Background for "KISS: Keep It Simple and Sequential"

cs264
Ras Bodik

spring 2005

Guest Lecture

- Shaz Qadeer, Microsoft Research
  - Monday, Jan 24, 4pm in 320 Soda
  - title: Context-Bounded Model Checking of Concurrent Software
  - based on two papers
    - KISS: Keep It Simple and Sequential [PLDI'04]
    - Context-Bounded Model Checking of Concurrent Software [MSR-TR]
- guest lecture replaces Wed, Jan 26, cs264 lecture
- read the first paper and write a summary
  - due Monday 3pm
  - email it to cs264@imail.eecs.berkeley.edu
  - graded Pass / Fail

Paper summary: 5 short paragraphs

1. Summary: what's the problem? what's the solution?
   - paraphrase the paper
   - offer a different spin
   - cut through the hype (if applicable)
2. Key new ideas: what's good about the paper?
   - what new insights or techniques made this paper possible?
   - or simply, what did you like?
3. Limitations: what's weak in the paper?
   - limitations not mentioned in the paper
   - impact of limitations, whether mentioned or not

Paper summary: 5 short paragraphs

4. What next?
   - applications of the technique
   - future directions
   - cs264 project ideas
5. Missing background
   - which parts of the paper you didn't understand
   - background you didn't have to understand the paper
   - unfamiliar formalisms: e.g., type inference rules, semantics
   - unfamiliar techniques: e.g., model checking
   - unfamiliar lingo: e.g., happens-before relation

Upcoming reading assignment

- Read Chapter 1 from textbook by Friday next week
  - overview of analysis techniques
  - 30 pages, somewhat technical
  - textbook may not be in the bookstore by Monday
  - but I'll have photocopies of Chapter 1 for you on Monday
  - pick them up at the guest talk or from mailbox on my door

Topics

- some background for the KISS paper
  - data race
  - assertions
  - model checking
  - instrumenting the program
  - under-approximation
  - abstract language

Data race

- Concurrency bugs
  - like other bugs, have various forms
  - is there one pattern that describes many of them?
- Data race: a race condition on data accesses
  - when two threads (may) access same memory location simultaneously (but two loads ok)
- It's (only) a heuristic
  - false positives: some data races are not bugs (example?)
  - false negatives: doesn't catch all bugs (example?)
- False positives inspire a refined definition
  - data race: when two accesses are not separated by a synchronization

Assertions

- Some bugs are program-specific
  - cannot be described with a general pattern like "data race"
- Programmers have a way to catch these bugs: assertions
  - check the program state at some program point
    - ex. does 'x.parent' indeed point to x's parent node?
  - check the program state across several program points
    - ex. has the tree been traversed in sorted order by the 'data' field?
    - may require instrumenting the program with a "state" variable
    - ex. a global variable storing the value of last 'data' field
- A programmer evaluates assertions at run-time
  - but they can also be "evaluated" by a static analyzer
  - static debugging: prove that no assertion can fail, for any input

Model checking

- KISS reduces parallel analysis into sequential analysis
  - translates a multithreaded program *P* into a sequential *P'*
  - *P'* simulates (some) parallelism in *P*
  - simulation by P' serving as its own scheduler
- The sequential P' analyzed in a model checker SLAM
- What's a model checker?
  - **very simplified view:**
    constant propagator that checks if any assertion can fail
  - **an important detail:**
    i) analyze branches to exclude infeasible paths
    ii) may also analyze each path separately (no path merges)

Instrumentation

Serves two purposes!

1. Adds assertions to check for temporal properties
   - ex. thread must not enter driver after driver was stopped
   - in this paper, done manually
2. Translates *P* into *P'*
   - adds simulation of thread scheduling
   - automatic

Abstract language

- Explain the technique for C is too hard
  - so the paper explains it on a simplified language
- Side note: KISS translates twice
  - from P in C to P in the "parallel language"
  - from parallel P to sequential P'
- The parallel language defined
  - using (abstract) syntax (in Fig 3)
  - semantic defined informally in the text

Under-approximation

- Soundness is hard for this problem
  - because the problem is undecidable
  - would likely lead to many false positives
- Solution:
  - under-approximate
  - in contrast to sound (conservative) static analysis, this collects a subset of facts that may happen
  - because it examines a subset of possible parallel executions
- how to approximate is the art of program analysis