Master Thesis

Slicing of Python Bytecode

Problem Statement

In our research project, we develop a runtime verification framework with look-ahead capabilities for Python programs. We analyze Python programs not directly, but the generated Python Bytecode which is an assembly-like intermediate representation for Python programs. Runtime verification checks, while the program is executing, whether a given property is fulfilled and returns a verdict. In order to derive a verdict about future program states a sliced version of the program shall be executed.

Program slicing computes irrelevant program parts for a specified value. In the figure below, you find a simple program on the left and on the right is the sliced program which had the value of variable **s** at the return statement as the input. Irrelevant for the computation of the sum, are in this example the logging instrution and the computation of the variable **p**.

```
def f(n: int):
  s = 0
  i = 0
  p = 1
  while i <= n:
    s = i + s
    i = i + 1
    p = i * p
  logger.log(p, s)
  return p/s
```

```
def f(n: int):
  s = 0
  i = 0
  while i <= n:
    s = i + s
    i = i + 1
  return p/s
```

EXAMLE PROGRAM (LEFT) AND A SLICED VERSION (RIGHT)

Your Tasks

In this thesis a slicing method for Python Bytecode should be developed and is to be evaluated using an industrial use case.

- Literature research on slicing techniques for assembly-like languages
- Adaption and implementation of a promising approach which can handle pointer aliasing
- Evaluation on an industrial use case

Your Profile

The analysis should be implemented in Python and therefore intermediate language skills are preferred. Optional is knowledge about formal methods and slicing, but general understanding of program analysis is helpful.

Contact

Thomas Henn, M. Sc. RWTH
henn@embedded.rwth-aachen.de