Title
McMaster University

Jason Balaci

Submission Date

Abstract

todo

Acknowledgements

todo

Contents

1 Introduction **5**
- 1.1 Context: Knowledge Capture & Encoding 5
- 1.2 Problem Statement 5
- 1.3 Thesis Outline 6

2 Background **7**
- 2.1 Project Focus & Goals 7
- 2.2 In practice/Architecture 7
  - 2.2.1 Chunks 7
  - 2.2.2 ChunkDB? 7
- 2.3 Methodology for locating and encoding knowledge – e.g., "bottom-up" approach 7
- 2.4 State of Drasil 8
  - 2.4.1 Short-term problems – leading into topics 8

3 Topic #1: Typing the Expression Language **9**
- 3.1 Background: Problem 9
- 3.2 Requirements & properties of a good solution 9
- 3.3 Solution 9
  - 3.3.1 Expression encodings discussion (GADTs, TTF) 9
  - 3.3.2 Dividing expression languages (CodeExpr, Expr, ModelExpr, Literals) 9
- 3.4 Continued problem with Expressions – leading into ModelKinds 10

4 Topic #2: ModelKinds – Theory types / discrimination – "Expressions in context" **11**
- 4.1 Problem 11
- 4.2 Requirements & properties of a good solution 11

4.3 Solution – ModelKinds 11
4.3.1 EquationalModels 11
4.3.2 EquationalRealms 12
4.3.3 EquationalConstraints 12
4.3.4 DEModel 12
4.3.5 Continued 12

5 Future Work **13**

6 Conclusion **14**

A Appendix **15**

Chapter 1

Introduction

1.1 Context: Knowledge Capture & Encoding

todo

TODO: While my MSc focus isn't necessarily exactly "When Capturing Knowledge Improves Productivity", I should have a reasonable amount of discussion here about it, or else readers might question the need for my work entirely. I guess I would also need a bit of justification for that programming ideology as well.

1.2 Problem Statement

As a domain expert transcribing knowledge encodings of some wellunderstood domain, one will largely be discussing the ways in which pieces of knowledge are *constructed* and *relate to each other*. In order for this abstract knowledge encodings to be usable in *some way*, it is vital to have "names" (*types*) for the knowledge encodings. In working to capture the working knowledge of a domain, it's of utmost importance to ensure that all "instances" of your "names" (types) are *always* usable in some meaningful way. In other words, all knowledge encodings should create an stringent, explicit set of rules for which all "instances" should conform to, and, arguably, also creates a justification for the need to create that particular knowledge/data type. As such, optimally, a domain expert would write their knowledge encodings and renderers in a general purpose programming language with a sound type system (e.g., Haskell, Agda, Java, etc) – preferring ones with a type system based on formal type theories for their feature richness.

In Drasil, we are focused in understanding families of scientific software, and creating systematic rules to generate families of software solutions (for any instance of a scientific problem that requires a scientific software solution). Specifically, Drasil is focused on mathematics and physics-based models. In both areas, we are concerned with what *kinds of theories* are wellunderstood, and ensuring that all created mathematical expressions are "valid". TODO: Still a bit awkward

1.3 Thesis Outline

todo

Chapter 2

Background

2.1 Project Focus & Goals

todo

2.2 In practice/Architecture

todo

2.2.1 Chunks

todo

2.2.2 ChunkDB?

todo

2.3 Methodology for locating and encoding knowledge – e.g., "bottom-up" approach

todo

2.4 State of Drasil

todo

2.4.1 Short-term problems – leading into topics

todo

Chapter 3

Topic #1: Typing the Expression Language

3.1 Background: Problem

todo

3.2 Requirements & properties of a good solution

todo

3.3 Solution

todo

3.3.1 Expression encodings discussion (GADTs, TTF)

todo

3.3.2 Dividing expression languages (CodeExpr, Expr, ModelExpr, Literals)

todo

3.4 Continued problem with Expressions – leading into ModelKinds

todo

Chapter 4

Topic #2: ModelKinds – Theory types / discrimination – “Expressions in context”

4.1 Problem

todo

4.2 Requirements & properties of a good solution

todo

4.3 Solution – ModelKinds

todo

4.3.1 EquationalModels

todo

4.3.2 EquationalRealms

todo

4.3.3 EquationalConstraints

todo

4.3.4 DEModel

todo

4.3.5 Continued

todo

Chapter 5

Future Work

todo

Chapter 6

Conclusion

todo

Appendix A

Appendix

todo