VeSTA : Vérification de la préservation des propriétés d'un composant lors de son intégration dans un système temporisé

Jacques Julliand, Hassan Mountassir, and Emilie Oudot

LIFC - Laboratoire d'Informatique de l'Université de Franche-Comté
16, route de Gray, 25030 Besançon Cedex
{julliand,mountass,oudot}@lifc.univ-fcomte.fr

L'intégration de composants est une méthode de développement incrémental utilisée dans le cadre des systèmes à base de composants. Il s'agit de vérifier qu'un composant s'intègre correctement dans un environnement, i.e., que les propriétés du composant sont préservées par l'intégration. Lors d'une vérification par model-checking, cela permet de contourner le problème d'explosion combinatoire en menant la vérification sur des modèles plus petits. Dans [1], nous avons défini une τ-simulation pour les systèmes temporisés, appelée *τ-simulation temporisée sensible à la divergence et respectant la stabilité*, qui préserve les propriétés MITL (Metric Interval Temporal Logic)[2], le non-zénonisme fort, et l'absence de blocage. La propriété de composabilité exprime que tout composant A simule sa composition avec d'autres composants. Elle garantit donc la préservation des propriétés de A lors d'une intégration. La simulation définie assure la composabilité vis-à-vis de l'opérateur de composition *non bloquant* de [3] (sous certaines hypothèses simples), mais pas vis-à-vis de l'opérateur classique de composition pour les systèmes temporisés. Ainsi, pour garantir la préservation lors d'une intégration avec cet opérateur, il faut vérifier algorithmiquement la simulation. C'est l'objet de l'outil VeSTA[1] (**Ve**rification of **S**imulations for **T**imed **A**utomata).

Présentation générale. VeSTA considère donc des systèmes temporisés à base de composants, modélisés par des automates temporisés [4] avec l'opérateur de composition classique noté $\|$. La fonctionnalité principale de VeSTA est de vérifier si les propriétés locales d'un composant A sont préservées quand A est intégré dans un environnement E. Pour ce faire, VeSTA vérifie si A simule l'automate $A\|E$ réalisant l'intégration. La simulation peut être vérifiée soit totalement, pour assurer la préservation de toutes les propriétés pouvant être exprimées sur A, soit de manière partielle, pour vérifier la préservation de propriétés spécifiques exprimées sur le composant. Cette seconde approche permet d'optimiser la vérification de la simulation, mais garantit uniquement la préservation des propriétés spécifiées. Dans les deux cas, si la vérification échoue, VeSTA fournit un diagnostic graphique, composé d'une trace de $A\|E$ qui n'est simulée par aucune trace de A, ainsi que de la trace de A à laquelle il est attendu qu'elle corresponde.

[1] VeSTA est disponible à l'adresse : http://lifc.univ-fcomte.fr/~oudot/VeSTA

Eléments techniques. VeSTA a été développé en JAVA (pour l'interface utilisateur) et en C (pour les modules de vérification de la simulation). Il utilise la bibliothèque SMI (Symbolic Model Interface), qui fournit une représentation symbolique basée sur les diagrammes de décision, pour des modèles finis décrits par des réseaux d'automates. Ce choix a été guidé par l'outil Open-Kronos [5], dont nous avons adopté pour VeSTA une architecture similaire. La partie centrale de l'outil comporte trois modules. Le module `translator` traduit les deux AT A et $A \| E$ en un fichier C, contenant les structures de données et fonctions permettant de générer un graphe symbolique pour chaque AT. Ce fichier est de plus créé de manière à pouvoir le connecter à la plate-forme de vérification Open-Caesar [6]. Les modules `simul` et `Profounder` (ce dernier provient d'Open-Kronos) permettent alors de vérifier la simulation et de retourner les diagnostics.

Expérimentations. Les premiers résultats obtenus sont prometteurs. Nous avons notamment traité l'exemple de la cellule de production de [7]. Nous avons identifié des propriétés locales à certains composants de ce système, et comparé la vérification classique (directement sur le modèle complet du système, avec le model-checker KRONOS[8]) et l'approche par intégration de composants (vérification locale avec KRONOS et vérification de la préservation avec VeSTA). Il s'avère que, même sur ce système de taille raisonnable, la méthode classique nécessite un temps de calcul d'environ 20 secondes, tandis que la méthode par préservation nécessite moins d'une seconde. Les résultats détaillés sont présentés dans [9].

Références

1. Bellegarde, F., Julliand, J., Mountassir, H., Oudot, E. : On the contribution of a τ-simulation in the incremental modeling of timed systems. In : FACS'05. Volume 160 of ENTCS., Macao, Macao, Elsevier (2005) 97–111
2. Alur, R., Feder, T., Henzinger, T. : The benefits of relaxing punctuality. Journal of the ACM **43** (1996) 116–146
3. Bornot, S., Sifakis, J. : An Algebraic Framework for Urgency. Information and Computation **163** (2000) 172–202
4. Alur, R., Dill, D. : A theory of timed automata. Theoretical Computer Science **126** (1994) 183–235
5. Tripakis, S. : The analysis of timed systems in practice. PhD thesis, Universite Joseph Fourier, Grenoble, France (1998)
6. Garavel, H. : OPEN/CAESAR : An Open Software Architecture for Verification, Simulation and Testing. In : TACAS'98, Lisboa, Portugal (1998)
7. Burns, A. : How to verify a safe real-time system : The application of model-checking and timed automata to the production cell case study. Real-Time Systems Journal **24** (2003) 135–152
8. Yovine, S. : KRONOS : A verification tool for real-time systems. Journal of Software Tools for Technology Transfer **1** (1997) 123–133
9. Bellegarde, F., Julliand, J., Mountassir, H., Oudot, E. : Experiments in the use of τ-simulations for the components-verification of real-time systems. In : SAVCBS'06, Portland, USA (2006) Available on ACM Digital Library.