Machine Learning pour le Test basé sur les modèles formels et la réduction de l'espace d'états

Supervision	Kais Klai - kais.klai@lipn.univ-paris13.fr Carlos Olarte - olarte@lipn.univ-paris13.fr Jaime Arias - arias@lipn.univ-paris13.fr
Lieu	LIPN, CNRS UMR 7030 Université Paris 13 99 avenue Jean-Baptiste Clément 93430, Villetaneuse, France

1 Contexte

L'évolution des systèmes critiques se caractérise par une complexité croissante. La vérification de leurs propriétés est reconnue comme un problème difficile. Les logiciels orchestrant de tels systèmes doivent réagir correctement et en particulier face aux situations critiques. La vérification de la correction d'un système vis-à-vis d'une spécification peut se faire de deux manières : par le test ou par la vérification formelle. La vérification par test est la technique la plus souvent utilisée. Elle consiste à exécuter le logiciel sur des données d'entrée et à évaluer chaque couple entrée/sortie produit par rapport aux spécifications du logiciel, en utilisant un oracle dédié [1]. La vérification par test souffre de plusieurs limites principales : La première est qu'elle n'est pas exhaustive dans le sens où elle opère sur un sous ensemble d'exécutions possibles (une sous approximation). Le fait que ce sous ensemble vérifie les propriétés désirées n'implique pas forcément que le système les vérifie. Le deuxième obstacle est que les différents cas nécessaires pour tester un logiciel sont de plus en plus nombreux et difficiles à trouver. On parle de couverture structurelle/fonctionnelle : les cas de test doivent couvrir un maximum de scénarios possibles. Les méthodes formelles de vérification de systèmes ont pour objectif d'assurer la fiabilité des systèmes, c'est-à-dire leur bonne spécification et l'absence d'erreur. Idéalement, pour concevoir le logiciel d'un système concurrent donné, il faudrait spécifier formellement ce système à l'aide d'un modèle mathématique à partir duquel on pourrait raisonner et vérifier les propriétés attendues. En réalité, ce processus se heurte à plusieurs problèmes d'ordre pratique et théorique. Un premier obstacle apparaît au niveau de la définition du langage de spécification utilisé. Si celui-ci est trop expressif, alors on ne peut pas, mathématiquement, l'analyser automatiquement. Un second obstacle à la vérification formelle des systèmes complexe, en particulier celles basées sur la technique du model checking [2], est l'explosion combinatoire des états possibles des systèmes. Malgré les avancées spectaculaires de la technologie des ordinateurs, il arrive que l'on soit incapable d'analyser intégralement des systèmes par manque d'espace mémoire ou de temps. Par ailleurs, les logiciels critiques étant de plus en plus complexes, l'utilisation d'une seule technique de vérification peut se révéler insuffisante. Les deux approches peuvent effectivement être combinées.

2 Objectifs

Comme le test basé sur les modèles formels repose sur la vérification des critères de couvertures, cette thèse vise à apporter des réponses aux conditions d'expression des critères de couvertures, comme moyen de qualification du processsus de test, sous la contrainte de l'emploi des techniques de réduction de l'espace d'état qui est nécessaire lors de la vérification. Cette proposition doit prendre en considération le problème de la mise à l'échelle en étudiant la pertinence du *machine learning*.

En effet, cette thèse s'inscrit dans la continuité d'un travail de l'équipe qui vise la génération des entrées de tests en utilisant les critères de couvertures structurelles du système cible en les extrayant du Graphe d'Observation Symbolique (SOG) correspondant [3, 4, 5, 6]. Dans un premier temps, nous avons défini les conditions en relation avec la structure du modèle exprimé à l'aide de réseau de Petri du système qui permet d'identifier des patterns de modèles et d'extraire un sous ensemble de transitions qui, lorsqu'elles sont couvertes, assurent la couverture de toutes les transitions du réseau. Le SOG est donc construit sur la base de l'observation de ce sous ensemble de transitions. La construction du SOG nous permet de générer des traces observées couvrant toutes les transitions observées. La génération de ces traces se fait à la volée. Elles sont ensuite complétées pour former des séquences abstraites (intercalant les transitions observées par des transitions non observées) afin de fournir des séquences couvrant l'ensemble des transitions du système [6]. Afin de fournir un ensemble de séquences de transitions optimal, nous avons eu recours à la programmation linéaire où l'objectif est de minimiser le nombre et la taille des séquences

couvrantes. Lorsque la taille du SOG croît en termes de nombre d'états à explorer, la programmation linéaire se heurte au problème de l'explosion combinatoire et devient inapplicable. Par conséquent, nous visons actuellement à utiliser le machine learning pour orienter l'exploration de l'espace d'états de gros systèmes (ayant un nombre d'états accessible trop grand), voire des systèmes infinis.
L'idée est de choisir un modèle IA approprié (une étude comparative des modèles IA existants sera nécessaire) qui sera entrainée à la volée sur une partie du graphe (dont la taille peut être supportée par la solution exacte). Dès que la taille du graphe devient trop conséquente, le modèle IA sera sollicité pour orienter le parcours et se rapprocher de la solution exacte (permettant la couverture totale des transitions du système).

Une fois ce travail fait, notre prochain objectif sera d'étendre notre approches à d'autres critères de couverture permettant une meilleur qualité des tests générés. En particulier la couverture des places d'un réseau de Petri, du flux de données et des spécifications seront considérés [7].

Enfin, un dernier objectif de cette thèse sera d'adapter cette approche basée IA au model checking LTL. En effet, nous pensons que cete technique de vérification souffre du même problème d'explosion combinatoire et est basée également sur un parcours de graphe à la recherche de cycles acceptant. Utiliser le machine learning pour orienter l'exploration d'un graphe afin d'atteindre ce cycle le plus rapidement possible et en consommant le moins de mémoire possible pourrait être prometteur.

3 Plan de travail prévisionnel

La première année de thèse sera consacrée aux deux tâches suivantes :

- Une étude comparative des modèles de machine learning pour les graphes et le choix du plus pertinent.
- Une mise en œuvre d'un outil pour la couverture des transitions d'un réseau de Petri basée sur l'IA.

Ensuite, lors de la deuxième année, l'extension de cette approche à d'autres critères de couverture sera étudiée et mise en oeuvre. Il s'agira en l'occurrence de

- Etudier les conditions qui assurent la couverture de toutes les places d'un réseau de Petri, la couverture du flux de données et la couverture des spécification.
- Mettre en œuvre et à l'échelle cette étude.

Enfin, la dernière année sera dédiée à l'adaptation de notre méthodologie au model checking LTL. Il s'agira en particulier de vérifier que le-s modèle-s choisis-s pour la problématique de couverture est aussi appropriée dans le cas du model checking (qui se ramène à la recherche d'un cycle acceptant dans un graphe).

References

[1] Paul Ammann and Jeff Offutt. *Introduction to Software Testing*. Edition 2. Cambridge University Press, New York, NY, 2017.

[2] Edmund M. Clarke, E Allen Emerson, and A Prasad Sistla. "Automatic verification of finite-state concurrent systems using temporal logic specifications". In: *ACM Transactions on Programming Languages and Systems (TOPLAS)* 8.2 (1986), pp. 244–263.

[3] Serge Haddad, Jean-Michel Ilié, and Kais Klai. "Design and evaluation of a symbolic and abstraction-based model checker". In: *International Symposium on Automated Technology for Verification and Analysis*. Springer. 2004, pp. 196–210.

[4] Kais Klai and Laure Petrucci. "Modular construction of the symbolic observation graph". In: *2008 8th International Conference on Application of Concurrency to System Design*. IEEE. 2008, pp. 88–97.

[5] Kais Klai and Denis Poitrenaud. "MC-SOG: An LTL model checker based on symbolic observation graphs". In: *International Conference on Applications and Theory of Petri Nets*. Springer. 2008, pp. 288–306.

[6] Kais Klai et al. "Symbolic Observation Graph-Based Generation of Test Paths". In: *Tests and Proofs*. Ed. by Virgile Prevosto and Cristina Seceleanu. Cham: Springer Nature Switzerland, 2023, pp. 127–146. ISBN: 978-3-031-38828-6.

[7] Hong Zhu and Xudong He. "A theory of testing high level Petri nets". In: *Proc. of the IFIP 16th world computer congress, Beijing, China*. 2000.