Software Engineering Seminar (WiSe 2016/17)

# Synthesizing Framework Models for Symbolic Execution

## Description

Symbolic execution [2] was introduced by James C. King more than 40 years ago and it succeeded in several recent applications [3], most influenced by the dramatic increase in the computational power of modern computers and, thereby connected, the improved capabilities of decision procedures. Although, recent work in this area made impressive achievements [4] 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 [4], symbolic execution is still limited to a small set of real-world applications. One problem is that such applications use third-party frameworks and important control and data flows occur via these frameworks. In most cases these frameworks cannot be executed symbolically because we have no access to the source code and they are way too large and complex for an efficient symbolic analysis. The standard solution would be to manually create a simplified (abstracted) framework model that can be used in a symbolic analysis. These models are created by hand and must be updated during the software evolution, hence, this is very error-prone and cumbersome. Jeon et al. [1] proposed with their paper *Synthesizing Framework Models for Symbolic Execution* the first step toward automatically synthesizing framework models.

The student is supposed to focus on Jeon et al. and investigate the state of the art.

### Prerequisites

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

## References


[1] Jinseong Jeon, Xiaokang Qiu, Jonathan Fetter-Degges, Jeffrey S. Foster, and Armando Solar-Lezama. Synthesizing framework models for symbolic execution. In *Proceedings of the 38th International Conference on Software Engineering*, ICSE '16, pages 156–167, New York, NY, USA, 2016. ACM.

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

[3] 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.

[4] 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):339–353, 2009.


## Contacts


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