Concolic Testing of Interpreters

Philippos Boon Alexaki

Supervisors: C. B. Poulsen
C. R. van der Rest

Motivation and Research Question

Students write interpreters for CPL. Giving feedback and examining correctness using predefined tests is not complete.

Can we find errors in student's interpreters using concolic testing? What errors are easy to catch?

```
void f(int x) {
    if (x ≠ 0) {
        print(1/x);
    } else {
        error("bug");
    }
}
```

Concolic Testing

Run the program recording all path constraints on the input

Path constraint: When an input variable is used in a conditional, we record what constraint was set

Generate new value by falsifying one of the path constraints and solving the set of constraints

Run the program again, but now we are guaranteed to follow a new path

Method

Develop concolic execution for a core language

Translate faulty interpreters into core language

Generate inputs for interpreters with concolic execution

Compare results with correct interpreters

Limitations

Constraint solving is not a one-to-one map with semantics of the language

Not all possible types are supported by the constraint solver

Nested pattern matching

Automatically detecting the class of the found error

Results

Finds all errors in a simple arithmetic expression language. Has difficulty finding errors in a language with closures. Promising first results.

`x=24` → `!(x ≠ 0) ⇒ x=0` → Found bug in program