Génération de l'espace d'états utilisant les MDD pour modèles avec fonctions

Afonso Sales, Brigitte Plateau

Laboratoire d'Informatique de Grenoble
Projet Mescal
51, avenue Jean Kuntzman
38330 Montbonnot
{afonso.sales, brigitte.plateau}@imag.fr

Mots-clés : génération de l'espace d'états, diagrammes de décision, formalismes structurés.

1. Introduction

L'utilisation des *Diagrammes de Décision Binaire* (BDD - *Binary Decision Diagrams*) [2] a eu une grande influence sur la vérification de modèles de systèmes. Les BDD ont fourni des gains de performance pour la génération de l'espace d'états atteignables (RSS - *Reachable State Space*) de modèles en utilisant des techniques traditionnelles explicites [3].

La génération de l'espace d'états est le principal défi pour la grande partie des outils de vérification formelle. L'ensemble d'*états initiaux* et la fonction "*next-state*" de modèles sont utilisés par les générateurs symboliques traditionnels de l'espace d'états afin d'obtenir le RSS représenté par un BDD du système modélisé. Toutefois, les BDD ne sont pas normalement performants pour des modèles de systèmes composés par sous-systèmes qui interagissent indépendamment et synchronisent entre eux par des événements partagés.

Les modèles de systèmes décrits par des formalismes structurés - comme les Réseaux de Petri Stochastiques (SPN - *Stochastic Petri Nets*) [1] et les Réseaux d'Automate Stochastiques (SAN - *Stochastic Automata Networks*) [5] - possèdent des représentations structurées des états et de la fonction "*next-state*". Les états sont représentés par des vecteurs, où chaque position du vecteur représente l'état d'un sous-système du modèle. L'ensemble des vecteurs d'états peut être naturellement représenté par des *Diagrammes de Décision Multi-valués* (MDD - *Multi-valued Decision Diagrams*) [6].

2. Diagrammes de Décision Multi-valués - MDD

Un MDD est une extension de la structure de BDD, appliquée à une logique non-binaire. Les MDD peuvent représenter facilement des espaces d'états de modèles structurés en utilisant des fonctions du type $\mathcal{S}_K \times \ldots \times \mathcal{S}_1 \rightarrow \{\mathbf{0}, \mathbf{1}\}$, où $\mathcal{S}_l$ est l'espace d'état local (fini) du sous-système l ($K \geq l \geq 1$) et K est le nombre de sous-systèmes du modèle.

Des concepts basiques préalablement étudiés pour les BDD (l'*ordre* et la *réduction* de l'espace d'états) sont aussi pertinents pour le format compact d'un MDD. De la même façon que les BDD, les opérations (union, différence et intersection) sont aussi utilisées directement pour les MDD.

La génération du RSS utilisant les MDD se fonde sur deux idées de base. La première est de représenter l'espace d'états par niveaux (sous-systèmes du modèle) de manière hiérarchique. La deuxième est de générer l'espace d'états *symboliquement*, *i.e.*, pour chaque itération de l'algorithme, on peut découvrir *potentiellement* un grand ensemble d'états atteignables. Vu que les opérations sur les MDD travaillent directement sur les noeuds (au lieu de travailler individuellement sur les états codés par les noeuds), la génération symbolique utilisant les MDD est très performante, autant en réduction de l'utilisation de mémoire qu'en coût de calcul.

2.1. Implantation

Pour profiter de la structure modulaire des formalismes structurés de haut-niveau pour la génération symbolique du RSS, on utilise des *expressions de Kronecker* pour représenter les fonctions caractéristiques "*next-state*" et des MDD pour représenter les espaces d'états. La fonction "*next-state*" d'un modèle structuré est représentée par un format disjonctif-conjonctif organisé par événement et sous-système. Grâce à cette nouvelle démarche, on peut profiter de la localité des événements de modèles structurés pour l'algorithme de génération symbolique de l'espace d'états.

Les noeuds d'un MDD sont divisés dans K *groupes*, un pour chaque niveau (sous-systèmes). Pour éviter des noeuds doubles, chaque groupe est géré par une *table de hachage* de dimension variable qui supporte insertions et suppressions de noeuds pendant l'exécution. Elle permet aussi un *temps constant* d'accès pour chaque élément. Pour les opérations sur les MDD, un *cache* est utilisé pour réutiliser des opérations avec les mêmes paramètres.

Les structures utilisées pour la génération du RSS

(les MDD et la fonction "*next-state*"), ainsi que l'extension de l'algorithme de génération, sont développées dans un prototype. Ultérieurement, un nouveau module du logiciel PEPS [7] sera développé afin de permettre la génération symbolique du RSS de modèles décrits par le formalisme SAN (qui permet que les taux des événements soient exprimés par fonctions).

3. Génération de l'espace d'états atteignables

L'idée de base de l'algorithme de *saturation* [4] est de tirer exhaustivement tous les événements du modèle pour un noeud quelconque du MDD (et ses descendants, *i.e.*, les noeuds des niveaux inférieurs) de telle façon que ce noeud trouve son format final : *saturé*. De plus, l'algorithme explore les noeuds des niveaux de *bas en haut*, *i.e.*, lorsque un noeud est exploré, tous ses noeuds descendants se trouvent déjà dans son format saturé. La fonction "*next-state*" est modifiée pour profiter de la localité des événements (tenant compte des sous-systèmes concernés par le tirage de l'événement).

Pour l'utilisation de modèles qui possèdent des taux d'occurrence des événements exprimées par des fonctions, on utilise aussi des MDD pour la représentation d'ensemble d'états où la fonction ne s'annule pas. On utilise la notation MDD_{f} pour indiquer cet ensemble d'états d'une fonction f. Toutefois, la génération du MDD_{f} est faite explicitement : l'algorithme de génération évalue *état-par-état* la fonction f. Si le résultat de la fonction f est différent de zéro pour un état évalué, cet état est *ajouté* (opération d'union) au MDD_{f}.

On propose une extension de l'algorithme de saturation afin d'obtenir la génération du RSS de modèles avec des taux d'occurrence des événements qui ont des fonctions. Un des avantages de l'utilisation des fonctions est de décrire, d'une façon plus simple et efficace, des interactions complexes entre les composantes du système modélisé [5].

4. Conclusion

Pour cette première étude, des résultats intéressants ont été obtenus pour la génération du RSS de systèmes complexes à grand espace d'états. Par exemple, on a modelisé le problème classique du dîner des philosophes, où chaque philosophe est modélisé par trois états (*pensant*, fourchette *droite* et *gauche*), pour deux modèles SAN : (1) modèle qui utilise seulement des événements synchronisants (*taux constants*) et (2) modèle qui utilise des événements locaux (avec des *taux fonctionnels*). Pour ces modèles avec 1 000 philosophes, où l'espace d'états potentiel est égal à $1,32 \times 10^{477}$ et le RSS est égal à $5,09 \times 10^{382}$, on a obtenu un MDD du RSS en $2,16 \times 10^{-1}$ secondes pour le modèle (1) et $2,39 \times 10^{-1}$ secondes pour le (2). Toutefois, dans (2), le temps pour générer le RSS est égal à $1,89 \times 10^{-1}$ secondes et les autres $5,08 \times 10^{-2}$ seconds sont passées pour obtenir les MDD_{f} du modèle.

Pour des modèles qui utilisent des fonctions à grand espace d'états (*e.g.*, le modèle de partage de ressources, où l'espace d'états de la fonction est égal à l'espace d'états produit du modèle), la génération des MDD_{f} est la partie qui demande le plus de calcul, vu que le MDD_{f} d'une fonction f est obtenu explicitement en évaluant état-par-état de son espace d'états.

Bien que la génération des MDD_{f} est encore la partie *la plus lente*, cette démarche permet la génération du RSS de modèles à gigantesques espaces d'états d'une façon performante.

Les perspectives de ce travail sont d'améliorer la génération explicite des MDD_{f} du modèle ou de proposer une autre démarche pour utiliser des fonctions pour la génération du RSS de modèles sans explorer tout l'espace d'états d'une fonction.

Bibliographie

1. M. Ajmone-Marsan, G. Balbo, G. Conte, S. Donatelli, et G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. John Wiley and Sons, 1995.
2. R. E. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8) :667–691, 1986.
3. J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, et L.J. Hwang. Symbolic Model Checking : 10^{20} states and beyond. *Information and Computation*, 98(2) :142–170, June 1992.
4. G. Ciardo, G. Lüttgen, et A. S. Miner. Exploiting interleaving semantics in symbolic state-space generation. *Formal Methods in System Design*, 31(1) :63–100, 2007.
5. P. Fernandes, B. Plateau, et W. J. Stewart. Efficient descriptor - Vector multiplication in Stochastic Automata Networks. *Journal of the ACM*, 45(3) :381–414, 1998.
6. T. Kam, T. Villa, R. K. Bryaton, et A. Sangiovanni-Vincentelli. Multi-valued decision diagrams : theory and applications. *Multiplie-Valued Logic*, 4(1-2) :9–62, 1998.
7. PEPS. Perf. Evaluation of Parallel Systems. http ://www-id.imag.fr/Logiciels/peps/.