## TastyTruffle: A Subtitle

by

### James You

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2022

© James You 2022

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

#### Abstract

This is the abstract.

### Acknowledgements

I would like to thank all the little people who made this thesis possible.

#### Dedication

This is dedicated to the one I love.

## Table of Contents

List of Tables								
Li	List of Figures							
A	bbre	viation	ns	x				
1	Inti	roduct	ion	1				
2	Background							
	2.1	Intern	nediate Representations	2				
		2.1.1	Java Bytecode	2				
		2.1.2	Scala Typed Abstract Syntax Trees	2				
		2.1.3	GraalVM Intermediate Representation	2				
	2.2	Mana	ged Runtimes	2				
		2.2.1	Type Erasure	2				
		2.2.2	Just-in-time Compilation	2				
3	Implementation							
	3.1	Tasty'	Truffle Intermediate Representation	3				
		3.1.1	Root Node	4				
		3.1.2	Read and Write Nodes	4				
		3.1.3	Control Flow Nodes	4				

		3.1.4	Call Nodes	4			
		3.1.5	Type Nodes	4			
		3.1.6	Allocation Nodes	4			
		3.1.7	Example	4			
	3.2	Specia	lization	5			
	3.3	Specia	lizing Methods	5			
		3.3.1	Code Path Duplication	5			
		3.3.2	Typed Dispatch	5			
4	Eva	luation	1	7			
5	Rela	ated W	Vork	8			
6	5 Future Work						
7	7 Conclusions						
References							
APPENDICES							

## List of Tables

# List of Figures

3.1	Example implementation of a checksum function	4
3.2	Graal IR graph of specialization checksum[Int] after Truffle tier	6

## Abbreviations

 ${\bf TASTy}\,$  Typed Abstract Syntax Tree 3

## Introduction

## Background

This section should mainly explore type erasure and how it relates to the various sections below.

### 2.1 Intermediate Representations

- 2.1.1 Java Bytecode
- 2.1.2 Scala Typed Abstract Syntax Trees
- 2.1.3 GraalVM Intermediate Representation
- 2.2 Managed Runtimes
- 2.2.1 Type Erasure
- 2.2.2 Just-in-time Compilation

### Implementation

### 3.1 TastyTruffle Intermediate Representation

As Scala programs in TASTy format are unsuitable for execution in a Truffle interpreter, programs in TASTy must be parsed and transformed into an executable intermediate representation in TASTYTRUFFLE. The following sections will introduce the nodes in TastyTruffle IR and how they are derived from Scala source and TASTy.

#### 3.1.1 Root Node

#### 3.1.2 Read and Write Nodes

(x:)

#### 3.1.3 Control Flow Nodes

- 3.1.4 Call Nodes
- 3.1.5 Type Nodes
- 3.1.6 Allocation Nodes

```
new Foo
new Array[Int]
new Array[T]
```

### 3.1.7 Example

```
def checksum[T](data: Array[T]): Int = {
    val sum: Int = 0
    var index: Int = 0
    while (index < data.length) {
        val sum += data[i].##
        index += 1
    }
}</pre>
```

Figure 3.1: Example implementation of a checksum function.

### 3.2 Specialization

Cover the types of specializable terms.

### 3.3 Specializing Methods

- 3.3.1 Code Path Duplication
- 3.3.2 Typed Dispatch
- 3.3.3 Partial Evaluation

Use pseudocode example to show partial evaluation? Algorithm? GraalIR?

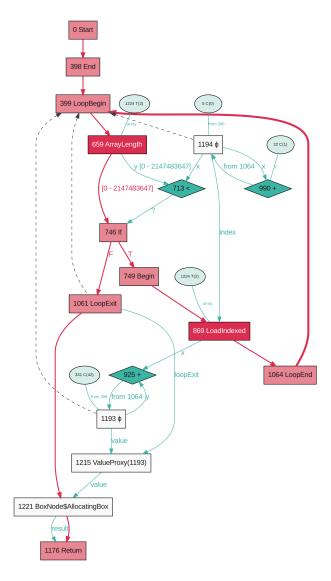


Figure 3.2: Graal IR graph of specialization checksum[Int] after Truffle tier.

Evaluation

Related Work

Future Work

Conclusions

### References

- [1] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LATEX Companion*. Addison-Wesley, Reading, Massachusetts, 1994.
- [2] Donald Knuth. The TeXbook. Addison-Wesley, Reading, Massachusetts, 1986.
- [3] Leslie Lamport. partial TEX A Document Preparation System. Addison-Wesley, Reading, Massachusetts, second edition, 1994.

## **APPENDICES**