The Practical Guide to Levitation

Ahmad Salim Al-Sibahi

Advisors:
Dr. Peter Sestoft
& David R. Christiansen
Submitted: September 1, 2014

IT University
of Copenhagen

Abstract

Goal: Implementation of levitation in a realistic setting, with practical performance benefits.

Contents

Chapter 1

Prologue

1.1 Introduction

1.2 Problem Definition

Chapter 2

Generic Programming

2.1 The Generic Structure of Inductive Data Types

How Generic Programming generally works.

2.2 The Importance of Genericity in Dependently-typed Languages

The similarity of structure and various slightly-different indexing of types.

2.3 The (Mostly) Gentle Art of Levitation

The elegance of a complete theorem for both ordinary and generic programming. Highlighting of possible issues with performance.

Chapter 3

Partial Evaluation

3.1 Functions and Constant Inputs

General introduction about partial evaluation.

3.2 Binding-time Analyses of Programs

Finding the relevant constant parts of the program.

3.3 Specialisation as a Form of Optimization

Performance benefits of program specialisation. Pitfalls.

Chapter 4

Levitating Idris

4.1 A Pragmatic Implementation of Levitation

How the general concept of levitation was transferred to Idris.

4.2 Data Type Synthesis from Descriptions

How levitated descriptions get transformed to ordinary data-types.

4.3 Static Initialization of Generic Functions

How algorithms that are dependent on the generic structure of a data-type are optimized. Discuss benefits of having a JIT/Profiling information for future work.

Chapter 5

Practical Examples

5.1 Generic Deriving

Examples of generic deriving of algorithms like decidable equality, pretty printing and possibly eliminators via generic structure.

5.2 Uniplate for Idris

A version of the Uniplate library for Idris based on `http://community.haskell.org/~ndm/uniplate/` *and* `http://www-ps.informatik.uni-kiel.de/~sebf/projects/traversal.html` *. This is useful for traversing structures in a generic fashion and especially when dealing with small changes in large data structures (such as compiler ADTs)*

Chapter 6

Epilogue