A Boogie Case Study: Project Description

Benjamin Lutz

April 18, 2006

This is the project description for a semester thesis I plan to do in the summer semester 2006 at the ETH Zurich under the guidance of Prof. Peter Müller. Is about gaining insight in modern research in the areas of code verification and correctness. I will use a part of the Mono libraries for a Boogie case study. The secondary goals are finding possible bugs in Boogie and Mono.

Spec# [2] is an extension of the C# programming language. It extends C# by offering contracts (method pre- and postconditions and object invariants), non-nullable types and checked exceptions. It comes with a static program verifier, *Boogie*, that generates logical verification conditions from a Spec# program. These are run through an theorem prover that analyzes these verification conditions to prove the correctness of a program.

Mono [1] is an open-source implementation of the .NET platform that allows .NET applications to run on various non-Windows platforms (it does support Windows too, however). It has been selected for this case study because it offers a large amount of free source code.

First, a selection of the mono code will be made. I expect to be assigned a part of the Mono .NET framework libraries. I will then look at this library part class by class, and extend it with contracts and ownership relations. In other words, I will port the C# code to Spec# and analyze the code with Boogie.

At this point, several outcomes are imaginable. One, everything works fine, in which case I'll manage to verify a large part of Mono's .NET framework implementation. Two, I find verification problems that can be attributed to errors in the Mono code. I will investigate such problems and, hopefully, correct them and have my modifications accepted by the Mono Project and imported into Mono. Three, I find problems in Boogie. I will analyze such problems, if necessary by investigating in the Boogie source code.

My personal goal in this project is to gain a deeper understanding and experience with tools for enhancing software correctness. Spec# seems to be on the forefront of developments in this area, and a nice continuation of the contract idea in Bertrand Meyer's Object-Oriented Software Construction [3]. I'm looking forward to finding out what the implications of using Spec# in practice are, and what advantages can be realized.

References

[1] The Mono Project. *Mono*. http://www.mono-project.com

[2] Microsoft Research. *Spec#*. http://research.microsoft.com/specsharp/

[3] Bertrand Meyer. *Object-Oriented Software Construction*. 2nd edition. Upper Saddle River, NJ: Prentice-Hall, 1997