Automated Error Diagnosis Using Abductive Inference

Işıl Dillig
College of William & Mary

Thomas Dillig
College of William & Mary

Alex Aiken
Stanford University

**Motivation**

- If we use sound program analysis tool to verify a property, answer is either yes or no
- If answer is yes, program is error-free
- If answer is no, there are two possibilities:
  - Either the program is indeed buggy
  - Or report is a false alarm

**When Verification Fails**

- When verifier fails to prove property, user must decide whether report is real bug or false alarm.
- But manually classifying error reports is time-consuming and error-prone.
- Furthermore, user must redo all the reasoning the tool performed just to discover where it became stuck.
- Very painful process for most users of static analysis tools!

**Our Goal**

- A new technique for semi-automating error report classification when automated program verification fails

**Key Ideas**

- Allows verifier to interact with user by asking small, relevant queries until report is classified as real bug or false positive
- Queries capture only the information verifier is missing ⇒ user contributes facts verifier could not decide on its own
- Answering queries much easier than classifying error report

---

**Key Ideas, cont.**

**Proof Obligation via Abductive Inference**

- Given known facts \( F \) and desired outcome \( O \), abductive inference finds simple explanatory hypothesis \( E \) such that \( F \land E \models O \) and \( \text{SAT}(F \land E) \)
- We use abductive inference to generate simple explanations that either guarantee that program is error-free or definitively buggy
- These abductive explanations are presented as queries to user

**Failure Witnesses**

- Proof obligation query used to show report is false alarm
- We generate another query, called failure witness query, to show report is a real bug
- To generate failure witness query, solve a dual abductive inference problem:
  \[ \Delta \land F \models O \] and \( \text{SAT}(\Delta \land F) \)
- If \( \Delta \) can hold in some program execution, then report is real bug!

---

**Computed Abductive Explanations**

- Abduction is useful, but how do we compute these explanations?
- Given invariants \( F \) and desired outcome \( O \), how to find explanation \( E \) s.t. \( F \land \neg E \land \text{SAT}(F \land E) \)
- Trivial solution is \( E = \neg O \), but useless b/c same as asking user to prove assertion
- Want solutions that are as simple and as general as possible!

**Experimental Evaluation**

- Performed user study to evaluate new technique
- Hired 56 programmers through oDesk and asked them to classify error reports
- Each programmer asked to classify (randomly selected) half of reports manually, and other half using our technique
- Manual classification: Given code and error report, decide if bug, false alarm, or unknown
- Our technique: Given code and series of queries, asked to answer “Yes”, “No”, or “Don’t know”
- Based on answers to queries, report classified automatically

**Results of User Study**

- With manual classification, programmers classified 51.1% of reports incorrectly
- With assisted classification, programmers classified only 7.3% of reports incorrectly
- Our technique dramatically improves classification accuracy
- Also dramatically reduces time needed to classify report
- Using manual classification, programmers need 293 seconds on average
- Using new technique, programmers take 55 seconds on average

**Summary**

- New technique to help programmers classify error reports as real bugs or false alarms
- Uses abductive inference to compute simple queries that capture what analysis is missing
- Interacts with user until report is classified as bug/false alarm