A Tool for Integrating Abstract Interpretation, Model Checking, and Deductive Verification

Subash Shankar
City University of New York (CUNY)

Sep 19, 2014

Static Analysis Techniques

- **Abstract Interpretation**: An approximation of program semantics based on mappings between concrete and abstract lattices $\Rightarrow$ symbolic evaluation in abstract domain
  - ☹ Usefulness of [nondeterministic, lossy] abstract program dependent on abstractions
  - ☹ Loops require unrolling, with loss of precision (or an indeterminate fixed point computation)
- **Deductive Verification**: The formal semantics of a program, viewed as a predicate transformer from a postcondition to a precondition
  - ☹ Loops require the *manual* identification of a loop invariant
  - ☹ Automation limited by theorem prover limitations
- **Model Checking and CEGAR**: Iteration over abstraction-model checking-refinement cycle to automatically prove program correctness
  - ☹ State space explosion
  - ☹ Success limited by choice of predicate abstractions

Static Analysis Techniques

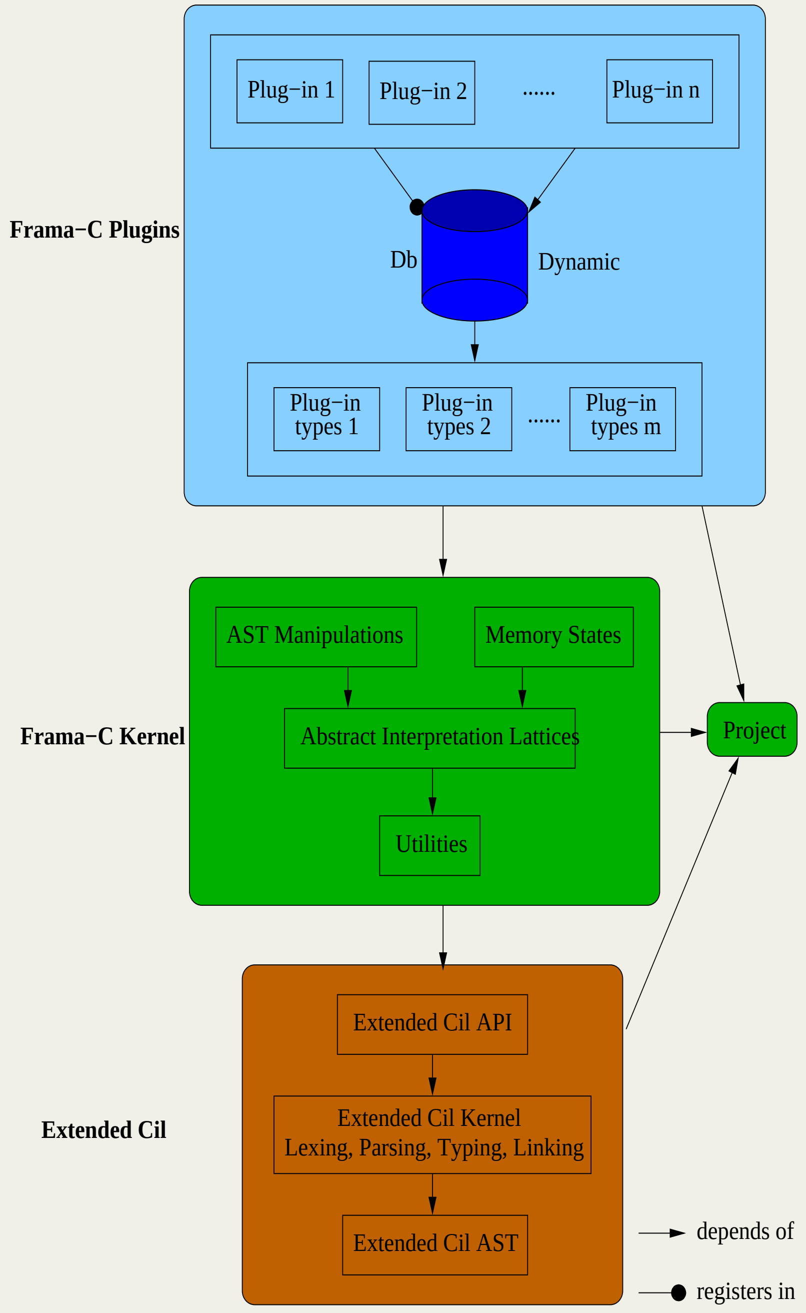
- **Abstract Interpretation**: An approximation of program semantics based on mappings between concrete and abstract lattices
  $\Rightarrow$ symbolic evaluation in abstract domain
  - ☹ Usefulness of [nondeterministic, lossy] abstract program dependent on abstractions
  - ☹ Loops require unrolling, with loss of precision (or an indeterminate fixed point computation)
- **Deductive Verification**: The formal semantics of a program, viewed as a predicate transformer from a postcondition to a precondition
  - ☹ Loops require the *manual* identification of a loop invariant
  - ☹ Automation limited by theorem prover limitations

} Frama-C

- **Model Checking and CEGAR**: Iteration over abstraction-model checking-refinement cycle to automatically prove program correctness
  - ☹ State space explosion
  - ☹ Success limited by choice of predicate abstractions

} SATABS

Frama-C Architecture



(From Frama-C Developer Manual)

Plugins:

- Interfaces to abstract syntax tree (AST), C intermediate language (CIL) extended with ANSI C Specification Language (ACSL) annotations, AI lattices, etc. provided by kernel
- Plugins used for either analysis ($\geq$ 1 AST) or source-to-source transformation ($>$ 1 AST)
- Statically-linked kernel-integrated plugins include value (abstract interpretation) and wp (weakest preconditions)
- Extensible through user-written plugins, typically linked dynamically
- Common plugin interface allows for information sharing, along with a central mechanism for combining plugin results.

- All programmed in OCAML

TOOL DEMO

Integrating AI, WP, and CMC (Ongoing and Future Research)

BUT individual analyzers often won't work on given examples . . .
⇒ Integrate analyses:

- **Loose coupling:**
  - Use core Frama-C to improve CMC results. Examples:
    1. Value analysis to filter initial states for model checking
    2. Frama-C to slice out irrelevant paths before CMC
    3. Use WP/AI to pick "good" initial abstractions?
  - Use CMC to improve deductive verification results.
- **Tight Coupling:** Develop a rigorous software analysis/verification mechanism and/or use cases that exploit the differing benefits of multiple analysis techniques.

Thank you for listening

And most of all, Thanks Ed!

Questions?

Frama-C: downloadable from `www.frama-c.com`

CMC Plugin: email `subash.shankar@hunter.cuny.edu`