Modularly Programmable Syntax and Type Structure

Cyrus Omar

TODO: TR Number

July 27, 2015

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Jonathan Aldrich, Chair
TODO: confirm rest of committee

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

TODO: Support Funded by DOE CSGF, NSF GRFP, NSF, NSA, ...?

Keywords: TODO: keywords

TODO: Dedication

Abstract

Functional programming languages like ML descend conceptually from minimal lambda calculi, but to be pragmatic, expose a concrete syntax and type structure to programmers of a more elaborate design. Language designers have many viable choices along these dimensions, as evidenced by the diversity of dialects that continue to proliferate around these languages. But such language dialects cannot be modularly combined, limiting the choices available to programmers. We describe and formally specify new language primitives designed to decrease the need for dialects by giving library providers the ability to safely and modularly control syntactic expansion, typechecking and translation to a minimal type-theoretic internal language.

TODO: should I expand abstract?

Acknowledgments

TODO: Acknowledgments

Contents

1 Motivation **1**

2 Language Overview **3**
2.1 External Language 3
2.2 Internal Language 3
2.3 Static Language 3
2.4 Module Language 3

I Modularly Programmable Syntax **5**

3 Motivating Examples **7**
3.1 Lists 7
3.2 HTML 7
3.3 Regular Expressions 7
3.4 Monadic Commands 7
3.5 Quasiquotation 7

4 Existing Approaches **9**
4.1 Dynamic String Parsing 9
4.2 Direct Syntax Extension 9
4.3 Term Rewriting 9

5 Typed Syntax Macros **11**
5.1 Examples 11
5.2 Minimal Formalization 11
5.3 Parameterized TSMs 11

6 Type-Specific Languages **13**
6.1 Examples 13
6.2 Minimal Formalization 13
6.3 Parameterized TSLs 13

II Modularly Programmable Type Structure **15**

7 Motivating Examples **17**

8 Existing Approaches **19**

9 Metamodules **21**

10 Conclusion & Future Work **23**

List of Figures

Chapter 1

Motivation

Chapter 2

Language Overview

2.1 External Language

2.2 Internal Language

2.3 Static Language

2.4 Module Language

TEST (This is a test).

TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST a sfdasdf (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).

TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
TEST (This is a test).
HELLO
GOODSBYE

Part I

Modularly Programmable Syntax

Chapter 3

Motivating Examples

3.1 Lists

3.2 HTML

3.3 Regular Expressions

3.4 Monadic Commands

3.5 Quasiquotation

Chapter 4

Existing Approaches

4.1 Dynamic String Parsing

4.2 Direct Syntax Extension

Related work I haven't mentioned yet:

- Fan: http://zhanghongbo.me/fan/start.html
- Well-Typed Islands Parse Faster:
  `http://www.ccs.neu.edu/home/ejs/papers/tfp12-island.pdf`

4.3 Term Rewriting

Chapter 5

Typed Syntax Macros

5.1 Examples

5.2 Minimal Formalization

5.3 Parameterized TSMs

Chapter 6

Type-Specific Languages

6.1 Examples

6.2 Minimal Formalization

6.3 Parameterized TSLs

Part II

Modularly Programmable Type Structure

Chapter 7

Motivating Examples

Chapter 8

Existing Approaches

Chapter 9

Metamodules

Chapter 10

Conclusion & Future Work

TODO: Remove outline around links
TODO: Bibliography style