VeriTest: Automatic Verification of Compiler Optimizations

VeriTest aims to provide feedback to GraalVM developers on their proposed optimizations and integrate it into their development workflow. This would make it easy for the developers to use VeriTest as you go, without being a "proof expert".

Achmad Afriza Wibawa
achmad.afriza@gmail.com

School of Electrical Engineering and Computer Science,
The University of Queensland

Supervisors
- Associate Professor Mark Utting
- Emeritus Professor Ian J. Hayes
- Brae J. Webb

01 Introduction

We introduce VeriTest: a software interface for automatic verification of GraalVM's [1] expression optimizations.

The ideal workflow is:

1. GraalVM developers write a proposed optimization rule in Veriopt's Domain Specific Language (DSL) [2].
2. The rule would be passed into VeriTest to:
   a. Find fast, obvious, feedback — by leveraging Isabelle [3].
   b. If it can't find any, it would be passed to "proof experts".

02 Isabelle

Isabelle is an interactive theorem prover that utilizes syntax translations of a theory definition into a set of inference rules that can be automatically reasoned within the system [3].

We focus on:
- Sledgehammer [3, Sec. 3] — to automatically verify rules.
- Nitpick [3, Sec. 4] and Quickcheck [3, Sec. 5] — to find counterexamples.

03 Veriopt's DSL

Veriopt's DSL [2] enables GraalVM developers to express optimizations as rewriting rules in a Java syntax similar to GraalVM's IR [4] within Isabelle.

```
optimization InverseLeftSub:
  (x - y) + y ↦ x
 1.trm(x) < trm(BinaryExpr BinAdd (BinaryExpr BinSub x y) y)
 2.BinaryExpr BinAdd (BinaryExpr BinSub x y) y ⊒ x
```

This optimization rule describes that 2 proof obligations must be met for the side-effect-free optimization to be correct:

1. Proof that the optimization rule will **terminate**;
2. Proof that each pass of the optimization rule will result in a **refinement** of the expression [2, Sec. 3].

04 Methodology

VeriTest utilizes Isabelle Client-Server [5] interactions as a **method of interfacing** with Isabelle tools.

Isabelle Server acts as the core Isabelle process that allows theorems and all the required facts (i.e., Veriopt's theory base) to be loaded up and **processed in parallel** [5, Ch. 4.2.6].

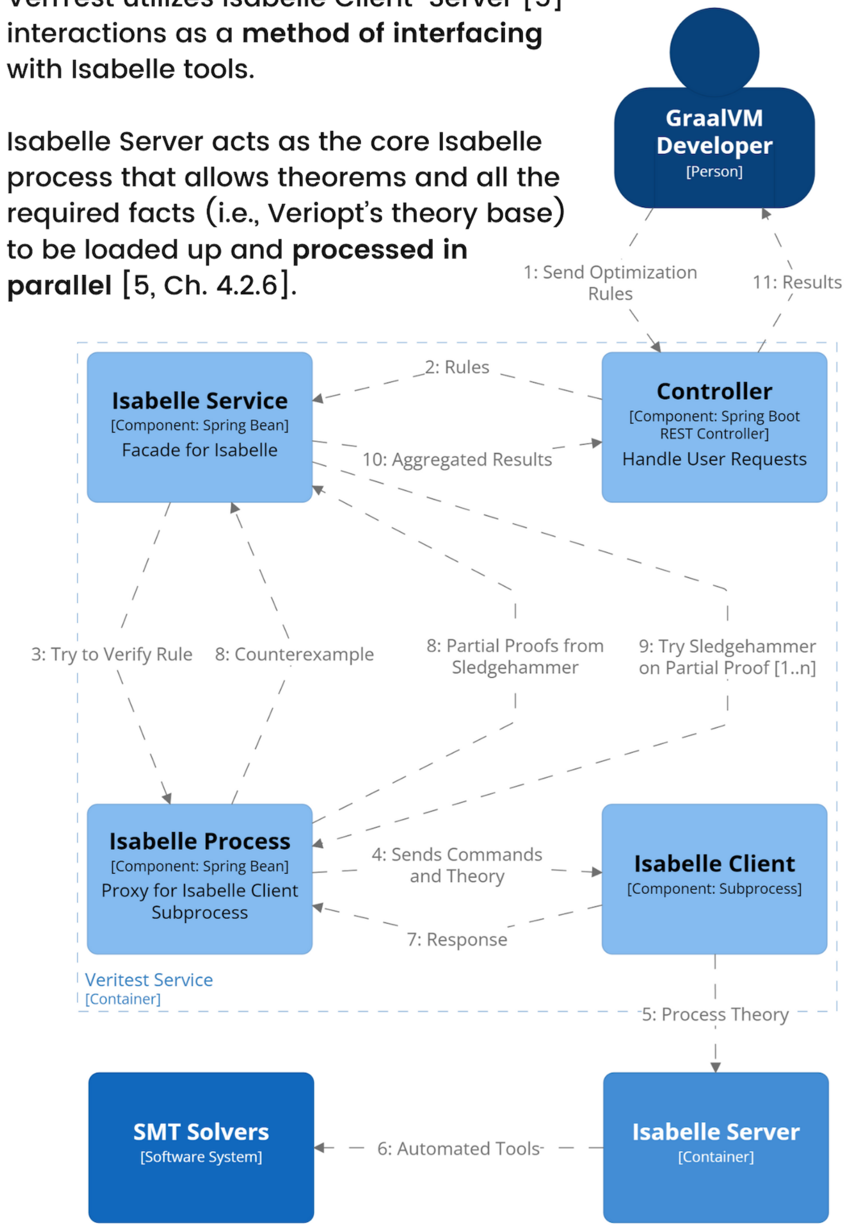

Figure 1. Sequence Diagram on Verifying Optimization Rules

05 Verifying InverseLeftSub

Request:

```
{
  "request_id":"InverseLeftSub",
  "theory":"(x - y) + y ↦ x"
}
```

⬇

Response:

```
{
  "request_id": "InverseLeftSub",
  "status": "FOUND_PROOF",
  "proofs": [
    "using RedundantSubAdd_Exp by blast"
  ]
}
```

06 Results

Result	# Rules	Mean ± SD
Failed	59	87.06 ± 49.42
Found Auto Proof	7	37.73 ± 3.92
Found Proof	32	82.91 ± 29.43
Found Counterexample	1	40.79 ± 4.02
Malformed	13	37.96 ± 4.02
No Subgoal	2	37.93 ± 3.38

Figure 2. Evaluation of each existing Veriopt optimization rules based on runtime (in seconds)

07 Summary

- VeriTest yields a **comprehensive analysis** of rules by utilizing a battle-tested automated theorem prover such as Isabelle.
- The abstraction of interacting with Isabelle allows VeriTest to provide fast feedback by **parallel processing** of optimization rules.
- The evaluation demonstrates the capability of **integrating existing lemmas** inside Veriopt's theory base to find proofs.
- Results suggest that the percentage of optimization rules that can be automatically proven will **improve** as the collection of theories grow.

References

[1] Oracle, "GraalVM: Run programs faster anywhere," 2020. [Online]. Available: https://github.com/oracle/graal
[2] B. J. Webb, I. J. Hayes, and M. Utting, "Verifying term graph optimizations using Isabelle/HOL," in Proceedings of the 12th ACM SIGPLAN International Conference on Certified Programs and Proofs, 2023, p. 320333.
[3] J. C. Blanchette, L. Bulwahn, and T. Nipkow, "Automatic Proof and Disproof in Isabelle/HOL," in Frontiers of Combining Systems, ser. LNCS, C. Tinelli and V. Sofronie-Stokkermans, Eds., 2011, vol. 6989, pp. 12–27.
[4] B. J. Webb, M. Utting, and I. J. Hayes, "A formal semantics of the GraalVM intermediate representation," in Automated Technology for Verification and Analysis, ser. LNCS, Z. Hou and V. Ganesh, Eds., vol. 12971, Oct. 2021, pp. 111–126.
[5] M. Wenzel, "The Isabelle System Manual." [Online]. Available: https://isabelle.in.tum.de/dist/Isabelle2023/doc/system.pdf