Measuring Similarity of Programs

Eric John IMT2017016
Srihari Vemuru IMT2017045
Seethamraju Purvaj IMT2017039

Types of Similarity

- Representational Similarity
    - Textual
    - Lexical
    - Structural
- Semantic or Behavioural Similarity
    - Functional
    - Execution

Functional Similarity

- Functional Similarity concerns about input/output behaviour of programs
- This can be defined mathematically , as the subject is related to mathematical approximation.

Microsoft Paper

Measuring Code Behavioral Similarity for Programming and Software Engineering Education

- Sihan Li, Xusheng Xiao, Blake Bassett, Tao Xie, Nikolai Tillmann

Definitions

- **Program Execution**

  A function exec: $P \times I \rightarrow O$

  I is the input domain

  O is the output domain of Program P

- **Behavioural Equivalence**

  Two programs P_1 P_2 are equivalent if $I_1 == I_2$ and

  $\text{exec}(P_1, I_1) = \text{exec}(P_2, I_2)$

- **Behavioural Difference**

  Two programs P_1 P_2 are different if I is the input domain of both and

  exec(P_1 , I) != exec(P_2, I) . i.e.

  $\exists$ i $\in$ I, exec(P_1 , i) != exec(P_2 , i)

- **Behavioral Similarity**

  Two programs P_1 P_2 are similar if I is the input domain of both and

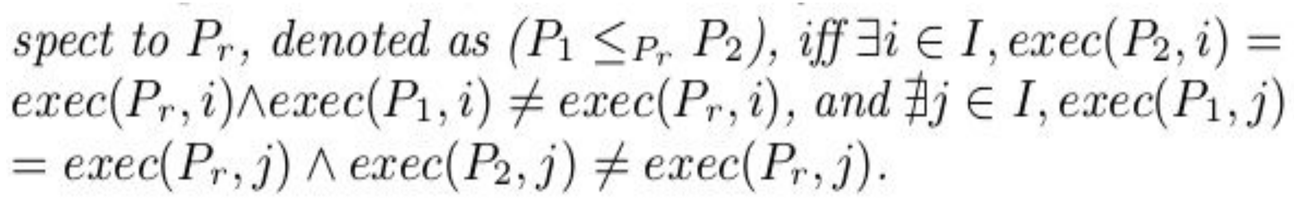
  $\exists$ I_s $\in$ I , such that exec(P_1 , I_s) == exec(P_2 , I_s) and

  $\forall$ j $\in$ I \ I_s , exec(P_i , j) != exec(P_2, j)

- **Behavioral subsumption**

  Let P1 P2 and Pr be the programs that share the same input domain I

  P2 behaviourally subsumes P1 with respect to Pr, when



spect to P_r, *denoted as* ($P_1 \leq_{P_r} P_2$), *iff* $\exists i \in I, exec(P_2, i) = exec(P_r, i) \wedge exec(P_1, i) \neq exec(P_r, i)$, *and* $\nexists j \in I, exec(P_1, j) = exec(P_r, j) \wedge exec(P_2, j) \neq exec(P_r, j)$.

Metrics to Measure Similarity

- Random Sampling
- Single Program Symbolic Execution
- Paired Program Symbolic Execution

Random Sampling (RS)

Randomly sample uniformly distributed inputs and run them on the programs that we have. The ratio of agreed inputs to all the inputs in the set is RS

Advantages: Low computational cost, gives a good approximation to the actual behavioral similarity

Disadvantages: Might overlook the corner cases which differ in terms of behaviour.

Single-program Symbolic Execution (SSE)

One program is considered as reference program and test inputs are generated for this using DSE (Dynamic Symbolic Execution). These are tested on the other program and the ratio of agreed inputs to all the inputs in the set is SSE.

Advantages: These test cases are more likely to cover corner cases.

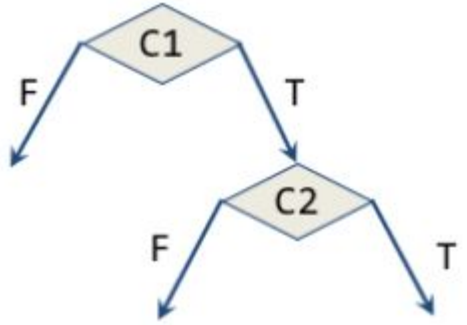
Disadvantages: 1. Doesn’t consider program under analysis 2. Programs may have infinitely many paths when there are loops.

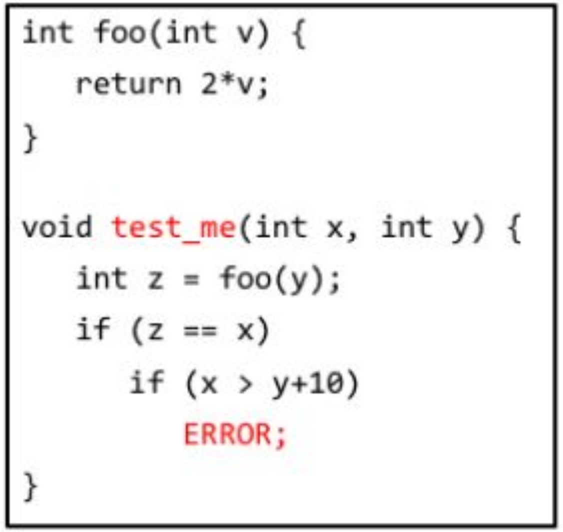
Paired-program Symbolic Execution (PSE)

The input domain of the paired program is exactly the same as that of the two programs. The paired program runs the two programs on the same input and asserts their outputs to be the same. Then we leverage DSE to generate test inputs on the paired program and compute the proportion of test inputs that pass the assertion over the generated inputs as the PSE value.

Advantages: Since the test generation for PSE is performed on the paired program, which is constructed from both programs, PSE is also sensitive to the program under analysis and likely to reveal more behavioral differences than SSE does.

DSE





```
int foo(int v) {
   return 2*v;
}

void test_me(int x, int y) {
   int z = foo(y);
   if (z == x)
      if (x > y+10)
         ERROR;
}
```

Implementations: KLEE (C family of languages), PEX(.NET Framework), jCute(JAVA)

Evaluation Methodology

Each sequence of programs satisfy either of the following properties:

- the sequence has a later program behaving equivalently as an earlier program.
- the sequence has a later program behaviorally subsuming an earlier program with respect to the reference program.

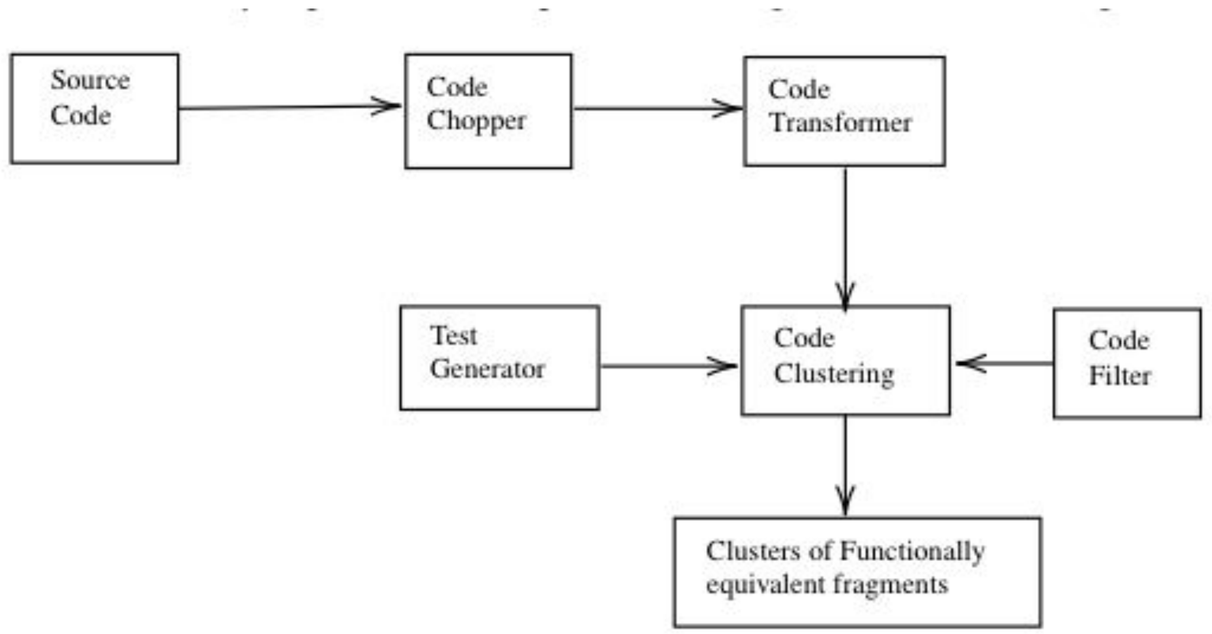
Apply combination of Metrics

- How effective are our metrics in ordering programs based on their progress towards the reference program?
- How accurate are our metrics in approximating the behavioral similarity?

Intended Market

- online programming
- software engineering education
- hint generation
- progress indication
- automatic grading
- reducing repeated code fragments in large code databases.

Structure of Evaluation Algorithm



"Automatic Mining of functionally equivalent codes via random testing" Lingxiao JIANG, Zhendong SU

Components

- Code Chopper
- Code Transformer
- Code Filter
- Test Generator
- Code Clustering

Code Chopper

- A collection of subsequences of statements
- Removing indentations and spaces
- Theoretically Take the pre order traversal of the syntax tree of the functions

Code Transformer

- Create compilable programs. (requirement)
- Add headers
- Declare data types

Code Filter

- Remove lexically similar code
- Decrease input for code clustering

Test Generator

- Primitive and derived types
- Type cast to largest type
- Type casr to smallest type

Code Clustering

- Pairwise similarity

Thank You