Verification of GPU Programs: Evaluation Challenges (Extended Abstract)

Hannah Zicarelli Tiago Cogumbreiro
Univeristy of Massachusetts Boston, Boston, USA
hannah.zicarelli001@umb.edu, tiago.cogumbreiro@umb.edu

This talk aims to facilitate tool-diverse experimental evaluations of static and symbolic analysis on datasets of hundreds of GPU programs. We will discuss the difficulty of executing large evaluations of GPU programs, the infrastructure we developed to address such challenges, along with extending our infrastructure to handle new analysis problems.

Impediments to evaluation at scale. Numerous existing published evaluations on GPU software verification lack tool diversity (*e.g.*, [1,3–5] compare two or fewer tools) and depth (*e.g.*, average lines of code analyzed: 13 for [6], 16 for [7], and 30 for [4]). A significant factor hampering comparative studies of static analysis and symbolic analysis (*e.g.*, symbolic execution) is that each tool requires tool-specific code alterations. Further, code alterations are incompatible across tools, which leads to dataset authors needing to maintain a slightly different version of the program being analyzed per tool. It is therefore challenging to conduct tool-diverse studies that target hundreds of real-world GPU programs.

Evaluation infrastructure. To address this challenge, we will present our evaluation infrastructure created for a recent comparative evaluation [2] of data-race freedom (DRF) verification tools. We store GPU kernels in a tool-agnostic format with metadata, and employ templates to generate tool-specific programs adapted to each tool's differing requirements. Additionally, we use automation for both the program generation process and the execution of reproducible experiments. This infrastructure enabled a comparative evaluation of 3 static DRF checkers on 227 real-world GPU programs and 5 static and dynamic data-race finders on 250 synthetic programs, a scale that could be prohibitive when relying on manual dataset preparation.

Ongoing improvements. While we developed our evaluation infrastructure for a study of static DRF checkers, work remains to apply it to other verification problems. One problem seeing ongoing work is static analysis quantifying GPU shared memory bank conflicts. Currently, the infrastructure can check tool output with regular expressions to identify DRF analysis results, but such checks are insufficient for tools reporting quantitative results in the form of arithmetic expressions. To address this shortcoming, we want to parse arithmetic expressions reported in tool output.

Presentation outline. This talk is structured in three parts. First, we expose obstacles limiting the scale of evaluations, including the differing requirements of static and dynamic tools. Second, we present our evaluation infrastructure. Third, we discuss ongoing work adapting our infrastructure to an evaluation of static analysis tools quantifying bank conflicts. Finally, we close on the goal of

Submitted to:
PLACES 2022

generalizing this evaluation infrastructure as a stand-alone open source tool to enable tool-diverse evaluations of hundreds of GPU programs as the default for future software verification publications.

References

[1] Adam Betts, Nathan Chong, Alastair F. Donaldson, Shaz Qadeer & Paul Thomson (2012): *GPUVerify: a Verifier for GPU Kernels*. In: *Proceedings of OOPSLA*, ACM, pp. 113–132, doi:10.1145/2384616.2384625.

[2] Tiago Cogumbreiro, Julien Lange, Dennis Liew & Hannah Zicarelli (2021): *Checking Data-Race Freedom of GPU Kernels, Compositionally*. In: *Proceedings of CAV*, Springer, pp. 403–426, doi:10.1007/978-3-030-81685-8_19.

[3] Guodong Li & Ganesh Gopalakrishnan (2012): *Parameterized Verification of GPU Kernel Programs*. In: *Proceedings of IPDPSW*, pp. 2450–2459, doi:10.1109/IPDPSW.2012.302.

[4] Guodong Li, Peng Li, Geof Sawaya, Ganesh Gopalakrishnan, Indradeep Ghosh & Sreeranga P. Rajan (2012): *GKLEE: Concolic Verification and Test Generation for GPUs*. In: *Proceedings of PPoPP*, 47, ACM, pp. 215–224, doi:10.1145/2370036.2145844.

[5] Peng Li, Guodong Li & Ganesh Gopalakrishnan (2014): *Practical Symbolic Race Checking of GPU Programs*. In: *Proceedings of SC*, IEEE, pp. 179–190, doi:10.1109/SC.2014.20.

[6] Phillipe Pereira, Higo Albuquerque, Hendrio Marques, Isabela Silva, Celso Carvalho, Lucas Cordeiro, Vanessa Santos & Ricardo Ferreira (2016): *Verifying CUDA Programs Using SMT-Based Context-Bounded Model Checking*. In: *Proceedings of SAC*, ACM, pp. 1648–1653, doi:10.1145/2851613.2851830.

[7] Mingyuan Wu, Yicheng Ouyang, Husheng Zhou, Lingming Zhang, Cong Liu & Yuqun Zhang (2020): *Simulee: Detecting CUDA Synchronization Bugs via Memory-Access Modeling*. In: *Proceedings of ICSE*, ACM, pp. 937–948, doi:10.1145/3377811.3380358.