PURDUE UNIVERSITY

Elmore Family School of Electrical and Computer Engineering

Optimizing Gradual C0 Program Verifier

Wen Fan (fan372@purdue.edu)

Advisor: Dr. Jenna DiVincenzo

OVERVIEW

Prior work on Gradual C0 [2] showed preliminary evidence of bottlenecks in the static verification performance of Gradual C0. In this work, we investigate bottlenecks further, and implement parallelism as an optimization motivated by our investigation. The evaluation shows performance is improved with parallelism.

1. Background

Gradual C0 (GC0) [1] is a gradual program verifier for heap data structures for the C0 language. GC0 combines static and run-time verification.

GC0's static verifier uses symbolic execution to verify each function. Thus, it forks execution at each branch point in a function.

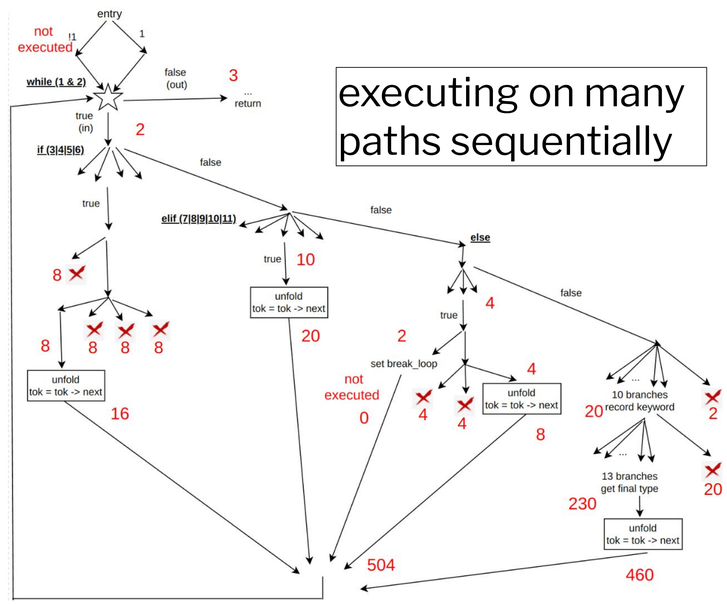
The static verification of GC0 suffers from slowness on a comparatively large program.

- e.g., ~1 min 20s on cparser.c0 [2] with ~3000 LOC.

2. Analysis

By analyzing the verification procedure on cparser.c0, we find that over 97% of the time is spent on function declspec, where

- there are 571 paths in total,
- paths are executed sequentially,
- > 73% of paths are executed within 0.6 s.



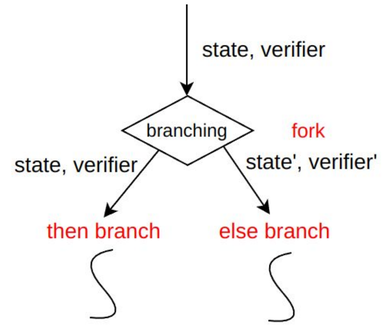
Function declspec in cparser.c0, with 571 paths in symbolic execution (red number shows the number of paths executed on the basic block)

3. Design

adding parallelism!

At a branching point, copy the state and verifier, then execute two branches in parallel.

- need to maintain the information (e.g., declaration and assumption) in the verifier,
- follow the parallel design of Viper [3], but garbage-collect such info to reduce the overhead.

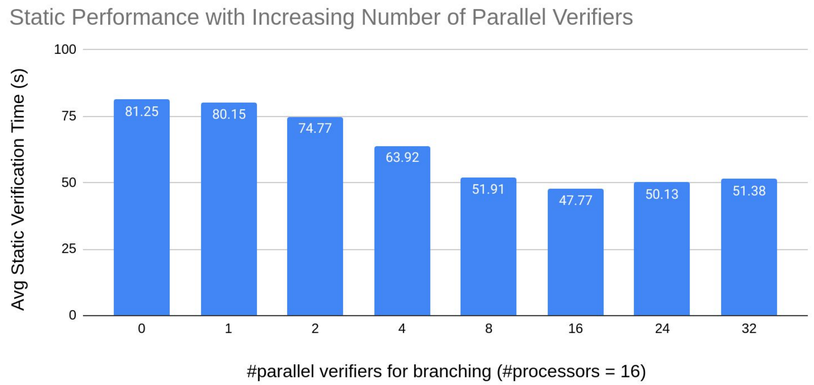


Overview of enabling parallelism at the branching point

4. Evaluation

other bottlenecks?

Static Verification time reduces as more parallelism is enabled, but it doesn't scale well with the number of parallel verifiers.



Reference:

[1] DiVincenzo, Jenna, et al. "Gradual C0: Symbolic Execution for Gradual Verification." *arXiv preprint arXiv:2210.02428* (2022).

[2] DiVincenzo, Jenna Wise. Gradual Verification of Recursive Heap Data Structures. Diss. Carnegie Mellon University, 2023.

[3] Schwerhoff, Malte H. Advancing automated, permission-based program verification using symbolic execution. Diss. ETH Zurich, 2016.