FAST : un model-checker pour systèmes à compteurs

Sébastien Bardin   Christophe Darlot   Alain Finkel

LSV & ENS de Cachan
61, av. du Président Wilson,
94235 CACHAN CEDEX, FRANCE
e-mail : {bardin,darlot,finkel}@lsv.ens-cachan.fr

Résumé

FAST est un outil pour la vérification de propriétés de sûreté pour des systèmes à compteurs. L'originalité de l'outil tient dans l'utilisation de représentations symboliques pour représenter des ensembles infinis et de techniques d'accélération pour augmenter les chances de convergence. FAST a été appliqué avec succès à un grand nombre de cas non triviaux. L'outil est disponible disponibles sur Internet à l'adresse suivante :

`http://www.lsv.ens-cachan.fr/fast/`

1 Vérification de systèmes à compteurs en pratique

Systèmes à compteurs. Les systèmes à compteurs sont des automates étendus par des variables entières non bornées et dont les transitions sont étiquetées par une fonction affine gardée par une formule de Presburger. Ce sont des modèles intéressants car ils permettent de généraliser de nombreux autres modèles (réseaux de Petri étendus, les protocoles Broadcast [EFM99],...). Notons que dans le cas général, même les propriétés d'accessibilité sont indécidables.

Ensembles de Presburger représentés par des automates. Une *bonne* représentation symbolique est nécessaire pour représenter de manière finie des ensembles infinis. Afin de vérifier des propriété de sûreté, elle doit être close par union, intersection et application d'une fonction de Presburger et le test du vide doit être décidable. Les *ensembles de Presburger* (ensembles de $\mathbb{N}^n$ solutions d'une formule de Presburger offrent une représentation symbolique adequat car ils peuvent être représentés et manipulés sous forme d'automates [WB00].

Calcul de l'ensemble d'états accessibles d'un système à compteurs. Pour calculer l'ensemble d'états accessibles d'un système à compteurs, la technique naïve consistant à déclencher les transitions du système une par une jusqu'à atteindre un point fixe a de grande chances de ne pas terminer. Les techniques dites d'*accélération* permettent d'améliorer la convergence en permettant de calculer en un pas l'effet d'un nombre quelconque d'itérations d'une boucle du système. Dans [FL02], il est prouvé que sous certaines conditions algébriques (souvent vraies en pratique), les ensembles de Presburger sont clos par accélération d'une fonction de Presburger. Le problème se réduit donc à trouver les "bonnes" boucles à accélérer, i.e. celles dont l'accélération conduira à l'obtention de l'ensemble des états accessibles. Nous utilisons pour cela une heuristique [BFL04] ainsi qu'une technique de réduction du nombres de cycles [FL02].

Améliorations. FAST implante maintenant les résultats de [BFL04]. Par rapport à [BFLP03], la version actuelle propose en plus une définition modulaire des systèmes à compteurs, l'accélération plus efficace limitée aux translations convexes et un langage de stratégies plus élaboré.

Outils similaires. A notre connaissance, seulement deux autres outils (LASH et TREX) utilisent des techniques d'accélération pour vérifier des système à compteurs. LASH [LAS] utilise comme FAST des automates pour représenter les ensembles de Presburger, mais est limité aux gardes convexes et l'utilisateur doit spécifier les cycles à accélérer. TREX [TRE] utilise des Matrices de Différences de Bornes Paramétrées Contraintes (CPDBM) [AAB00] comme représentation symbolique des états. Les CPDBM sont plus expressives que Presburger, la contrepartie étant l'indécidabilité d'opérations simples telles que le test du vide ou d'inclusion. De plus, les gardes se limitent à un sous ensemble des convexes et les fonctions traitées sont plus simples que celles de FAST et LASH. Néanmoins, TREX ne se limite pas aux systèmes à compteurs mais offre également la possibilité de spécifier des horloges (denses).

2 L'outil Fast

Entrées et sorties. Fast prend en entrée à la fois un modèle et une stratégie et retourne en sortie un message indiquant si le système est sûr ou non.

Le *modèle* décrit le système à compteurs à analyser sous forme d'un ensemble de systèmes à compteurs inter-opérant via des transitions synchronisées et des variables partagées[1]. La *stratégie* décrit la séquence de calculs à effectuer pour vérifier la validité d'un système grâce à un langage script qui opère sur les régions (ensembles d'états). Les opérations classiques sur les ensembles (union, intersection, ...) et sur le calcul d'ensembles accessibles (post, accélération, en avant, en arrière) sont fournis.

Architecture. L'architecture client-serveur de Fast est présentée dans la figure 1. Le serveur est implanté en C++ et l'interface en Java.

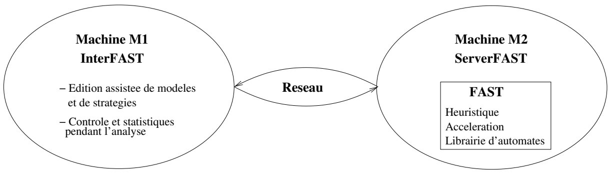

Fig. 1 – Architecture

Le *serveur*, moteur de calcul de Fast, contient une librairie d'automates de Presburger, les algorithmes d'accélération et des heuristiques de recherche. Le *client* (qui communique par sockets avec le serveur) dispose d'une interface graphique permettant l'édition guidée des modèles et stratégies ainsi que l'utilisation de stratégies prédéfinies. Une fois le calcul lancé, le client affiche différentes statistiques (temps écoulé, mémoire utilisée, ...).

3 Conclusion

Fast a été appliqué avec succès à un grand nombre d'études de cas (environ 40) disponibles sur le net (essentiellement tirées de [Del]). Ces exemples vont de réseaux de Petri à des abstractions de programmes multi-thread Java et des systèmes embarqués complexes (TTP [BFL04]). Fast affiche un taux de réussite de 80%. Cela montre que les choix de conception mis en œuvre (semi-algorithme) sont raisonnables en pratique. La figure 2 donne des statistiques sur quelques exemples.

Actuellement, Fast fonctionne bien sur des systèmes d'une vingtaine de variables. La prochaine étape est le passage à l'échelle. Pour cela, nous travaillons à une bibliothèque de Presburger optimisée pour la vérification.

[1] L'expressivité est identique à celle d'un seul système à compteur mais le travail de spécification est facilité.

Étude de cas	variables	transitions	temps (s)	mem.(MB)	n. états	n. acc	taille cycle	n. cycles
Dekker ME	22	22	21.72	5.48	5	36	1	22
CSM	13	13	45.57	6.31	6	32	2	35
FMS	22	20	157.48	8.02	21	23	2	46
Swimming Pool	9	6	111	29.06	30	9	4	47
Producer/Consumer (Java) - N	18	14	723.27	12.46	58	86	2	75
Lift Controller - N	4	5	4.56	2.90	14	4	3	20
TTP (1 faute - N stations)	10	17	1186.24	73.24	1140	31	1	17
TTP (2 faute - 10 stations)	20	21	12365	588	273 427	-	1	21

FIG. 2 – Résultats sur un Intel Pentium 933 MHz avec 512 Mbytes

Références

[AAB00] A. Annichini, E. Asarin, and A. Bouajjani. Symbolic techniques for parametric reasoning about counter and clock systems. In *Proc. 12th Int. Conf. Computer Aided Verification (CAV'2000), Chicago, IL, USA, July 2000*, volume 1855 of *Lecture Notes in Computer Science*, pages 419–434. Springer, 2000.

[BFL04] S. Bardin, A. Finkel, and J. Leroux. Faster acceleration of counter automata in practice. In *10th International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS 2004)*, Lecture Notes in Computer Science, Barcelona, March 2004. Springer.

[BFLP03] S. Bardin, A. Finkel, J. Leroux, and L. Petrucci. FAST : Fast Acceleration of Symbolic Transition systems. In *Proc. 15th Int. Conf. Computer Aided Verification (CAV'2003), Boulder, CO, USA, July 2003*, volume 2725 of *Lecture Notes in Computer Science*, pages 118–121. Springer, 2003.

[Del] G. Delzanno. *Home Page – Giorgio Delzanno.* `http://www.disi.unige.it/person/DelzannoG/`.

[EFM99] J. Esparza, A. Finkel, and R. Mayr. On the verification of broadcast protocols. In *Proc. 14th IEEE Symp. Logic in Computer Science (LICS'99), Trento, Italy, July 1999*, pages 352–359. IEEE Comp. Soc. Press, 1999.

[FL02] A. Finkel and J. Leroux. How to compose Presburger-accelerations : Applications to broadcast protocols. In *Proc. 22nd Conf. Found. of Software Technology and Theor. Comp. Sci. (FST&TCS'2002), Kanpur, India, Dec. 2002*, volume 2556 of *Lecture Notes in Computer Science*, pages 145–156. Springer, 2002.

[LAS] LASH *web.* `http://www.montefiore.ulg.ac.be/~boigelot/research/lash/`.

[TRE] TREX *homepage.* `http://www.liafa.jussieu.fr/~sighirea/trex/`.

[WB00] P. Wolper and B. Boigelot. On the construction of automata from linear arithmetic constraints. In *Proc. 6th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2000), Berlin, Germany, Mar.-Apr. 2000*, volume 1785 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 2000.