Formalisation des diagrammes états-transitions UML concurrents

Etienne André, Mohamed Mahdi Benmoussa, Christine Choppy

Formalisation des diagrammes états-transitions UML concurrents

Étienne André, Mohamed Mahdi Benmoussa, Christine Choppy

LIPN, UMR CNRS 7030 – Université Paris 13, Sorbonne Paris Cité
99 avenue Jean-Baptiste Clément, 93430 Villetaneuse, France
{Prénom.Nom}@lipn.univ-paris13.fr

RÉSUMÉ. *Nous présentons dans ce travail une extension de la transformation des diagrammes états-transitions vers les réseaux de Petri colorés ((André* et al.*, 2012)) au cas concurrent. Cette extension est complétée par une implémentation à l'aide de l'outil Acceleo.*

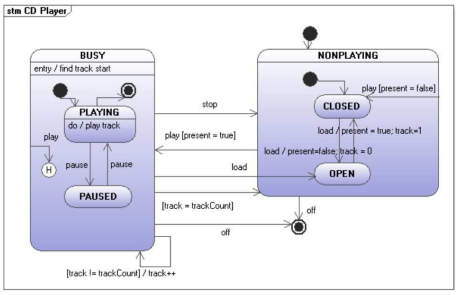
ABSTRACT. *We present in this work an extension of the transformation of state machines diagrams to colored Petri nets in the case of concurrency. This extension is completed with an implementation using Acceleo tool.*

MOTS-CLÉS : *diagrammes UML, réseaux de Petri colorés, concurrence, transformation*

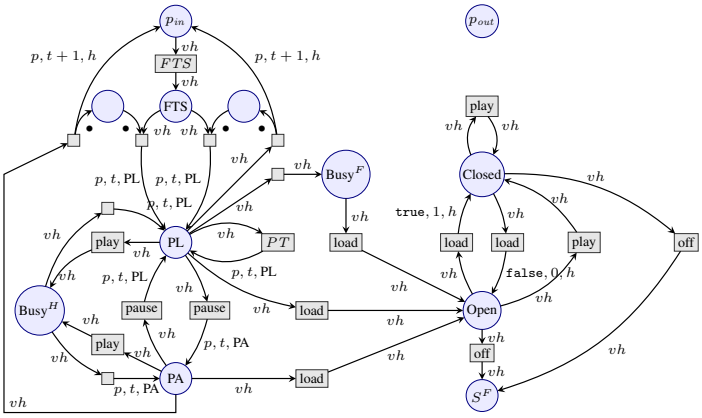
KEYWORDS: *UML diagrams, colored Petri nets, concurrency, transformation*

De nos jours les systèmes sont souvent critiques et très complexes dans leur structure ainsi que leur composition. C'est pour cette raison que ces systèmes doivent être modélisés et vérifiés formellement avant leur mise en œuvre. Un des langages les plus utilisés dans la modélisation informelle est UML (OMG, 2011). Ce dernier permet une représentation complète de ces systèmes avec une syntaxe très riche. Parmi les différents diagrammes du langage UML, les diagrammes états-transitions permettent la description des changements d'un état en réponse aux interactions avec d'autres états. Cependant, la sémantique d'UML est décrite informellement et, par conséquent, il n'est pas possible de faire une vérification formelle. Par ailleurs, plus le système est complexe plus l'analyse de sa modélisation devient difficile; c'est pour cette raison qu'un passage du modèle UML vers un modèle formel tel que les réseaux de Petri colorés est nécessaire. Il existe plusieurs travaux dans ce contexte, par exemple (André *et al.*, 2012 ; Elhillali *et al.*, 2010 ; Hu, Shatz, 2004). Notre travail est basé sur les travaux effectués dans l'article (André *et al.*, 2012), dont le principe est la transformation des diagrammes états-transitions vers les réseaux de Petri colorés en définissant une représentation et des algorithmes pour ce processus. Cet article ne considère pas la concurrence, c'est-à-dire que l'article ne prend pas en compte de multiples régions

dans un état composite qui s'exécutent en parallèle; par conséquent, les pseudos-états *fork* et *join* ne sont pas pris en compte. La figure 1 montre un exemple de diagramme états-transitions pour un système de lecture de CD (Zhang, Liu, 2010) avec une partie de sa transformation vers un réseau de Petri coloré selon (André *et al.*, 2012).



(a) Diagramme état-transition UML



(b) Réseau de Petri coloré

Figure 1. Une partie du lecteur de CD vers un réseau de Petri coloré

La première partie de notre travail a été de modifier des algorithmes ainsi que quelques définitions pour la correction d'erreurs et de cas non pris en compte dans les algorithmes de la transformation (tels que les transitions en boucle qui vont d'un état vers lui-même). Le point difficile dans cette partie est la détection des cas non pris en considération, car il n'y a pas de support logiciel associé à cet article théorique, ce qui rend donc plus difficile la compréhension de la traduction en pratique.

La seconde partie consistait à élargir l'ensemble syntaxique pris en compte dans la transformation proposée dans (André *et al.*, 2012), et en particulier l'introduction de la concurrence. Les travaux dans cette seconde partie étaient, d'une part, l'extension de la traduction proposée dans l'article au cas concurrent et, d'autre part, la définition d'une représentation pour les pseudos-états *fork* et *join*. Ceci n'était pas trivial : d'une part, l'adaptation nécessitait de nombreuses modifications dans la structure de

la sémantique et dans ses définitions. D'autre part, il est difficile de prendre en considération tous les cas possibles de façon cohérente ; cela est dû à la structure de la traduction qui était définie au départ pour le cas non-concurrent. Un autre point qui rend l'adaptation difficile est l'ambiguïté de la définition établie par l'OMG dans la spécification d'UML (OMG, 2011) qui décrit souvent les éléments sans rentrer dans les détails, ce qui pose des problèmes pour certains cas particuliers. Les modifications apportées à la traduction décrite dans l'article (André *et al.*, 2012) permettent désormais à cette dernière de prendre en considération les diagrammes états-transitions UML concurrents.

Afin d'exploiter la partie théorique définie et afin d'automatiser la transformation, une partie de notre travail a consisté à implémenter les différents algorithmes élaborés en utilisant l'outil Acceleo. Ce dernier est un outil de transformation basé sur les modèles et qui permet à partir d'un modèle de départ de générer un autre modèle. D'une part, notre implémentation permet d'appliquer différents exemples pour ensuite faire de la vérification, ce qui est le but initial de la transformation. D'autre part, elle permet de détecter les différents cas non pris en considération et, par conséquent, d'améliorer la transformation. Cependant, Acceleo n'était pas entièrement adapté au cas de notre transformation, ce qui a rendu difficile l'implémentation complète des algorithmes.

Dans le cade de notre travail, des modification et des définitions ont été ajoutées pour étendre la transformation des diagrammes états-transitions vers les réseaux de Petri colorés dans le cas concurrent. Nos perspectives incluent, d'une part, de prendre en considération dans la transformation d'autres éléments syntaxiques des diagrammes états-transitions (tels que l'aspect temporel, les événements différés) et, d'autre part, l'amélioration de la partie implémentation.

Remerciements *Ce travail a été effectué dans le cadre du stage de M2 de Mohamed Mahdi Benmoussa au LIPN (Université Paris 13, Sorbonne Paris Cité, France).*

Références

André É., Choppy C., Klai K. (2012). Formalizing non-concurrent UML state machines using colored Petri nets. *ACM SIGSOFT Software Engineering Notes*, vol. 37, n° 4, p. 1-8.

Elhillali K., Allaoua C., El Bay B., Ouassila L. (2010). A UML and colored Petri nets integrated modeling and analysis approach using graph transformation. *Journal of Object Technology*, vol. 9, p. 25–43.

Hu Z., Shatz S. M. (2004). Mapping UML diagrams to a Petri net notation for system simulation. In *SEKE*, p. 213-219.

OMG. (2011, August). *Unified Modeling Language Superstructure Version 2.4.1.*

Zhang S., Liu Y. (2010). An automatic approach to model checking UML state machines. In *SSIRI-C'10*, p. 1–6. IEEE.