Fault Localization for Multi-language Programs

Taehoon Kwak, Yiru Jeon, Yunho Kim, Shin Hong, Moonzoo Kim
Software Testing & Verification Group, KAIST

Bongseok Ko, Byeongcheol Lee
Programming Systems Laboratory, GIST

KAIST

Gwangju Institute of Science and Technology

Summary

- Develop a mutation-based fault localization for locating complex bugs in multi-language programs

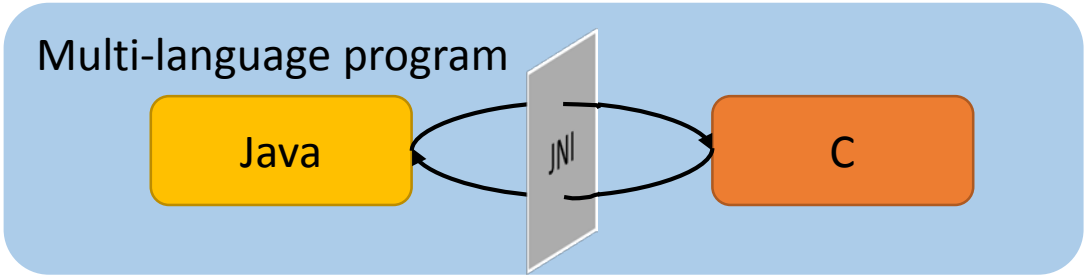
Multi-language programs

- A program which consists of more than two different programming languages.
  - Mix-and-match benefits of different languages
  - Reuse legacy code
  - Examples
    - special-purpose language(SQL) + general purpose language(C)
    - procedural language(C) + object-oriented language(JAVA)

Challenge

- Debugging of multi-language programs is difficult because a bug may be involved with code fragments of different languages.
  - For example, a memory leak bug occurs if a Java object is illegally manipulated by C code fragments.

- Research target: JAVA/JNI programs
  - JNI enables programmers to directly use native languages such as C, C++, and assembly with Java programs
  - Widely used in Android applications



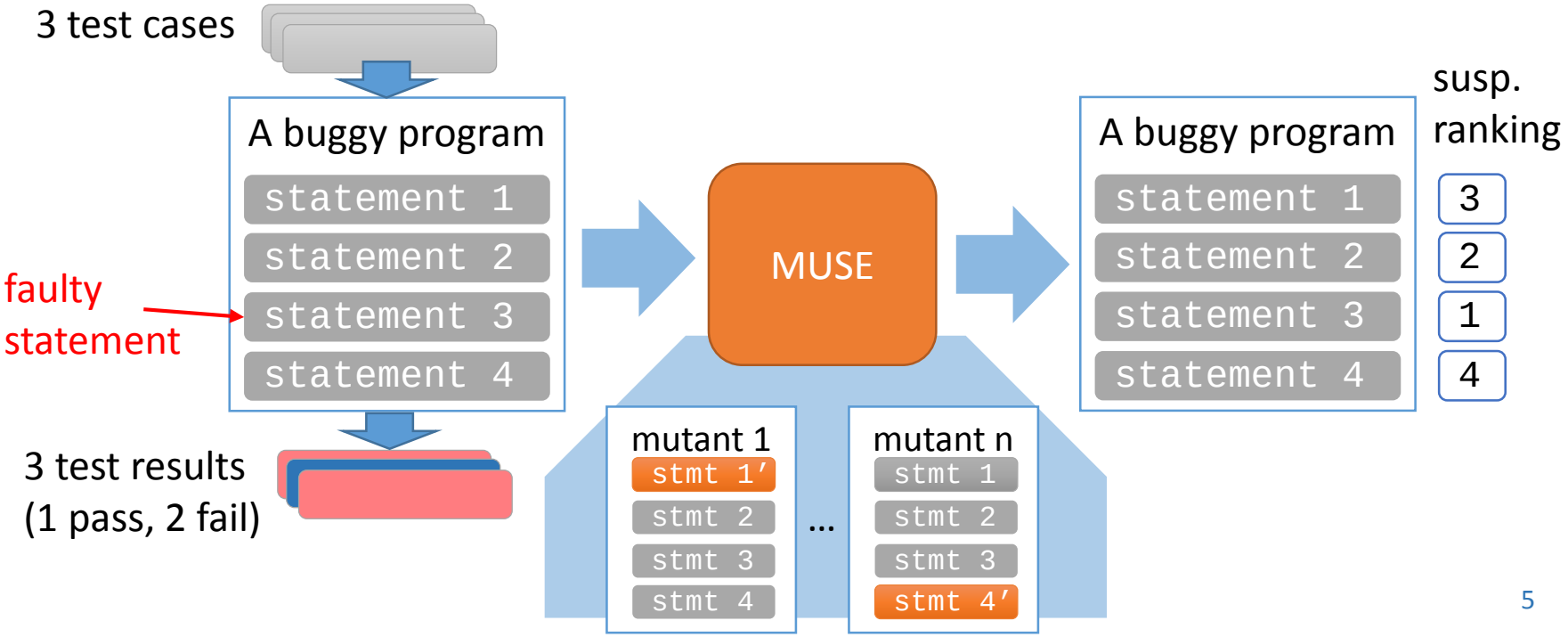
Approach: **MU**tation-ba**SE**d fault localization (MUSE)

For a buggy program with failing and passing test cases

(1) Mutate each line of a target program to generate *mutants*

(2) Check if the mutants from a line are *likely fixed* the bug (*golden mutants*)

(3) Report the line with many golden mutants as bug location



Progress

- 4 complicate bugs are collected from the bug repository.
  - 2 bugs in sqlite-jdbc
  - 1 bug in Java-gnome
  - 1 bug in Azureus
- MUSE ranks the buggy statement at the top of the suspiciousness ranking for 2 out of 4 collected bugs.
- An example of MUSE output

line	Statement	Pass	Fail	F2P	P2F	#mutants	Susp. ranking
⋮							
2526	`Java_org_sqlite_NativeDB_changes (...)`	586	5	0	0	0	12
2528	`jint ret = 0;`	586	5	0	0	0	12
2529	`ret = sqlite3_changes (gethandle (...));`	586	5	**12**	967	13	**1**
2530	`return ret;`	586		4	325	22	2
⋮							

402 lines

A buggy statement in sqlite-jdbc

m_1 : `ret *= sqlite3_changes(gethandle(...));`
m_2 : `ret >>= sqlite3_changes(gethandle(...));`
m_3 : `return ret;`
m_4 : `;`

Future Work

- Create mutation operator specialized for Java/JNI programs
  - Memory leak bugs
  - Pending exceptions
  - API misuses

- Study more real-world Java/JNI bugs
  - Bugs with complicated error scenarios

Fault Localization for Multi-language Programs

Taehoon Kwak, Yiru Jeon, Yunho Kim, Shin Hong, Moonzoo Kim
Software Testing & Verification Group, KAIST

Bongseok Ko, Byeongcheol Lee
Programming Systems Laboratory, GIST

KAIST

Gwangju Institute of Science and Technology