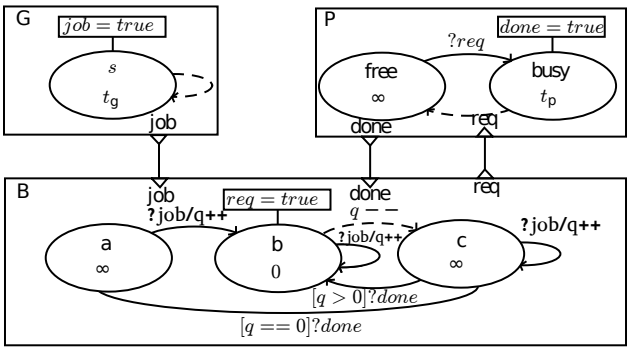

FIGURE 2 – Exemple GBP

Le tableau ci-dessous donne la trace d'exécution jusqu'à t=6 pour l'exemple du GBP. t est le temps de simulation, q est le nombre de tache contenu dans le buffer. job|2 signifie que l'événement job (en sortie du générateur) arrivera dans 2 unités de temps.

t=0	t=2	t=2	t=4	t=5	t=5	t=6
q=0	q=1	q=0	q=1	q=1	q=0	q=1
job|2	req|0	job|2	done|1	req|0	job|1	job|2
	job|2	done|3	job|2	job|1	done|3	done|2

FIGURE 3 – Trace d'execution de l'exemple GBP

4 Pseudo code partiel du simulateur à événements discrets

Le simulateur gère un ensemble de composants atomiques. Il est responsable de l'avancement du temps de simulation globale t et de l'affectation des 4 variables temps (e, tr, tl et tn) de chaque composant. Il est également responsable de l'ordonnancement des composants en terme de séquence d'appel de leurs fonctions de transition et de sorties et de la transmission des valeurs véhiculées par les ports.

L'algorithme ci-dessous donne le principe de fonctionnement de ce simulateur. Soit C l'ensemble des composants atomiques du modèle de simulation à exécuter.

Algorithm 1 Ordonnanceur

$t = 0$
$t_{fin} = 10$

for c in C
    initialiser c ($s = s_0$ et $tr = ta(s_0)$)

while($t < t_{fin}$)
    for c in C
        construire une liste de composant imminents imms ($tr = tr_{min}$)

stocker le plus petit tr dans tr_{min}

$t = t + tr_{min}$
for c in C
mise à jour de e et de tr

for c in imms
c.lambda()
for c in C
construire la liste des entrées ins impactées par les sorties produites
pour chaque sortie produite, transmettre la valeur contenue dans cette sortie aux entrées connectées
for c in C
if c in imms et c.ins = null
c.internal
mise à jour de tr, e, tl et tn du composant c
else if c not in imms et c.ins not null
c.external
mise à jour de tr, e, tl et tn du composant c
else if c in imms et c.ins not null
c.conflit
mise à jour de tr, e, tl et tn du composant c

end

5 Exemple ODE du premier ordre

Cet exemple va permettre de valider plus finement les gestions du temps de votre simulateur ainsi que le transfert de données d'une sortie vers une entrée.

5.1 Modélisation de la consigne (12/20)

Modéliser un composant atomique "step" qui génère un échelon à un temps donné. Ce composant aura trois paramètres : x_i qui est la valeur initiale de l'échelon, x_f la valeur finale de l'échelon et t_s le temps d'occurrence de l'échelon. x_i est généré en sortie au démarrage de la simulation puis x_f est généré à t_s.

Instancier 4 composants "step" avec les valeurs suivantes :
— step1 : $x_i = 1, x_f = -3, t_s = 0.65$
— step2 : $x_i = 0, x_f = 1, t_s = 0.35$
— step3 : $x_i = 0, x_f = 1, t_s = 1$
— step4 : $x_i = 0, x_f = 4, t_s = 1.5$

Modéliser un composant "adder" qui lorsqu'il reçoit une nouvelle valeur en entrée, additionne toutes les valeurs reçues en entrées jusqu'à cet instant et envoie immédiatement l'addition en sortie.

Instancier un composant additionneur avec quatre entrées et une sortie. Connecter la sortie de chaque instance du composant "step" sur une entrée du composant "adder". Envoyer la sortie de l'additionneur sur le graphe de sortie de la simulation et simuler.

5.2 Modélisation d'un intégrateur à temps discret (14/20)

Un intégrateur à temps discret est un composant qui prends une dérivée en entrée et fournit son intégrale en sortie. Lorsque le composant reçoit une nouvelle valeur de la dérivée en entrée, il l'intègre immédiatement puis l'envoie sur sa sortie. Tant que le composant ne reçoit pas de nouvelle valeur en entrée, il calcul, à chaque pas de temps h_{step}, la valeur de l'intégrale à $t + h_{step}$ en fonction de la valeur de la dérivée à t et de la valeur de l'intégrale à t et envoie cette nouvelle valeur sur la sortie.

Connecter la sortie de l'additionneur à l'entrée de votre intégrateur à temps discret. Envoyer la sortie de l'intégrateur sur le graphe de sortie de la simulation et simuler.

5.3 Modélisation d'un intégrateur à événement discret (16/20)

Un intégrateur à événement discret est un composant qui prends une dérivée en entrée et fournit son intégrale en sortie. Lorsque le composant reçoit une nouvelle valeur de la dérivée en entrée, il l'intègre immédiatement puis l'envoie sur sa sortie. Tant que le composant ne reçoit pas de nouvelle valeur en entrée, il calcul le temps qu'il faut à l'intégrale pour passer d'un état discret à un autre. Lorsque ce temps est écoulé, le composant incrémente ou décrémente, suivant le signe de la dérivée, la valeur de l'intégrale de la valeur du quantum ΔQ et envoie cette nouvelle valeur sur la sortie.

Connecter la sortie de l'additionneur à l'entrée de votre intégrateur à événement discret. Envoyer la sortie de l'intégrateur sur le graphe de sortie de la simulation et simuler.

6 Exemple ODE du second ordre (18/20)

Modéliser un composant "constante" qui envoie une valeur $x = -9.81$ sur une sortie g à $t = 0$.

Connecter la sortie du composant constante à l'entrée dv d'un intégrateur à événement discret et connecter la sortie v de ce dernier à l'entrée dh d'un autre intégrateur à événement discret. Fixer la valeur initiale de h avec $h_0 = 10$. Envoyer la sortie h de l'intégrateur sur le graphe de sortie de la simulation et simuler.

7 Exemple du Boucing ball (20/20)

Modifier l'exemple précédent afin de modéliser un événement d'état sur la valeur de h de sorte que lorsque $h <= 0$ le signe de v est inversé en y appliquant un léger coefficient de restitution. Envoyer la sortie h de l'intégrateur sur le graphe de sortie de la simulation et simuler. Fixer la valeur initiale de h avec $h_0 = 10000$ et simuler.