



VRIJE UNIVERSITEIT AMSTERDAM

MASTER'S THESIS

*Submitted in partial fulfillment of the requirements for
the degree of Master of Science in
Parallel and Distributed Computer Systems.*

I don't yet know what to call it, lol

Christopher Esterhuyse

(ID: 2553295)

supervisors

Vrije Universiteit Amsterdam
dr. J. ENDRULLIS

Centrum Wiskunde & Informatica
dr. F. ARBAB

July 13, 2019

Abstract

TODO

Contents

| | |
|---|-----------|
| I Preliminaries | 5 |
| 1 Introduction | 6 |
| 2 Background | 7 |
| 2.1 Reo | 7 |
| 2.1.1 Goal | 7 |
| 2.1.2 Language | 7 |
| 2.1.3 Semantic Models | 7 |
| 2.1.4 The Reo Compiler | 7 |
| 2.2 Target Languages | 7 |
| 2.2.1 Affine Types | 7 |
| 2.2.2 The Rust Language | 7 |
| 2.2.3 Programming Patterns | 7 |
| II Contributions | 8 |
| 3 Protocol Translation | 9 |
| 3.1 Two-Phase Generation | 9 |
| 3.1.1 Motivation | 9 |
| 3.1.2 Imperative Rule Form | 9 |
| 3.2 Code Generation | 9 |
| 3.2.1 Reo Side | 9 |
| 3.2.2 Rust Side | 9 |
| 4 Protocol Runtime | 10 |
| 4.1 Reference Implementation: Java Generator | 10 |
| 4.1.1 Structure: Components, Ports, and Threads | 10 |

| | | |
|------------|--|-----------|
| 4.1.2 | Behavior: Rules | 10 |
| 4.1.3 | Observations | 10 |
| 4.2 | Goals | 10 |
| 4.2.1 | Functional Requirements | 10 |
| 4.2.2 | Non-Functional Requirements | 10 |
| 4.3 | Runtime Properties | 11 |
| 4.3.1 | User-Facing | 11 |
| 4.3.2 | Internal | 11 |
| 5 | Static Governors | 12 |
| 5.1 | Problem: Unintended Constraints | 12 |
| 5.2 | Governor Defined | 12 |
| 5.3 | Solution: Static Governance with Types | 12 |
| 5.4 | Functionality | 12 |
| 5.4.1 | encoding CA and RBA as Type-State Automata | 12 |
| 5.4.2 | Rule Consensus | 12 |
| 5.4.3 | Governed Environment | 12 |
| 5.5 | Feasibility | 13 |
| 5.5.1 | RBA Simplification | 13 |
| 5.5.2 | RBA Preprocessing | 13 |
| 5.5.3 | Opt-in Simplification | 13 |
| 5.5.4 | Match Macros | 13 |
| 6 | Benchmarking | 14 |
| 6.1 | Goal | 14 |
| 6.2 | Experimental Setup | 14 |
| 6.3 | Results | 14 |
| 6.4 | Observations | 14 |
| III | Reflection | 15 |
| 7 | Discussion | 16 |
| 7.1 | Future Work | 16 |
| 7.1.1 | Distributed Components | 16 |
| 7.1.2 | Runtime Governors | 16 |
| 7.2 | Conclusion | 16 |

List of Figures

Part I

Preliminaries

Chapter 1

Introduction

Chapter 2

Background

2.1 Reo

2.1.1 Goal

2.1.2 Language

2.1.3 Semantic Models

2.1.4 The Reo Compiler

2.2 Target Languages

2.2.1 Affine Types

2.2.2 The Rust Language

2.2.3 Programming Patterns

2.2.3.1 Type-State

2.2.3.2 Proof of Work

Part II

Contributions

Chapter 3

Protocol Translation

3.1 Two-Phase Generation

3.1.1 Motivation

3.1.2 Imperative Rule Form

3.2 Code Generation

3.2.1 Reo Side

3.2.1.1 **Compiler Internal Representation**

3.2.1.2 **Sequentializing**

3.2.1.3 **Type Constraining**

3.2.2 Rust Side

3.2.2.1 **Checking and Errors**

3.2.2.2 **Optimization**

3.2.2.3 **Commandification**

Chapter 4

Protocol Runtime

4.1 Reference Implementation: Java Generator

4.1.1 Structure: Components and Ports

4.1.2 Behavior: Rules

4.1.3 Observations

4.2 Goals

4.2.1 Functional Requirements

features, safety

4.2.1.1 Features

4.2.1.2 Safety

4.2.2 Non-Functional Requirements

performance, maintainability

4.3 Runtime Properties

4.3.1 User-Facing

4.3.1.1 Protocol Construction

4.3.1.2 Port Construction

4.3.1.3 Destruction and Termination

4.3.2 Internal

4.3.2.1 Protocol Object Architecture

4.3.2.2 Rule Firing

4.3.2.3 Design Choices and Optimizations

Chapter 5

Static Governors

5.1 Problem: Unintended Constraints

5.2 Governor Defined

5.3 Solution: Static Governance with Types

5.4 Functionality

5.4.1 encoding CA and RBA as Type-State Automata

5.4.2 Rule Consensus

5.4.3 Governed Environment

5.4.3.1 Governor Entrypoint

words

5.4.3.2 Port Wrappers

5.5 Feasibility

5.5.1 RBA Simplification

5.5.1.1 Motivation

5.5.1.2 Consequence: Loss of Distinction

5.5.2 RBA Preprocessing

5.5.2.1 Projection and Hiding

5.5.2.2 Normalization

5.5.3 Opt-in Simplification

5.5.4 Match Macros

Chapter 6

Benchmarking

- 6.1 Goal
- 6.2 Experimental Setup
- 6.3 Results
- 6.4 Observations

Part III

Reflection

Chapter 7

Discussion

7.1 Future Work

7.1.1 Distributed Components

7.1.2 Runtime Governors

7.2 Conclusion

Bibliography