Software Engineering Seminar (WiSe 2016/17)

Efficient Symbolic Execution

Description

Symbolic execution [1] was introduced by James C. King more than 40 years ago and it succeeded in several recent applications [2], most influenced by the dramatic increase in the computational power of modern computers and, thereby connected, the improved capabilities of decision procedures. We still see great advancement for this technique like the handling of complex and recursive data structures, as well as multi-threading and a hybrid approach called dynamic symbolic execution to handle native code [3]. Nevertheless, there are remaining key challenges like constraint solving and the path explosion. One of the latest approaches for more efficient application of symbolic execution is the work by Yang et al. *Memoized Symbolic Execution* [4].

The student is supposed to focus on *Memoized Symbolic Execution* and investigate the state of the art.

Prerequisites

A basic knowledge of software verification techniques is preferable, otherwise it needs some more effort to get into the topic.

References

[1] James C. King. Symbolic execution and program testing. *Commun. ACM*, 19(7):385–394, July 1976.

[2] Alessandro Orso and Gregg Rothermel. Software testing: A research travelogue (2000–2014). In *Proceedings of the on Future of Software Engineering*, FOSE 2014, pages 117–132, New York, NY, USA, 2014. ACM.

[3] Corina S. Păsăreanu and Willem Visser. A survey of new trends in symbolic execution for software testing and analysis. *International Journal on Software Tools for Technology Transfer*, 11(4):339, 2009.

[4] Guowei Yang, Corina S. Păsăreanu, and Sarfraz Khurshid. Memoized symbolic execution. In *Proceedings of the 2012 International Symposium on Software Testing and Analysis*, ISSTA 2012, pages 144–154, New York, NY, USA, 2012. ACM.

Contacts

Yannic Noller (`noller@informatik.hu-berlin.de`)
Software Engineering Group
Institut für Informatik
Humboldt-Universität zu Berlin