



NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES

DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

POSTGRADUATE STUDIES PROGRAM

MASTER THESIS

Faster Scala Collections with Macros

Georgios Kollias

Supervisor: Yannis Smaragdakis, Associate Professor NKUA

ATHENS

MAY 2013



ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

**Γρηγορότερες Δομές Δεδομένων στη Scala
με χρήση Macros**

Γεώργιος Κόλλιας

Επιβλέπων: Γιάννης Σμαραγδάκης, Αναπληρωτής Καθηγητής ΕΚΠΑ

ΑΘΗΝΑ

ΜΑΙΟΣ 2013

MASTER THESIS

Faster Scala Collections with Macros

Georgios Kollias

RN: M1049

SUPERVISOR:

Yannis Smaragdakis, Associate Professor NKUA

THESIS COMMITTEE:

Yannis Smaragdakis, Associate Professor NKUA

Panos Rondogiannis, Associate Professor NKUA

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

**Γρηγορότερες Δομές Δεδομένων στη Scala
με χρήση Macros**

Γεώργιος Κόλλιας

ΑΜ: M1049

ΕΠΙΒΛΕΠΩΝ :

Γιάννης Σμαραγδάκης, Αναπληρωτής Καθηγητής ΕΚΠΑ

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ:

Γιάννης Σμαραγδάκης, Αναπληρωτής Καθηγητής ΕΚΠΑ

Παναγιώτης Ροντογιάννης, Αναπληρωτής Καθηγητής ΕΚΠΑ

Περίληψη

ΦιΞμε
Φαταλ:
Ρεπλασε
με

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: ...

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: ...

Abstract

...

SUBJECT AREA: ...

KEYWORDS: ...

Acknowledgements

Fixme
Fatal:
Add me

Contents

1	Introduction	12
2	Background	13
3	Over-Approximating Escaped Objects	14
4	Safe Publication	15
5	Experimental Results	16
6	Related Work	17
7	Conclusions	18
	Acronyms and Abbreviations	19
	Appendices	20
A	Escape Analysis Code	20

List of Figures

List of Tables

Preface

fixme
Fatal:
Replace
me

Chapter 1

Introduction

This work describes the implementation of specific Scala collections operations using its recent compile-time metaprogramming capabilities.

Scala [cite] is a relatively new statically typed programming language that tries to unify the object-oriented and functional programming paradigms into one coherent paradigm, recently called object-functional.

Scala version 2.10, released on [date], introduced a new reflection subsystem adding both compile and run time metaprogramming capabilities. The new run-time reflection is much more general and feature complete compared to the Java's reflection. Compile-time reflection is quite rare and, currently, it can be found only in more exotic functional languages like Haskell [cite Template Haskell] and ML [cite Meta-ML]. Compile-time reflection enabled the introduction of an experimental version of type-safe macros which are mostly known in the dynamic functional programming community and especially the Lisp community.

Compile-time reflection capabilities are

That system enabled the introduction of an experimental version of
whose main implementation runs on the JVM [cite].

The primary motivation for this work was to check if macros can help us create faster collections by inlining operations at the call site. The project is based on Paul Phillips's declosurify project (github.com/paulp/declosurify) and modifies it to make the functionality available at the standard Scala library level, so that all operations implemented with macros can be used on plain Scala collection types (e.g., List, Array, etc.) without the need of creating new specialized types. Wherever the macro expansion is not feasible or appropriate we can fallback to the default `normal` implementation. Also, the use of macros here is not typical since they are used from inside the Scala Library where no macro detection/expansion functionalities are directly available (the Scala library doesn't depend on the Scala compiler or Scala reflect packages). The results are encouraging since initial ScalaMeter benchmarks show a 30% speedup. You can check the project's progress here github.com/geo-kollias/scala/tree/declosurify.

Fixme
Warning:
explain
the
concept
in the
back-
ground
chapter

Chapter 2

Background

Fixme
Fatal:
Add me

Chapter 3

Over-Approximating Escaped Objects

Fixme
Fatal:
Replace
me

Chapter 4

Safe Publication

Fixme
Fatal:
Replace
me

Chapter 5

Experimental Results

Fixme
Fatal:
Replace
me

Chapter 6

Related Work

Fixme
Fatal:
Replace
me

Chapter 7

Conclusions

Fixme
Fatal:
Replace
me

Acronyms and Abbreviations

Abbreviation	Full Name
--------------	-----------

Appendix A

Escape Analysis Code

Fixme
Fatal:
Add me