DOI https://doi.org/10.30525/978-9934-26-597-6-9

AUTOMATION OF PROVING THE PROGRAMS CORRECTNESS IN THE FRAMA-C SYSTEM

Yevhen Holovko*[1], Viktors Gopejenko[1,2]
[1]ISMA University of Applied Sciences, Latvia, [2]Ventspils University of Applied Sciences, Latvia
**Corresponding author's e-mail: e.golovko@outlook.com*

Abstract

The automation of program correctness verification is a crucial aspect of modern software development, especially for systems where reliability and security are paramount. This research investigates the use of Frama-C, a framework for deductive verification of C programs, implementing formal methods to improve software correctness. The paper presents the integration of weakest precondition (WP) calculus and ACSL (ANSI/ISO C Specification Language) annotations, demonstrating how Frama-C's WP plugin facilitates automated verification, using external theorem provers like Alt-Ergo, CVC4, Z3, and Yices. By formalizing program properties, this work explores how formal methods can ensure the correctness of C programs, as because traditional testing approaches have limitations so are not preferrable in high-reliable or secure systems. The findings underscore Frama-C's potential in developing high-assurance software.

Keywords: Program correctness, formal methods, Frama-C, weakest precondition, automated verification

1 Introduction

The increasing complexity of software systems and their integration into critical applications – such as healthcare, aviation, and automotive industries – has made software correctness a crucial factor. Traditional testing approaches are unable to *guarantee* that a program is error-free under all conditions, they can only validate a program for a subset of possible inputs. Which is not acceptable particularly in mission-critical applications where even a single error can lead to catastrophic consequences. As a result, formal methods, specifically automated program correctness verification, are the best option.

Frama-C is a modern framework that leverages formal verification to ensure the correctness of C programs. By using weakest precondition (WP) calculus and ACSL annotations, Frama-C offers a way to mathematically

prove that programs *meet their specifications*. This paper explores the capabilities of Frama-C in automating the verification of C programs, highlighting its strengths in comparison to traditional testing methods.

2 Overview

The need for automated verification arises from the limitations of manual testing. Manual or auto-testing, while useful in identifying errors, does not provide the mathematical guarantee that a program behaves correctly for all possible execution paths. Deductive verification, on the other hand, allows for a formal proof of correctness that covers all possible execution scenarios.

Frama-C, a tool designed for deductive verification of C programs, enables such formal verification. This framework integrates the weakest precondition (WP) calculus with ACSL annotations. ACSL allows developers to define preconditions, postconditions, and other program properties in a formal way. By annotating C code with ACSL specifications, developers can describe the intended behavior of the program, which Frama-C then uses to generate proof obligations. These obligations are then validated using theorem provers, which determine whether the program satisfies the specified properties. The main idea here is to prove that *the conditions under which a program's execution results in a desired postcondition*.

```
//@ ensures \result == ((a < b) ? b : a);
int maxab (int a, int b)
{
 return (a < b) ? b : a;
}
```

LISTING 1: Source Code of the maxAB.c Program with ACSL Contract

By using Frama-C's WP plugin, developers can automatically generate verification conditions, which can be proven using external theorem provers such as Alt-Ergo, CVC4, Z3, and Yices.

3 Research Methodology

This paper focuses on automating the proof of program correctness using Frama-C and its WP plugin. The research methodology includes the following steps:

1. *ACSL Annotation*: C programs are annotated with ACSL specifications that define the program's behavior, including preconditions, postconditions, and invariants.

2. *Verification*: The verification conditions are generated using Frama-C's WP plugin. These conditions are then passed to external theorem provers for proof validation.

3. *Experimentation*: Various C programs are tested using Frama-C, and the results of the verification process are analyzed to evaluate the effectiveness of the tool.

4 Decision

The experiments demonstrate that Frama-C is capable of automatically proving the correctness of C programs that are annotated with ACSL specifications. For simple programs, the WP plugin successfully generates verification conditions, which are then proven by the theorem provers. However, the complexity of the programs and the annotations required can impact the efficiency and scalability of the verification process.

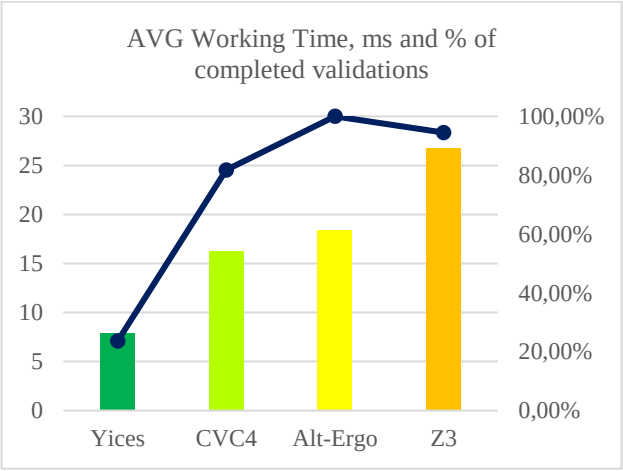

Figure 1. Average working time and percentage of completed validations per prover

The integration of multiple theorem provers increases the reliability of the verification process, as different provers excel at solving different types of verification conditions. Alt-Ergo and Z3 were the most effective at proving the generated goals, with Z3 performing better on complex logical formulas, while Alt-Ergo was more efficient in terms of runtime.

5. Conclusion

This research demonstrates how to automate program correctness verification using Frama-C instead of manual or auto-testing. Using formal methods such as weakest precondition calculus and ACSL annotations,

Frama-C allows developers to formally prove that a program meets its specifications, providing strong guarantees, unlike traditional testing approaches.

However, adopting formal methods still presents challenges, especially when it comes to creating complex ACSL annotation. Future research could focus on simplifying the process of annotating programs and integrating Frama-C more seamlessly into standard software development workflows.

References

[1] Dijkstra, E. (1976). A Discipline of Programming. Prentice-Hall.

[2] Gries, D. (1987). The Science of Programming. Springer-Verlag.

[3] Cuoq, P., Filliâtre, J.-C., Marché, C., et al. (2012). Frama-C: A Tool for Static Analysis of C Programs. Formal Methods and Software Engineering, 423-434.

[4] Hoare, C. A. R. (1969). An Axiomatic Basis for Computer Programming. Communications of the ACM, 12(10), 576-580.

[5] Dijkstra, E. (1976). The Art of Programming. Addison-Wesley.

[6] Alt-Ergo. (2025). About Alt-Ergo. Available at: https://alt-ergo.ocamlpro.com

[7] Frama-C. (2025). Frama-C User Manual. Available at: https://www.frama-c.com

Authors

Yevhen Holovko, 1989, Dnipro, Ukraine.
Current position, grades: master student at ISMA University.
Scientific interest: theorem proving, weakest precondition, formal verification.

Viktors Gopejenko, Riga, Latvia
Current position, grades: Professor, Dr. sc. ing., Vice President for Research at ISMA University of Applied Sciences, Director of Study Programme Computer Systems (MSc)
Scientific interest: Computer Modeling, System Dynamic