Software Analyzers

Tools for Program Understanding

How Ivette + Eva can help you C

Andre Maroneze, Frama-C/Eva Team

June 13, 2024 @ Frama-C Days

CEA-List, Université Paris-Saclay, Software Safety and Security Lab

Outline

Outline

Frama-C-ing code you didn't write yourself

- Target audience of this presentation: testers, QA, auditors, newly-arrived colleagues, and people who have not yet tried Ivette

- How can Frama-C help understand a program?
  - By abstracting hardware details (following the C standard)
  - By reasoning about it (thanks to ACSL)
  - By showing it differently (expliciting the implicit)
  - By offering tools and interactions (visitors, transformers, GUI)

ENSURING C SEES THE CODE THE WAY YOU SEE IT

Based on a true story! Do not try this at home.

```c
#include <stdint.h>
struct port {
  uint32_t flags;
} p;

void set_flag(uint8_t flag) {
  p.flags &= ~0xf8000000UL;
  p.flags |= flag << 24;
}

int main() {
  set_flag(42); // test 1
  set_flag(142); // test 2
}
```

Demonstration 1: `ivette shift-happens.c -eva`

Outline

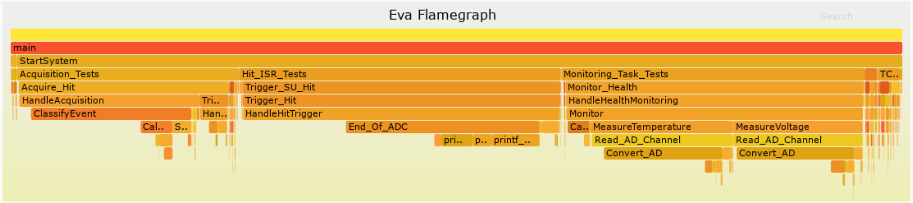
Not reinventing the wheel

Node.js: >2M packages in npm to choose from

- Several visualization tools already available

Example: Eva Flamegraphs for profiling the analysis time

- Currently: command-line script + browser
  - No live refresh
  - Cannot navigate to code
- Under development: Flamegraphs in Ivette!
  - Live update, click to navigate



Demonstration 2: Flamegraphs on Ivette

STUDIA: INTERPROCEDURAL DEPENDENCIES FOR EVA USERS

- Shows read/write locations impacting a given l-value
- Previously on Frama-C GUI: *Dependencies* $\rightarrow$ *Show defs*
- Main usage: find the origins of alarms and imprecise values

Demonstration 4: Studia in Papabench

Dive: dynamic dependence graphs

- Studia allows swimming through the code, one jump at a time
- Dive allows going deeper, faster, seeing ahead
- Graphical representation with lookahead

Demonstration 5: Dive in Papabench

Outline

Frama-C scripts: the command-line toolbox

- `frama-c-script command <options>`
- Motivations
  - Do something *before* parsing succeeds
  - Modify Frama-C's command line
  - Do things *around* C, but out of scope for Frama-C
- Dogfooding scripts on Open Source Case Studies
  - More experimental than Frama-C itself; moving fast and (sometimes) breaking things

Example: analysis template for Eva (under ongoing development)

- Example using `find-fun` and `make-wrapper`

Demonstration 6: Chrony

Outline

Frama-C: using AIs since 2008

Features not demoed here by lack of time (*live demo available!*):

- Pivot Tables: Ivette component for metrics and reporting
- fc-estimate-difficulty: “guesstimates” how hard it will be to analyze a given code (stand-alone binaries for Windows/macOS/Linux)
- frama-c-script creduce: minimizes and obfuscates crashing code for bug reports

Conclusion: Frama-C uses several kinds of AI to help program understanding

- Abstract Interpretation (Eva)
- Automatic-solver-based deductive Inference (WP)
- Advanced run-time Instrumentation (E-ACSL)
- Augmented Interface (Ivette)

FRAMA-C