Dynamic Partial Order Reduction et Directed Model-Checking pour la vérification de programmes distribués asynchrones

Martin Quinson, Equipe Myriads (https://team.inria.fr/myriads/)
Thierry Jéron, Equipe SUMO (http://www.irisa.fr/sumo/)

Mots-clés : directed model-checking, dynamic partial order, programmes distribués

Contexte: On s'intéresse à la correction de programmes distribués communiquant de manière asynchrone par canaux. Dans ce cadre, le *software model checking* vise à vérifier automatiquement des propriétés logiques directement sur le logiciel (sans phase de modélisation préalable) par exploration exhaustive de ses comportements.

Malheureusement l'exploration de systèmes réels, même de taille modérée, est coûteuse, à cause de l'explosion explosion combinatoire due à l'exploration: le graphe d'état de n processus séquentiels finis concurrents est exponentiel par rapport à la taille de chaque processus. Depuis plusieurs décennies, des approches visent donc à réduire cet espace d'états, en exploitant par exemple des propriétés de redondance, de symétrie, etc.

Parmi elles, DPOR (dynamic partial ordering reduction) [DPOR] tire parti du fait que deux événements causalement indépendants n'ont aucune influence mutuelle sur la satisfaction de propriétés dans certaines logiques (e.g. LTL-X). Pour le cas plus simple de la détection des blocages, il suffit par exemple d'explorer un graphe d'état réduit contenant une exécution par classe de comportements équivalents par commutation d'événements adjacents indépendents (trace de Mazurkiewicz). Les variantes de DPOR proposées différent essentiellement par la définition du sous-ensemble de transitions suffisant à explorer (stubborn set, persistent set, ample set, source set, etc) pour garantir la couverture de tous les comportements utiles [ODPOR][UDPOR]. Pour gagner en efficacité, un compromis doit être trouvé entre la réduction du nombre de traces à explorer, et la précision de cet ensemble de transitions.

D'autre part, le model checking dirigé [DirectedMC], [DirectedMC1] est une approche visant à prioriser l'ordre d'exploration afin d'arriver plus rapidement aux états problématiques, s'inspirant en particulier de l'algorithme A* utilisé en IA. La priorité est basée sur la somme de la distance depuis l'état initial et une heuristique qui approxime une distance entre l'état courant et le but à atteindre (e.g. l'état d'erreur). Si le système est correct, l'exploration reste exhaustive, mais si une erreur est effectivement atteignable, la performance et la longueur des contre-examples peut être nettement améliorée. Là aussi plusieurs variantes existent, diffèrant par l'heuristique employée.

Objectif : L'objectif de ce stage est d'étudier la combinaison de ces deux approches, en particulier dans le contexte du modèle de programmation distribué asynchrone. L'ensemble des transitions nécessaires à DPOR doit alors être géré comme une liste à priorité, cette priorité dépendant d'une distance à définir en fonction du modèle de programmation et de la classe de propriété visée. Une telle combinaison a déjà été expérimentée dans le cas des ample sets et stubborn sets, pour d'autres modèles de programmation et les réseaux de Petri [DirectedDPOR0],[DirectedDPOR1],[DirectedDPOR2]. On s'intéressera ici à au moins l'un des algorithmes DPOR [ODPOR] ou [UDPOR], techniques plus récentes et a priori plus précises et performantes. On étudiera en particulier quelles sont les heuristiques d'intérêt à la fois pour notre modèle de programmation et la logique LTL-X. Ce qui suppose a priori d'exhiber des abstractions du modèle de programmation et de la logique, qui permettent de définir les distances appropriées et des heuristiques efficaces à calculer. L'objectif final est

d'implémenter et d'expérimenter ces algorithmes dans l'outil SimGrid [McSimGrid17], [Pham19].

Références

[DPOR] Dynamic partial-order reduction for model checking software
C. Flanagan, P. Godefroid
POPL 2005, ACM Sigplan Notices, 2005 - dl.acm.org
https://dl.acm.org/doi/pdf/10.1145/1047659.1040315

[DirecteMC] Survey on Directed Model Checking,
S. Edelkamp, V. Schuppan, D. Bošnački, A. Wijs, A. Fehnker, H. Aljazzar.
Workshop on Model Checking and Artificial Intelligence (MoChArt'08).
https://link.springer.com/chapter/10.1007/978-3-642-00431-5_5

[DirectedMC1] Directed model checking with distance-preserving abstractions
K. Dräger, B. Finkbeiner, A. Podelski
Int J Softw Tools Technol Transfer (2009) 11:27–37 DOI 10.1007/s10009-008-0092-z
https://link.springer.com/content/pdf/10.1007/s10009-008-0092-z.pdf

[DirectedDPOR0] Partial Order Reduction in Directed Model Checking,
A. Lluch-Lafuente, S. Edelkamp, and S. Leue.
SPIN 2002,
https://sci-hub.se/https://link.springer.com/content/pdf/10.1007/3-540-46017-9_10.pdf

[DirectedDPOR1] Partial-order reduction and trail improvement in directed model checking
S. Edelkamp, S. Leue, A. Lluch-Lafuente
Int J Softw Tools Technol Transfer (2004) 6: 277–301 / Digital Object Identifier (DOI) 10.1007/s10009-004-0151-z
https://link.springer.com/article/10.1007/s10009-004-0151-z

[DirectedDPOR2] Automata-Driven Partial Order Reduction and Guided Search for LTL Model Checking.
P. Jensen, J. Srba, N. J. Ulrik, and S. M. Virenfeldt.
International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'22).
https://homes.cs.aau.dk/~srba/files/JSUV:VMCAI:22.pdf

[McSimGrid17] Verifying MPI Applications with Mc SimGrid,
T. A. Pham, T. Jéron, M. Quinson.
Workshop on Software Correctness for HPC Applications (Correctness'17)
https://hal.inria.fr/hal-01632421

[Pham19] Unfolding-based Dynamic Partial Order Reduction of Asynchronous Distributed Programs,
T. A. Pham, T. Jéron, M. Quinson.
39th International Conference on Formal Techniques for Distributed Objects, Components, and Systems (FORTE'19)
https://hal.inria.fr/hal-02109769/document

Compétences requises : Le stage s'adresse à un étudiant intéressé par les méthodes

formelles et l'algorithmique et motivé par le développement d'outils.