Facebook Infer

Dhruv Maroo

What is Infer?

- Infer is a static analysis tool for programs written in Java, C++, Objective-C and C
- It’s developed by Facebook (or Meta) and written using OCaml
- Infer uses a modular analysis technique, useful for analyzing small code changes
- Infer uses *separation logic* and *bi-abduction* to do this
- Infer.AI is a relatively newer general analysis engine which uses *abstract interpretations*

Separation Logic

- Extension of Hoare logic
- Introduces a new operator (*), known as *separating conjunction*
- Separating conjunction is used to refer to disjoint heaps
- Reasons about programs using *frame rule*
- Frame rule states that if a program executes safely in a small state, it can also execute in a bigger state, where the additional state will stay unaffected
- Side condition for frame rule requires that the program does not modify any free variables in the additional state
- This allows for modular analysis of the programs

Bi-abduction

- Used to find that “small” and the “big” state required for separation logic
- Does this by finding a *frame* and an *anti-frame* to be used for the frame rule
- This, in combination with separation logic, forms the mathematical base for Infer and it’s analysis

Analyzers provided

- Infer provides many analyzers and issues
- There are analyzers to detect memory issues, concurrency issues, compute complexity, program analysis
- Some analyzers use code annotations in their analysis and are only available for some of the languages
- Some analyzers are also unmaintained and/or deprecated

Annotation Reachability

- Checks if any `@Expensive` annotated function can be reached from any `@PerformanceCritical` function
- Annotations in C/C++ done using a JSON file which specifies which functions are expensive and which are critical
- Supported for C/C++/ObjC, Java, C#/.NET

InferBO

- InferBO (BO := buffer overrun) checks for buffer overflows and out of bounds accesses
- Supported for C/C++/ObjC, Java, C#/.NET

Runtime Complexity Analysis

- Computes the time complexity of functions
- Cost analysis statically estimates an upper bound on the worst-case execution cost of a program
- Computes the cost of instructions and how many times it is executed for each node in the CFG
- The total cost of the node is the product of these two
- Supported for C/C++/ObjC, Java, C#/.NET

Eradicate

- Check `@Nullable` annotations for Java and C#/.NET programs
- `@Nullable` annotation used for arguments which could be null and thus need to be handled accordingly
- Performs flow-sensitive analysis

Inefficient `keySet` Iterator

- Checks for inefficient uses of iterators that iterate on keys then lookup their values, instead of iterating on key-value pairs directly
- Supported for Java, C#/.NET
- Not working, even on the provided example in Infer docs

Liveness

- Detects dead stores
- Supported for C/C++/ObjC
- Pretty common feature in IDEs now

Loop Hoisting

- Detects opportunities to hoist function calls that are invariant outside of loop bodies for efficiency
- Uses purity analysis to determine function purity
- Supported for C/C++/ObjC, Java, C#/.NET
- No demo, since the loop hoisting seems to be broken
- Most likely because the purity analysis is experimental

Pulse

- Performs interprocedural memory safety analysis
- Only reports errors when all conditions on the erroneous path are true regardless of input (conservative analysis)
- Meant to replace original bi-abduction analyzer of Infer
- Supported for C/C++/ObjC, Java, C#/.NET
- Can detect
  - Constant address dereferences
  - Memory leaks
  - NULL dereferencing
  - Use-after-free and other lifetime memory issues

Purity

- Detects pure (i.e. side-effect-free) functions
- Performs inter-procedural analysis
- Requires alias-analysis, which is not implemented entirely correctly in Infer
- For now, they use InferBO’s alias-analysis
- Runtime complexity analysis and loop hoisting requires purity analysis
- Experimental for all the languages

RacerD

- Performs thread-safety analysis
- Analysis is (almost) sound, but not complete
- We can use annotations to help RacerD in analysis
- Again, because of poor alias-analysis, it misses race conditions for aliasing variables
- Supported for C/C++/ObjC, Java, C#/.NET

Starvation

- Detects various kinds of situations when no progress is being made because of concurrency errors
- Also detects deadlocks
- Supported for C/C++/ObjC, Java, C#/.NET

Uninitialized Value

- Warns when values are used before having been initialized
- Common feature in modern IDEs as well
- Supported for C/C++/ObjC

Deprecated but notable analyzers

- **Immutable Cast**: Detects object cast from immutable types to mutable types
- **AST Language (AL)**: Declarative linting framework over the Clang AST
- **`printf()` Argument Types**: Detect mismatches between the Java `printf` format strings and the argument types

Similar frameworks/tools

- SonarQube
- Coverity
- ReSharper
- CodeQL
- PVS-Studio
- Many of these are proprietary tools
- Commonly used in IDEs as extensions (PVS, ReSharper) and in CI/CD for finding defects (Coverity, CodeQL)

Questions & Answers