CodeSpider: Automatic Code Querying with Multi-modal Conjunctive Query Synthesis

Chengpeng Wang, Prism Group, The Hong Kong University of Science and Technology

Existing Code Querying Tools

- IDEs
  - String match and structural search
  - E.g., IntelliJ
- Code affairs

Restrictive search template

Laborious manual effort

Stage I: Sketch Generation

Stage II: Query Refinement

Stage III: Query Selection

CodeSpider: Synthesizing Conjunctive Queries

Figure 1: Structure of CodeSpider

Figure 2: Code affairs

Figure 3: CodeSpider: Finding Best Abstraction

Figure 4: Evaluating CodeSpider

ID Description
1. Float values of which the identifier contains “code”
2. Cast expressions from double-type to float type
3. Expressions comparing long int with int
4. Cast expressions casting long to int
5. Expressions comparing a variable and boolean literal
6. New expressions of ArrayList
7. Logical and expressions with literal as operand
8. The import of LocalDateTime
9. The import of the classes in log4j
10. Labeled statements
11. If-statements with a boolean literal as a condition
12. For-statements with a boolean literal as a condition
13. Public methods with void return type
14. Methods receiving a parameter with Log4jUtils type
15. Classes with a login method
16. Classes containing a field with float type

Figure 5: Program IR: Relational Representation

Figure 6: CodeSpider: Synthesizing Conjunctive Queries

Figure 7: Ease of use: Use Datalog-based analyzers as a black box

Figure 8: Capability: Leverage various relations describing program properties

Figure 9: Separating positive tuples from negative tuples

Figure 10: Dual metrics

Figure 11: High Efficiency

Figure 12: Average time cost: 3.35 sec

Figure 13: Maximal time cost: 8.91 sec

Figure 14: Minimal time cost: 2.23 sec

Figure 15: 14 tasks finished in 4 sec