Cegarmc

Integrating Software Model Checking into Frama-C

Artjom Plaunov

City University of New York

N. Kosmatov, A. Plaunov, S. Shankar, and J. Signoles, “Combining Analyses Within Frama-C,” in Guide to Software Verification with Frama-C, Springer, 2024.

Table of contents

1. (Very) Brief Overview of Model Checking
2. Software Model Checking
3. CegarMC

Model Checking

Model Checking

Model checking is an automated verification technique for finite state + concurrent systems

1981: CTL **Explicit** Model Checking - Independently developed by Clarke/Emerson and Sifakis/Quielle. (EMC model checker 1982)

Model Checking

Model checking is an automated verification technique for finite state + concurrent systems

1981: CTL **Explicit** Model Checking - Independently developed by Clarke/Emerson and Sifakis/Quielle. (EMC model checker 1982)
1990: Symbolic Model Checking (SMV in 1992)

Model Checking

Model checking is an automated verification technique for finite state + concurrent systems

1981: CTL **Explicit** Model Checking - Independently developed by Clarke/Emerson and Sifakis/Quielle. (EMC model checker 1982)
1990: Symbolic Model Checking (SMV in 1992)

1990s: Successful applications verifying hardware in industry

Model Checking

Model checking is an automated verification technique for finite state + concurrent systems

1981: CTL **Explicit** Model Checking - Independently developed by Clarke/Emerson and Sifakis/Quielle. (EMC model checker 1982)
1990: Symbolic Model Checking (SMV in 1992)

1990s: Successful applications verifying hardware in industry

1998: Bounded Model Checking using SAT

Model Checking

Model checking is an automated verification technique for finite state + concurrent systems

1981: CTL **Explicit** Model Checking - Independently developed by Clarke/Emerson and Sifakis/Quielle. (EMC model checker 1982)
1990: Symbolic Model Checking (SMV in 1992)

1990s: Successful applications verifying hardware in industry

1998: Bounded Model Checking using SAT

2000: Counterexample Guided Abstraction Refinement **(CEGAR)**

Model Checking

Model checking is an automated verification technique for finite state + concurrent systems

1981: CTL **Explicit** Model Checking - Independently developed by Clarke/Emerson and Sifakis/Quielle. (EMC model checker 1982)
1990: Symbolic Model Checking (SMV in 1992)

1990s: Successful applications verifying hardware in industry

1998: Bounded Model Checking using SAT

2000: Counterexample Guided Abstraction Refinement **(CEGAR)**

2000 - Present: Software Model Checking

Model Checking

Pros

- Turnkey Verification: Automatic
- Good with Finite + Control + Concurrent Structures
- Counterexample Generation (BMC, Bug Finding)

Cons

- Doesn't Scale (State Space Explosion)
- Limited Verification Complexity (e.g., no tricky loop invariants)
- Limited Expression of Properties (Assert Statements).

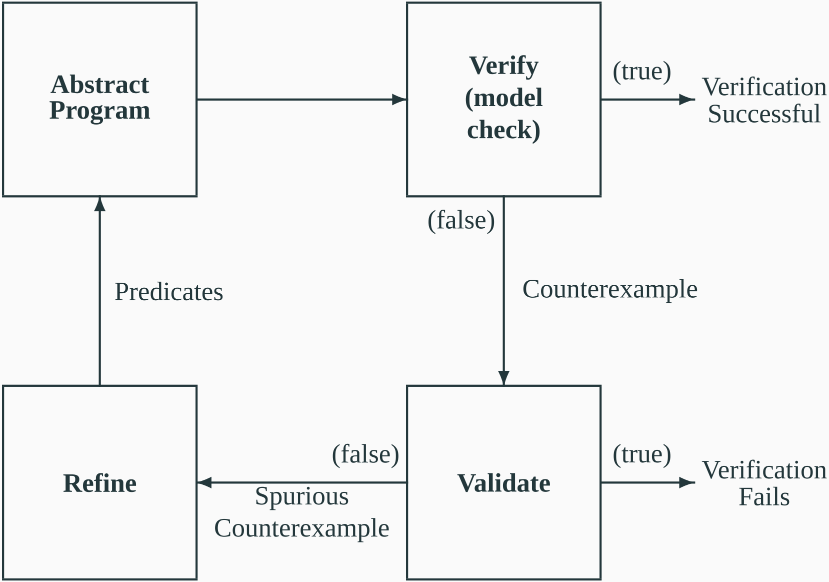
Software Model Checking + CEGAR

Software Model Checking

```
        ...
        // unlock phase
        if (p1 != 0) {
            __VERIFIER_assert(lk1 == 1);
            lk1 = 0;
        }

        if (p2 != 0) {
            __VERIFIER_assert(lk2 == 1);
            lk2 = 0;
        }
        ...
```

CEGAR


Abstract Program
Verify (model check)
(true)
Verification Successful
(false)
Counterexample
Predicates
Refine
(false)
Spurious Counterexample
Validate
(true)
Verification Fails

Example: CPAchecker

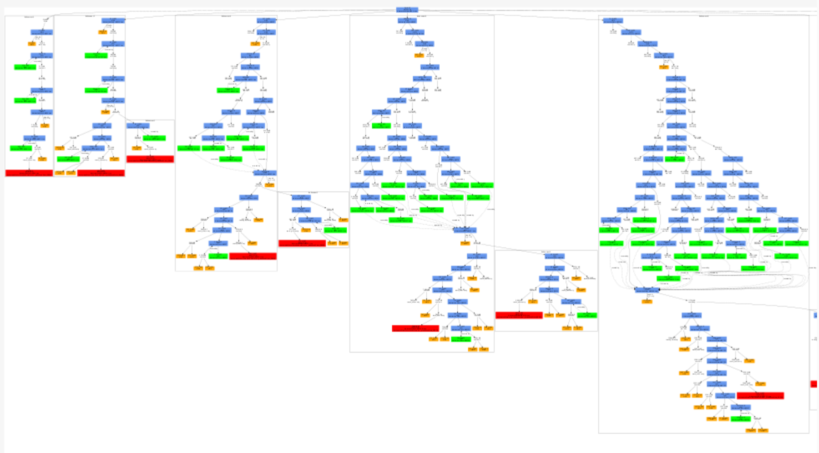


Figure 2: CPAchecker Abstraction Refinements

Cegarmc

CegarMC: Current Implementation

Verification Interface:

```
    /*@
        requires R1;
        ensures E1;
    */                       // Proved by WP
    int foo() {
        /*@
            requires R1;
            ensures E1;
        */                   // Proved by Cegarmc
        S1;
        /*@
            requires E1;
            ensures E2;
        */                   // Proved by WP
        S2;
    }
```

CegarMC: Current Implementation

Translate ACSL Statement Contracts into Reachability Problems:

```
    Declarations;
    __VERIFIER_assume(R1);
    <S1 Translation>
    __VERIFIER_assert(E1);
```

CegarMC Options

Context Flag: Use EVA analysis to provide context for statement contract.

```
Declarations;
__VERIFIER_assume(EVA ANALYSIS);
__VERIFIER_assume(R1);
<S1 Translation>
__VERIFIER_assert(E1);
```

CegarMC Options

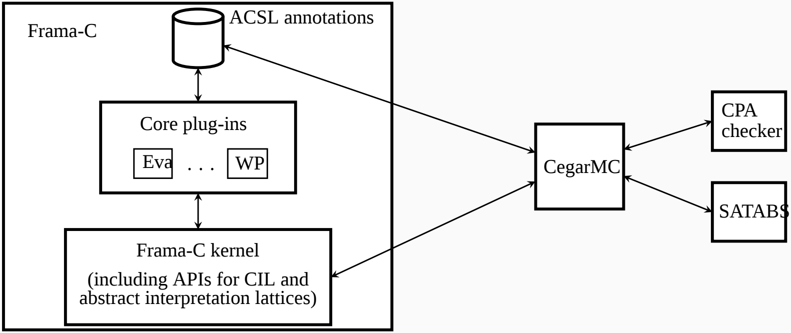
- **Abstract Calls Flag:** Use already verified function contracts to reduce model checking state space.

```

/*@ ensures \result == 1; */
int foo () {
    ...
}

int main() {
    int x;
    /*@ ensures E1; */
    {
        // Original code:
        x = foo();
        // Translation (Simplified):
        //__VERIFIER_assume(x == 1);
        // ...
        //__VERIFIER_assert(E1);
    }
}
```

Architecture + Demo



Aside: Model Checking vs. Abstract Interpretation

Why use model checking when we have Abstract Interpretation?

- Abstract Interpretation is much more scalable.
- Model Checking is not scalable.
- Model Checking has CEGAR, non-monadic properties, and can be more path-sensitive.
- **Caveat** Distinction is blurring: 1) CPAchecker, 2) CEGAR for Abstract Interpretation.

(Bruni Roberto, Giacobazzi Roberto, Gori Roberta, and Ranzato Francesco. 2022. Abstract interpretation repair.)

Current/Future Work

- Port CegarMC to most recent Frama-C version
- Bug Finding: Bounded Model Checking
- Extend pointer support
- Use EVA to improve model checking
- Feedback - what would you like to see in this tool?