A Study on Checking Equivalence of Software Programs using Symbolic Execution
Bachelor thesis

Supervisors: Prof. Dr. Alexander Pretschner, Dr. Martin Ochoa, Sebastian Banescu
Email: {alexander.pretschner, martin.ochoa, sebastian.banescu} @ in.tum.de
Phone: +49 89 289 – 17, 314
Starting date: immediately

Context
Software transformations occur very frequently in practice. Such transformations are done for various purposes though manual refactoring or automatic transformations. However, in this thesis we focus on transformations that must preserve the functional behavior of the software that is being transformed. For instance, software transformations applied by compiler optimizations or obfuscation transformations must preserve the functional behavior of the software, i.e. the original and the transformed counterpart must yield the same output when given the same input.

Symbolic execution has been proposed as a technique which could be used to find whether 2 programs have the same functional behavior [1, 2, 3]. Since it is assumed that symbolic execution can explore a large part of the execution tree of a program, it can also find an input for which the output of the original and transformed programs differs. However, in practice there are many limitations to this approach [4, 5]. In this thesis we intend to uncover the impact these limitations have on equivalence testing.

Goal
The goal of this thesis is to design and implement a system that: given a software tool which performs software transformations, can check if these transformations preserve the functional behavior of a set of programs. Note that we do not want to prove the behavior preserving property, but simply be able to point out whether there are programs for which certain transformations of the tool under test break the original functionality. Therefore, we suggest that the underlying system use a symbolic execution engine.

The second goal of this thesis is to perform an evaluation of the implemented system based on case studies using multiple source code or intermediate language obfuscation engines and several software samples as the transformation targets.

Work-plan
1. Develop understanding of symbolic execution from the provided references and relevant literature cited by these.
2. Familiarization with the KLEE symbolic execution engine and with various software transformation engines.
3. Design and implementation of the equivalence checking system:
   a. The system receives as input:
      i. The binary or the path of the software transformation tool
      ii. The set of command line arguments for that tool
      iii. (Optional) The set of software programs which are the transformation targets.
   b. The system outputs a report specifying the test cases which prove that a certain transformation does not preserve the behavior of the
transformation targets.
4. Evaluation of the implementation based on case-studies
5. The final thesis document must contain:
   a. Description of the problem and motivation for the chosen approach
   b. Description of the theoretical background
   c. Implementation description
   d. Evaluation of implementation
   e. Discussion on potential threats to validity
   f. Conclusions and future work.

Deliverables
- Virtual machine able to run a demo of the implementation, including instructions on how to run the demo.
- The VM should also include the source code of the implementation.
- Technical report with comprehensive documentation of the implementation, i.e. design decision, architecture description, API description and usage instructions.
- Final thesis report written in conformance with TUM guidelines.

References