Sujet de stage

—

Génération d'invariants de programmes par un grand modèle de langage

Mots-clés : vérification de programme ; génération d'invariants ; apprentissage ; grand modèle de langage ; déduction automatique

Description

La génération automatique d'invariants de programmes est un verrou scientifique majeur dans le domaine de la vérification automatique de programmes. Pour vérifier qu'un programme est correct (c.-à-d. qu'il satisfait une spécification donnée), il est nécessaire d'annoter ce programme avec des propriétés logiques vraies durant l'exécution du programme, les *invariants* [3]. Ce processus peut être effectué manuellement, mais c'est une tâche difficile et fastidieuse, il est donc nécessaire de l'automatiser pour permettre aux techniques de vérification de passer à l'échelle et de trouver une utilisation plus large.

Les grands modèles de langage (LLM, pour *large language models*) ont eu ces dernière années un impact spectaculaire dans le domaine du traitement automatique des langues, en obtenant d'excellents résultats sur des tâches variées, y compris celles qui concernent des langages de programmation [1]. La capacité des LLM à produire des énoncés d'après un contexte donné semble très adaptée pour générer des invariants de programmes [5], toutefois le fonctionnement des LLM ne tient pas compte de la sémantique des langages, et il est donc très peu probable qu'un tel modèle génère seul des invariants corrects. D'un autre côté, il existe des outils de déduction automatique (prouveurs automatiques de théorèmes, solveurs SMT, etc.) qui peuvent être utilisés pour vérifier si une formule donnée est un invariant ou non [6]. L'objectif de ce stage est de combiner l'approche statistique des

LLM avec les capacités de déduction automatique dans un algorithme de génération d'invariants de programmes, afin d'analyser automatiquement des programmes et générer leurs invariants. Le LLM génèrera des candidats qui seront vérifiés par déduction automatique. Les invariants corrects seront retenus, et pour les autres, un contre-exemple sera fourni au LLM afin qu'améliorer sa réponse.

Tâches

Selon le temps disponible, le stagiaire sera amené à effectuer les tâches suivantes :

1. création d'un corpus d'entraînement constitué de programmes annotés (avec des conditions de vérification et les invariants nécessaires à leur vérification), par exemple en prenant appui sur les programmes utilisés pour des compétitions de vérification automatique [4, 2] ;
2. choix et entraînement d'un grand modèle de langage pour générer des invariants ;
3. développement d'un algorithme combinant la génération par LLM et la déduction automatique pour générer, vérifier et raffiner des invariants ;
4. évaluation de l'algorithme développé.

Compétences

La personne candidate devra suivre un master en informatique ou un diplôme équivalent, et avoir un intérêt pour les méthodes d'apprentissage ainsi que pour la logique et la vérification logicielle. Des compétences en programmation seront nécessaires pour implémenter les solutions développées.

Informations pratiques

Le stage sera effectué au LIRMM (Laboratoire d'Informatique, Robotique et Microélectronique de Montpellier), dans les équipes MaREL et Texte. La durée du stage est de cinq mois, et il sera gratifié conformément aux dispositions légales en vigueur. Il sera encadré par

— Simon Robillard (équipe MaREL) `simon.robillard@lirmm.fr`
— Maximos Skandalis (équipe Texte) `maximos.skandalis@lirmm.fr`

Références

[1] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv :2107.03374*, 2021.

[2] Claire Dross, Carlo A Furia, Marieke Huisman, Rosemary Monahan, and Peter Müller. Verifythis 2019 : a program verification competition. *International journal on software tools for technology transfer*, 23(6) :883–893, 2021.

[3] Carlo A Furia, Bertrand Meyer, and Sergey Velder. Loop invariants : Analysis, classification, and examples. *ACM Computing Surveys (CSUR)*, 46(3) :1–51, 2014.

[4] Vladimir Klebanov, Peter Müller, Natarajan Shankar, Gary T Leavens, Valentin Wüstholz, Eyad Alkassar, Rob Arthan, Derek Bronish, Rod Chapman, Ernie Cohen, et al. The 1st verified software competition : Experience report. In *International Symposium on Formal Methods*, pages 154–168. Springer, 2011.

[5] Kexin Pei, David Bieber, Kensen Shi, Charles Sutton, and Pengcheng Yin. Can large language models reason about program invariants ? In *Proceedings of the 40th International Conference on Maching Learning*, 2023.

[6] Natarajan Shankar. Automated deduction for verification. *ACM Computing Surveys (CSUR)*, 41(4) :1–56, 2009.