Accessibilité dans les systèmes à compteurs continus: théorie et pratique

Vincent Antaki

Université de Montréal

antakivi@iro.umontreal.ca

24 mai 2015

Contexte : La vérification formelle

La vérification formelle vise à confirmer qu'un système a le comportement attendu.

- Circuits combinatoires et circuits numériques
- Moteur d'inférences logiques
- Logiciels

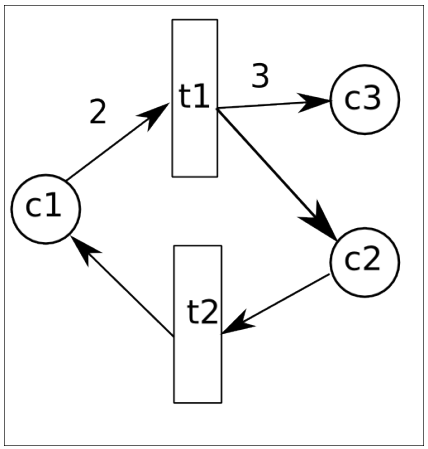
Formalise des modèles et des questions :

- Existe-t-il une séquence infinie dans cet automate ?
- Le graphe suivant possède-t-il un chemin d'un état x à un état y ?
- L'automate accepte-t-il un ensemble de mots X ?
- Une expression est-elle satisfiable dans un système de logique donné ?

Les réseaux de Petri (PN)

Un réseau de Petri $P = (P, T, Pre, Post)$ où

- P un ensemble fini de *places* (compteurs) ;
- T un ensemble fini de *transitions* ;
- Pre une matrice $|P| \times |T|$ indiquant le coût pour activer chaque transition ;
- $Post$ une matrice $|P| \times |T|$ indiquant l'ajout dans les compteurs suite à l'activation de chaque transition

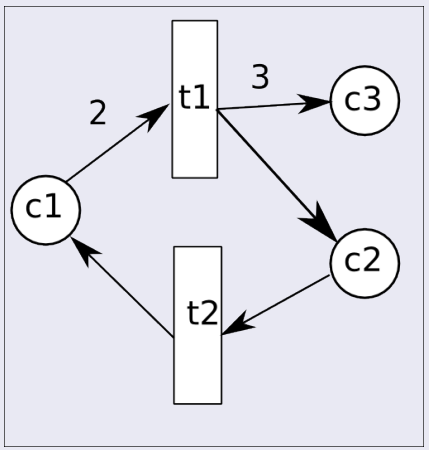


Exécution dans un PN

Exemple

Séquence d'activation : $t_1\, t_2$

Exécution : $(2,0,0)(0,1,3)(1,0,3)$



Le problème d'accessibilité

Étant donné un système S, une configuration initiale x_0 et une configuration finale x, existe-t-il une exécution $x_0 \xrightarrow{*} x$ dans S ?

Historique de la décidabilité du problème

- EXPSPACE-ardu : Lipton 1976
- Preuve partielle (erronée) : Sacerdote & Tenney 1977
- Preuve complète : Mayr 1981
- Simplifications : Kosaraju 1982, Lambert 1992
- Nouvelle preuve : Leroux 2009, 2011, 2012
- Borne supérieure (F_{ω^3}) : Leroux 2015

Les réseaux de Petri continus (CPN)

- Mêmes données qu'un PN $(P, T, Pre, Post)$
- Marquage dans les réels $m \in \mathbb{R}^P_{\geq 0}$
- Chaque transition est appliquée avec un coefficient $\alpha \in (0, 1]$
- Peut converger avec des séquences d'activation infinies
- Problème d'accessibilité P-Complet : Fraca & Haddad 2013

Le Projet

Objectif :

- Développement de module pour un outil permettant de résoudre partiellement le problème d'accessibilité dans les VASS

Résultats :

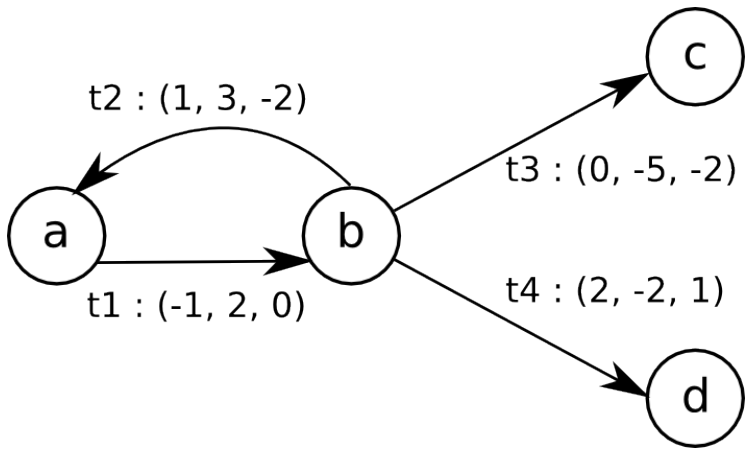
- Deux modèles définis (CVASSM et CVASSU)
- CVASSM prouvés équivalents aux CPN
- CVASSU constituent une nouvelle relaxation des VASS
- Implémentation d'un algorithme résolvant le problème d'accessibilité pour les CPN

Les systèmes d'additions de vecteurs avec états (VASS)

Un d-VASS est une paire (Q, δ) où :

- d la *dimension* (nombre de compteurs) ;
- Q un ensemble fini d'*états* ;
- $\delta \subseteq Q \times \mathbb{Z}^d \times Q$ un ensemble fini de *transitions.*

Les VASS sont équivalents aux PN.

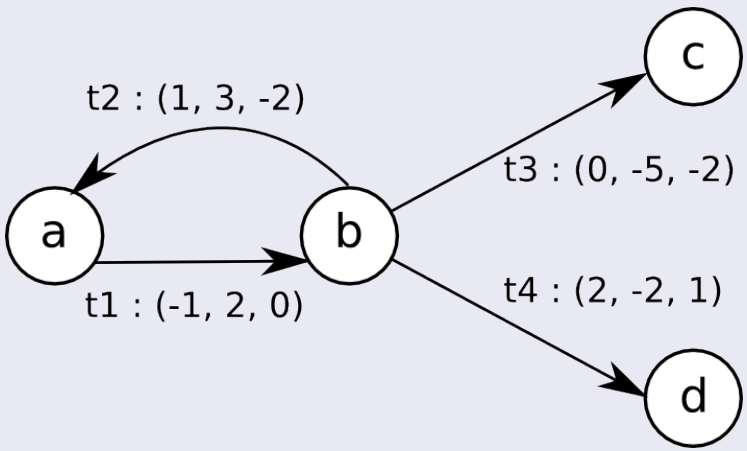


Exécution dans un VASS

Exemple

Séquence d'activation : $t_2 t_1 t_3$

Exécution : $(b,(0,0,4))(a,(1,3,2))(b,(0,5,2))(c,(0,0,0))$



Les systèmes d'addition de vecteurs continus avec états

- Même données qu'un VASS (Q, δ, d)
- Chaque transition avec un coefficient $\alpha \in (0, 1]$
- Les compteurs ont un nombre de jetons $x \in \mathbb{R}_{\geq 0}$

Avec états uniques (CVASSU) :

- Change entièrement d'état lorsqu'on emprunte une transition, indépendamment du coefficient α

Avec états multiples (CVASSM) :

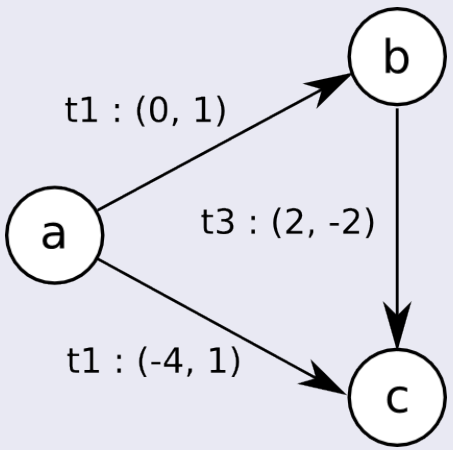
- α pondère le changement d'état ainsi que l'effet de la transition sur les compteurs

CVASS avec états multiples (CVASSM)

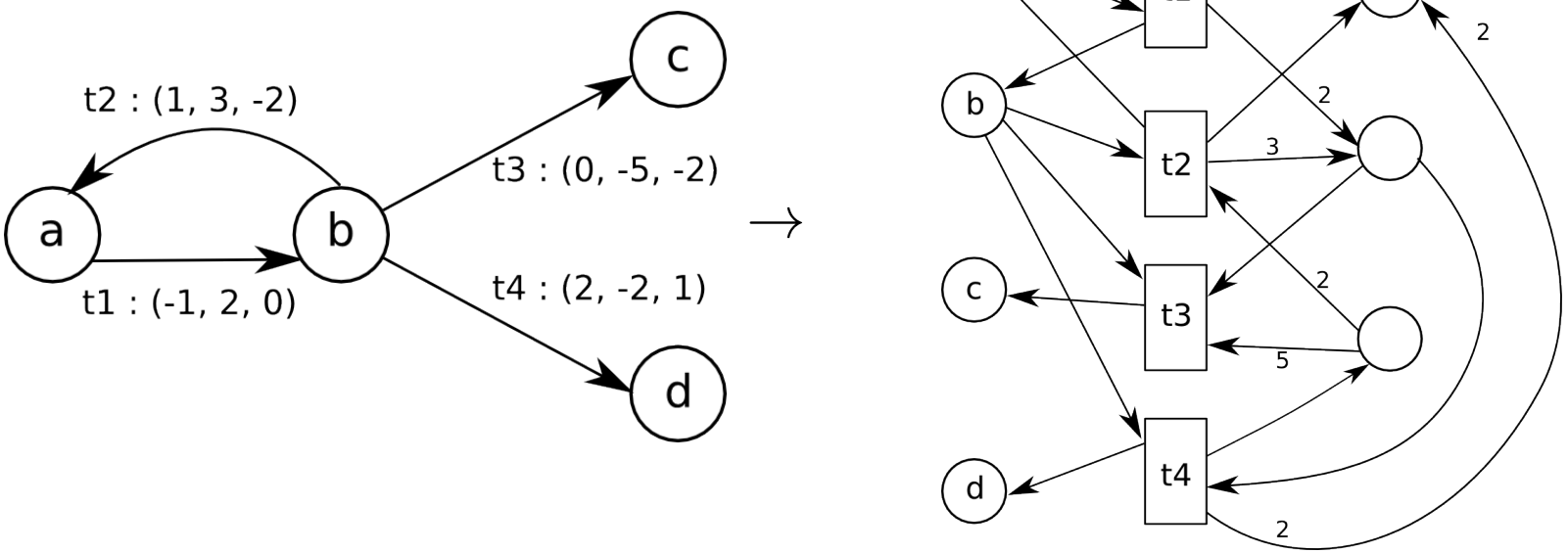
Exemple d'exécution

Séquence d'activation : $0.5t_1 0.5t_2 0.5t_3$

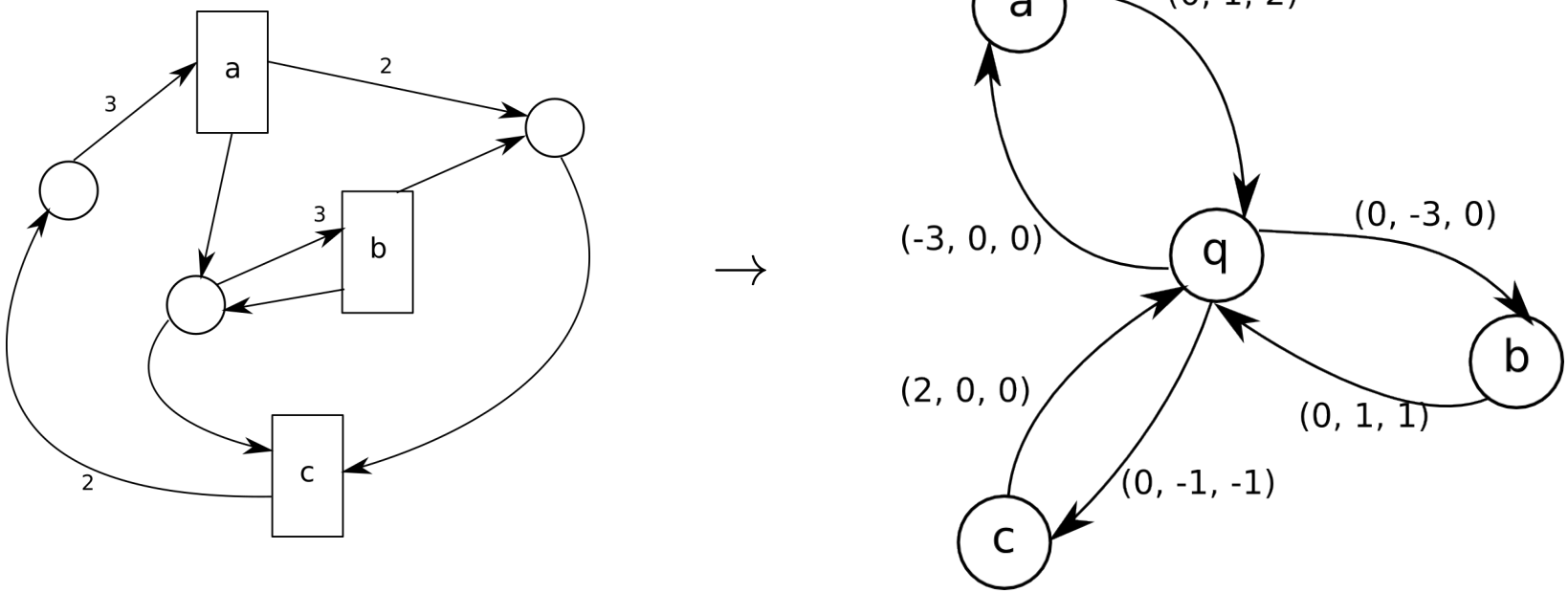
Exécution : $((1,0,0),(2,0))$ $((0.5,0.5,0),(2,0.5))$ $((0,0.5,0.5),(0,1))$ $((0,0,1),(1,0))$



Conversion CVASSM à CPN



Conversion CPN à CVASSM



Ordonnancement d'une séquence d'activation

Exemple

- On suppose que l'on commence et termine dans l'état q
- Soit une séquence désordonnée :
  $0.5\delta_{qc}\ 1.0\delta_{qb}\ 0.5\delta_{bq}\ 0.1\delta_{qb}\ 0.6\delta_{bq}\ 0.5\delta_{cq}$

1. $0.5\delta_{qc}\ 0.5\delta_{cq}\ 1.0\delta_{qb}\ 0.5\delta_{bq}\ 0.1\delta_{qb}\ 0.6\delta_{bq}$
2. $0.5\delta_{qc}\ 0.5\delta_{cq}\ 1.0\delta_{qb}\ 1.0\delta_{bq}\ 0.1\delta_{qb}0.1\delta_{bq}$

Algorithme de Fraca et Haddad

Algorithm 2: Decision algorithm for reachability

`Reachable`$(\langle \mathcal{N}, \boldsymbol{m}_0 \rangle, \boldsymbol{m})$: status
Input: a CPN system $\langle \mathcal{N}, \boldsymbol{m}_0 \rangle$, a marking $\boldsymbol{m}$
Output: the reachability status of $\boldsymbol{m}$
Output: the Parikh image of a witness in the positive case
Data: $nbsol$: integer; $\mathbf{v}, \mathbf{sol}$: vectors; T': subset of transitions

1 **if** $\boldsymbol{m} = \boldsymbol{m}_0$ **then return** (**true**,$\mathbf{0}$)
2 $T' \leftarrow T$
3 **while** $T' \neq \emptyset$ **do**
4 $\quad nbsol \leftarrow 0$; $\mathbf{sol} \leftarrow \mathbf{0}$
5 $\quad$ **for** $t \in T'$ **do**
6 $\quad\quad$ **solve** $\exists ?\mathbf{v}\ \mathbf{v} \geq \mathbf{0} \wedge \mathbf{v}[t] > 0 \wedge \boldsymbol{C}_{P \times T'} \mathbf{v} = \boldsymbol{m} - \boldsymbol{m}_0$
7 $\quad\quad$ **if** $\exists \mathbf{v}$ **then** $nbsol \leftarrow nbsol + 1$; $\mathbf{sol} \leftarrow \mathbf{sol} + \mathbf{v}$
8 $\quad$ **end**
9 $\quad$ **if** $nbsol = 0$ **then return false else sol** $\leftarrow \frac{1}{nbsol}\mathbf{sol}$
10 $\quad T' \leftarrow [\![\mathbf{sol}]\!]$
11 $\quad T' \leftarrow T' \cap \mathtt{maxFS}(\mathcal{N}_{T'}, \boldsymbol{m}_0[{}^\bullet T'^\bullet])$
12 $\quad T' \leftarrow T' \cap \mathtt{maxFS}(\mathcal{N}_{T'}^{-1}, \boldsymbol{m}[{}^\bullet T'^\bullet])$ `/* deleted for lim-reachability */`
13 $\quad$ **if** $T' = [\![\mathbf{sol}]\!]$ **then return** (**true**,**sol**)
14 **end**
15 **return false**

Système a résoudre

Résoudre $\exists ?v$ **tel que** $v \geq 0 \wedge v[t] > 0 \wedge C_{P \times T'} v = m - m_0$

Comment résoudre ce système avec un solveur qui ne traite pas les inégalités strictes ?

1. **Maximiser** t

   Sujet à $C_{P \times T'} v = m - m_0 \wedge v \geq 0$

2. **Minimiser** t

   Sujet à $C_{P \times T'} v = m - m_0 \wedge v \geq 0 \wedge v[t] \geq 1$

Solveur exact

- Les solveurs conventionnels utilisent l'arithmétique à virgule flottante, insuffisante dans le cadre de la vérification formelle.
- Solveur exact en arithmétique rationnelle plus lent
- QSopt_ex : Simplexe exact en C++

Questions ouvertes et travaux futurs

- Solveurs alternatifs et tests de performances
- Accessibilité dans les CVASSU
- Prochains modules

Merci