CSC2/455 Software Analysis and Improvement
Program Analysis

Sreepathi Pai

URCS

April 22, 2019

Outline

And now for something (not so completely) different

Program Analysis in Industry

Basic Notions

Discussion

Outline

And now for something (not so completely) different

Program Analysis in Industry

Basic Notions

Discussion

So far

- Data flow analysis
- Loop analysis
- What next?

Compilers are not the only program analyzers

- Compilers are probably the most used program analyzers
- But are severely time constrained
- Finding *program* errors is not primary goal
  - Syntax errors, type errors
  - Code generation primary goal

Program/Software Analysis

- Software is increasingly mission-critical
- Can kill people!
  - Boeing 737 MAX(?)
  - Therac-25 (X-ray)
  - Industrial Robotics
- (less extreme?) Can lose money
  - Software crashes
  - Data loss
- Can we analyze programs for *functional* correctness?
  - Topic of the next few lectures

Outline

And now for something (not so completely) different

Program Analysis in Industry

Basic Notions

Discussion

SLAM (Microsoft, early 2000s)

- MS isolated most crashes to buggy drivers
- Static Driver Verifier project
  - Would verify driver code (in C) for correctness
- Used *model checking*
  - Models programs as finite-state machines
  - I used a similar tool (CBMC) to check your assignments

Infer (Facebook, early 2010s)

- Checks C, C++, Objective C, Java and Android code
- Used for checking Facebook's mobile apps
- Open source, `https://fbinfer.com/`
  - Used by Amazon, Mozilla, Uber and Facebook and its affiliates, JD.com, etc.
- Comes with its own language AL to describe analyses
- Uses *separation logic*
  - high-level: converts programs to logic

SPARTA (Facebook, late 2010s)

- Language-independent analyzer
  - a C++ framework
- Open source, `https://code.fb.com/open-source/sparta/`
- Used in FB's RedEx tools
  - for analyzing Android binary code (`.dex`)
- Uses *abstract interpretation*
  - very similar to data flow analysis frameworks

Other efforts

- Stanford Checker
  - commercialized by Coverity, late 2000s
  - CACM article, "A few billion lines of code later: using static analysis to find bugs in the real world"
- Google's static analysis tools
  - Checker Framework for Java programs
  - Shipshape (abandoned?) (Google Tricorder)
  - CACM article, "Lessons from Building Static Analysis Tools At Google"
- Oracle's Soufflé
  - Soufflé: Logic Defined Static Analysis

Outline

And now for something (not so completely) different

Program Analysis in Industry

Basic Notions

Discussion

Limitations

- None of these frameworks and tools can escape the fact that analysis is an undecidable problem
- All compute approximations
- Must be designed to be *sound*
  - Approximations are conservative/safe
- Leads to imprecision (i.e. *incomplete*)
  - May model behaviour not in original programs
  - (recall IDEAL vs MOP vs MFP)

States and Transitions

- A program's state is a mapping of variables to values
- Programs move from one state to another
  - begin execution in subset of (initial) states
- Notions of state *before* a program point (i.e. a statement) and *after* a program point
- Relation that maps before-states to after-states is called a *transition* relation (t)
  - $\langle x, y \rangle$ (x is before-state, y is after-state)

Traces

- An execution trace of a program is a sequence of states
  - $s_0 s_1 s_2 \ldots s_n$
- An execution trace may be finite or infinite
  - $s_0 s_1 s_2 \ldots$
- The collection of partial traces can actually happen (i.e. state transitions obey the transition relation) is called the *collecting semantics*
  - I.e. for all $s_i s_j$ in trace, $\langle s_i, s_j \rangle \in t$

Example

```
x = 0
while(x < 100)
   x = x + 1;
```

- states are $\mathbb{Z}$
- initial state is $\{0\}$
- transition relation is $\{\langle x, x' \rangle | x < 100 \wedge x' = x + 1\}$
- is 0 1 2 3 part of the collecting semantics?
- is 0 2 4 6 part of the collecting semantics?

Patrick Cousot and Radhia Cousot, *Basic Concepts of Abstract Interpretation*

Outline

And now for something (not so completely) different

Program Analysis in Industry

Basic Notions

Discussion

Topics

- Abstract Interpretation in a Nutshell
- Abstract Interpretation-based Formal Methods and Future Challenges
- Liveness Analysis in SPARTA