Metamorphic Testing of Android Graphics Drivers

Alastair F. Donaldson
Google
London, UK
Email: afdx@google.com

***Abstract*—My MET 2019 invited talk will focus on GraphicsFuzz, a technique and tool for automated metamorphic testing of the shader compiler components of graphics drivers. Starting out as a research project at Imperial College London, GraphicsFuzz led to a spin-out company of the same name, which was acquired by Google in 2018. The technology has since been open sourced and is being used by Google to test Vulkan and OpenGL ES graphics drivers and supporting tools.**

***Keywords*-software testing; compilers; device drivers;**

Graphics drivers are hard to build and hard to test, yet their correct operation is critical. They are hard to *build* because (a) they must be highly optimized in order to serve the competing demands of multiple processes and the operating system, and (b) graphics architectures and APIs are constantly moving targets. They are hard to *test* because graphics APIs, such as Vulkan and OpenGL ES, deliberately under-specify the required results of computations. While this facilitates efficient API implementations from a range of hardware vendors, it makes it hard or impossible to dictate with certainty what counts as an acceptable result for a computation. And they are *critical* because they are the interface between the operating system, graphics hardware and display: a mobile device that is functioning perfectly well with the exception of its display is not very useful!

In this talk I will describe our experience building and using GraphicsFuzz [1], an automated testing tool for graphics drivers based on metamorphic testing [2], [3]. GraphicsFuzz originated as a research project at Imperial College London, which led to the GraphicsFuzz spin-out company that was acquired by Google in 2018. GraphicsFuzz is now a central line of defence in ensuring the quality of graphics drivers for the Android platform.

Originally presented at MET 2016 [4] and later elaborated into a full paper [5], GraphicsFuzz focuses on finding bugs in *shader compilers*. A shader is a program that runs on GPU hardware, and shader compilers, which translate shaders from high level and intermediate representations such as GLSL and SPIR-V to low-level GPU machine code, are among the most complex components in a graphics driver. Inspired by work on equivalence modulo inputs testing for C compilers [6], in turn extended to test OpenCL compilers [7], GraphicsFuzz automatically finds cases where a shader compiler has generated wrong code by: **(1)** starting with an original, high-value shader (e.g. captured from a game); **(2)** applying *semantics-preserving transformations* to this shader to produce a family of *equivalent* shaders that should render identical or very similar images to the original when executed on the same GPU; **(3)** calling out compiler bugs by identifying significant mismatch between images; **(4)** homing in on the root causes of bugs by automatically reducing a transformed shader until the difference between the original and transformed shaders is as small as possible whilst still preserving the mismatch.

The key strength of metamorphic testing relates to step 3 above: by comparing images for semantically-equivalent shaders, we can identify cases where the shader compiler has generated wrong code via a mismatch between images, without requiring an oracle to tell us what the correct image should actually be.

As well as providing technical details of how the approach works, I will discuss some of the main open problems and opportunities related to applying metamorphic testing automatically at scale, including how to cope with potential false alarms related to floating-point round-off error, how to automatically triage and de-duplicate bug reports, and how to test the metamorphic testing tool itself.

References

[1] The GraphicsFuzz Authors, "GraphicsFuzz testing framework," 2019, https://github.com/google/graphicsfuzz.

[2] T. Chen, S. Cheung, and S. Yiu, "Metamorphic testing: a new approach for generating next test cases," Department of Computer Science, The Hong Kong University of Science and Technology, Tech. Rep. HKUST-CS98-01, 1998.

[3] S. Segura, G. Fraser, A. B. Sánchez, and A. R. Cortés, "A survey on metamorphic testing," *IEEE Trans. Software Eng.*, vol. 42, no. 9, pp. 805–824, 2016.

[4] A. F. Donaldson and A. Lascu, "Metamorphic testing for (graphics) compilers," in *MET*. ACM, 2016, pp. 44–47.

[5] A. F. Donaldson, H. Evrard, A. Lascu, and P. Thomson, "Automated testing of graphics shader compilers," *PACMPL*, vol. 1, no. OOPSLA, pp. 93:1–93:29, 2017.

[6] V. Le, M. Afshari, and Z. Su, "Compiler validation via equivalence modulo inputs," in *PLDI*. ACM, 2014, pp. 216–226.

[7] C. Lidbury, A. Lascu, N. Chong, and A. F. Donaldson, "Many-core compiler fuzzing," in *PLDI*. ACM, 2015, pp. 65–76.