Chair of Programming Methodology
Master's Thesis

Generating BoogiePL for Scala

Valentin Wüstholz

Supervisors: Prof. Peter Müller, Dr. Yannis Kassios

March 23, 2009

Introduction

Scala [1, 2] is an multi-paradigm programming language started in 2001 at Ecole Polytechnique Fédérale de Lausanne (EPFL) by Martin Odersky. It tries to combine the respective strengths of (purely) object-oriented and functional programming. Some notable features are its strong static type system, type inference and support for generics, higher-order functions, pattern matching and traits. Its main implementation also provides seamless integration with the Java programming language by targeting the Java Virtual Machine.

BoogiePL is a procedural language for checking object-oriented programs. It was developed at Microsoft Research as part of their Spec# programming system [3, 4]. The Spec# programming language is a superset of the C# programming language with support for design by contract. One major component of the Spec# programming system is Boogie [5], a modular reusable verifier for object-oriented programs. It is reusable in the sense that it can also be used to statically verify programs in other programming languages by translating them to BoogiePL [6] first. Several programming languages already make use of this feature.

Goals

The main goal of this project is to design and implement a tool that translates Scala programs to BoogiePL. Since Scala offers quite a few features that separate it from mainstream languages like C# and Java, we will focus on these features. But obviously, we will still have to implement a basic subset of the Scala language first. For now, we will only deal with sequential and (pure) Scala programs.

Minimal Requirements

The minimal requirement for this project is to define and implement the transformation from a useful subset of the Scala language to BoogiePL code. This subset includes basic language features such as assignments, loops, conditionals and methods, as well as some of the more interesting features such as traits, singleton objects, higher-order functions and closures.

Another important requirement is that we want to use good software engineering practices to ensure that new features can be added easily and existing ones could be changed if necessary. The implementation should therefore be well documented and tested.

Possible Extensions

These are a few straightforward extensions we could come up with so far:

- Concurrency: Scala supports quite a few concurrency models. Adding support for one or more of them would be an interesting extension.
- More language features: e.g. pattern matching, generators, by-name parameters, exceptions

Project Plan

Activity	Weeks
Learn about Scala's basic features and Boogie/BoogiePL; design a memory model	3
Model Scala's basic features	2
Write down the results	1
Implementation of the required features	9
Write down the results	2
Design and implement more features (→ possible extensions)	4
Complete the thesis	2
Prepare the presentation	1
Total	24

References

[1] `http://www.scala-lang.org/`.

[2] Martin Odersky. *The Scala Language Specification 2.7*. École polytechnique fédérale de Lausanne, Switzerland, January 2009.

[3] `http://research.microsoft.com/en-us/projects/specsharp/`.

[4] Mike Barnett, K. Rustan M. Leino, and Wolfram Schulte. The Spec# programming system: An overview. In Gilles Barthe, Lilian Burdy, Marieke Huisman, Jean-Louis Lanet, and Traian Muntean, editors, *Construction and Analysis of Safe, Secure, and Interoperable Smart devices (CASSIS 2004)*, volume 3362 of *LNCS*, pages 49–69. SV, 2005.

[5] Mike Barnett, Bor-Yuh Evan Chang, Robert DeLine, Bart Jacobs, and K. Rustan M. Leino. Boogie: A modular reusable verifier for object-oriented programs. In *Formal Methods for Components and Objects (FMCO) 2005, Revised Lectures*, volume 4111 of *LNCS*, pages 364–387. SV, 2006.

[6] Rob DeLine and K. Rustan M. Leino. BoogiePL: A typed procedural language for checking object-oriented programs. Technical Report MSR-TR-2005-70, Microsoft Research, 2005.