IWSC 2016

Challenges in Behavioral Code Clone Detection

Fang-Hsiang Su,
Jonathan Bell, and
Gail Kaiser
Columbia University

Why Detecting Code Clones

- Code Clone: *Similar* code
- How to define *similar:*
  - Look-alike, function-alike, behavior-alike
  - 4 types of clones
  - Syntactically, Structurally, Semantically similar, etc.
- Helpful for Developers
  - Comprehend programs
  - Search for useful APIs
  - Re-engineer software systems

State Of The Art

- General Procedure
  - Abstract programs + Compute similarity
- Static Analysis
  - Token based: CCFinder, Baxter's, etc.
  - Abstract Syntax Tree: Deckard, Bellon's, etc.
  - Program Dependence Graph: JPlag, Krinke's, etc.
- Dynamic Analysis
  - Observe program I/Os: EQMiner, Deissenboeck's, etc.
  - Observe program side effects: Blanket Execution, etc.

Goal: Detect behave/function-alike programs

Question:
Static analysis detect all?

Argument: Probably no, static *approximates* dynamic

- What:
  - Detect programs with similar behavior
- How:
  - Effective abstraction for runtime behavior
  - Appropriate metrics to measure behavior
  - Powerful algorithm to compute similarity

Application

- Program comprehension
  - An user study shows how developers comprehend programs [1]
  - 50% of comprehension strategies relevant to similar code
- Cross-binary detection of similar programs
  - Detect similar programs under different languages, instruction sets
  - Not only software engineering, but also security community
- Code search
- More

1. W.Maalej,R.Tiarks,T.Roehm,and R.Koschke.On theComprehension of Program Comprehension. *ACM Transactions on Software Engineering Methodology*, 23(4):31:1–31:37, Sept. 2014.

Research Schedule

- Detect programs with the same (similar) I/Os
  - Deissenboeck's challenges to detect *identical* I/O clones proposed by EQMiner in Object Oriented languages
  - What are *I/Os*, how to generate *valid inputs*, how to *compare program outputs*
  - Our work : an *in-vivo* approach with *configurable* I/O comparison models to detect *functionally similar* programs
- Detect programs with similar runtime behaviors
  - Interpret runtime behaviors of *programs as graphs* at instruction level
  - Design a powerful *(sub) graph isomorphism solver* to detect patterns (clones) among programs

Conclusion

- Most current work focuses on static approach
- Static clones are *approximation* of real program behavior
- Dynamic approaches to detect *similar code* are challenging
  - What’s the abstraction of runtime behavior?
  - What’s the metric to evaluate runtime similarity?
  - What’s the effective computational model?

We look forward to overcoming them!