Research in Computer Science - Project Proposal

Integrating dynamic test generation with sound verification

Patrick Emmisberger

November 21, 2014

1 Abstract

Dafny is a programming language with built-in specification constructs to support sound static verification. The Dafny program verifier is powered by Boogie and Z3 and targets functional correctness of programs. Delfy is an automatic test generation tool based on dynamic symbolic execution for the Dafny language. This project explores the deep integration of sound verification, as implemented in the Dafny verifier, with dynamic test generation, as implemented in Delfy.

2 Problem statement

The goal of this project is to use a combination of the existing verifier and dynamic symbolic execution engine to improve the feedback on failing assertions and verify as many properties as possible. This combination will be available to the user through an extension for Visual Studio, which we call wizard, that simplifies development with the Dafny language. The following core problems should be addressed:

- Run the verification and dynamic symbolic execution in parallel in the background to improve the feedback on assertions by combining the results of both analyses. In particular, if the dynamic symbolic execution detects a failing assertion, then we definitely know that the assertion cannot be verified. If the verifier can verify a particular piece of code, there is no need to run dynamic symbolic execution on this code. In case we can fully explore straight-line code with Delfy but the Dafny verifier reports a verification error, then we have proved that this error is a false positive.
- Find ways to determine the tool more suitable for verifying a piece of code.

- Prioritize assertions for the user to provide more specifications. This will be done based on the level of coverage that can be achieved by dynamic symbolic execution. For example, we mark an assertion that is not fully explored by the dynamic symbolic execution with a high priority, while an assertion in fully explorable code has a low priority.

The following suggestions could be implemented as an extension to the core part:

- If the dynamic symbolic execution is able to prove properties about a method (e.g., by an exhaustive path search of straight-line code), add these properties as an assumption to strengthen the verification.
- Improve the performance of the dynamic symbolic execution engine to provide feedback more rapidly and extract the concrete execution from the Visual Studio process to prevent IDE crashes by faulty test code.
- Learn specifications, especially about sets, and implement a feedback loop from Delfy to Dafny.

3 Project definition

The project is subdivided into the following parts:

3.1 Conceptual phase

Goal: Developing ideas on how to approach the main problems. If appropriate, build a small proof of concept for some of the ideas.

3.2 Implementation phase

Goal: Implementation of the wizard which will be integrated into Visual Studio.

3.3 Learning phase

Goal: Implementation of a feedback channel from Delfy to Dafny.

3.4 Writing phase

Goal: Documentation of the approach, implementation and evaluation results.