Eva: Evolved Value Analysis

Frama-C's extensible
abstract interpreter

Frama-C

- An open-source platform for the analysis of C programs

  Visit http://frama-c.com

- Common kernel / AST / specification language (ACSL)
- Plus multiple analysis plugins:
  - EVA (abstract interpretation
  - WP (deductive verification)
  - Aoraï (temporal logic)
  - Mthread (Pthread like-concurrency)
  - E-ACSL (runtime monitoring)

EVA

- Frama-C's abstract interpreter
- Over-approximates all the possible behaviors of the program
  - Reports **alarms** when an undefined behavior may occur (as ACSL assertions)
  - ```
    /*@ assert Value: division_by_zero: v ≢ 0; */
    q = 1 / v;
    ```
- Extensible: new analysis domains may be plugged in
  - *This tutorial*
    ```

EVA in a nutshell

- Two key concepts:
  - Values
    - Abstraction of the values of expressions (integers, floating-point, pointers)
    - Forward and backward transformers for arithmetic and comparison operators (+, <, ==), etc
    - Also, memory locations (~ pointers)
    - See *src/plugins/value/values/abstract_values.mli*
  - States (= domains)
    - Abstraction of the entire memory
    - Forward transformers for assignments and conditionals
    - See *src/plugins/value/domains/abstract_domain.mli*
  - No direct communication between domains, but instead through values

This tutorial

- Extend two new domains:
  a. **Inout** domain (computation of memory locations input and outputs)
     - "Observational" domain: does not actively participate to the evaluation
     - **Goals**: extend the domain, make the computation more modular
     - See *src/plugins/value/domains/inout_instructions.mli*

  b. **Signs** domain (positive / negative / zero)
     - Improves the precision of evaluation (in some cases)
     - **Goals**: write abstract transformers for division and comparisons, collaborate on the emission of alarms
     - See *src/plugins/value/domains/signs_instructions.mli*

- Both parts are independent

First domain: Inout

- Showcase for a non-reduced product
  - Eva's main domain and Inout are computed simultaneously
  - Inout reuses some information made available by the main domain
  - But does not contribute back to the evaluation
- The domain mainly uses the precise pointer analysis offered by Eva
  - Good template for a flow-information based plugin
- Computation of memory locations read and written by functions

Second domain: signs

- Very classical and standard domain
  - Positivity, negativity and zero-ness of variables
- A good example of a new abstraction for values
  - Other interesting abstractions that could be added: strided intervals, bitwise abstraction, etc
- Uses one of EVA's available domains to represent the memory
  - Only integer variables
  - Other, more involved abstractions are available
  - Not in this tutorial
- **Goal**: extend the forward and backward transformers for signs

How to use Frama-C (for this tutorial)

- Use the virtual machine (Virtualbox required)
- **Or** compile Frama-C from source
  - *Not for the faint-hearted!...*
  - You need OCaml >= 4.02.3, OCamlfind and OCamlgraph already installed
    - Use/install opam
    - opam install depext merlin
    - opam depext frama-c
    - Install the packages listed as missing using your linux distribution
    - opam install zarith lablgtk conf-gtksourceview conf-gnomecanvas
  - git clone https://github.com/Frama-C/Frama-C-snapshot.git
  - git checkout tuto-POPL
  - autoconf ; ./configure ; make -j ; make merlin