



UNIVERSITY OF GOTHENBURG

Implementing incremental and parallel parsing

A subtitle that can be rather long

Master of Science Thesis in Computer Science

TOBIAS OLAUSSON

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, April 2014

The Author grants to Chalmers University of Technology and University of Gothenburg the non-exclusive right to publish the Work electronically and in a non-commercial purpose make it accessible on the Internet. The Author warrants that he/she is the author to the Work, and warrants that the Work does not contain text, pictures or other material that violates copyright law.

The Author shall, when transferring the rights of the Work to a third party (for example a publisher or a company), acknowledge the third party about this agreement. If the Author has signed a copyright agreement with a third party regarding the Work, the Author warrants hereby that he/she has obtained any necessary permission from this third party to let Chalmers University of Technology and University of Gothenburg store the Work electronically and make it accessible on the Internet.

Implementing incremental and parallel parsing

A subtitle here that can be quite long

TOBIAS OLAUSSON

© TOBIAS OLAUSSON, April 2014

Examiner: PATRIK JANSSON

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Cover: an image that is used as a cover image

Department of Computer Science and Engineering
Göteborg, Sweden, April 2014

Abstract

This thesis shows an incremental implementation of everything

Contents

1	Background	3
1.1	Introduction	3
1.1.1	Incrementality	3
1.1.2	Parallelism	3
1.1.3	Parsing	3
1.1.4	Motivation	3
1.2	Lexing	3
1.3	Context-free grammars	4
1.3.1	Chomsky Normal Form	4
1.4	Parsing	4
1.4.1	CYK algorithm	4
1.4.2	Improvement by Bernardy & Claessen	4
1.5	Dependently typed programming	4
1.5.1	Kinds, Types and Values	4
2	Implementation	5
2.1	Measuring	5
2.1.1	Pipeline of measures	5
2.2	Lexing	5
2.3	Parsing	5
2.3.1	BNFC	5
2.3.2	Dependently typed programming with charts	5
2.3.3	Oracle and unsafePerformIO	5
2.4	Testing	5
2.5	Final product	5
3	Results	6
3.1	Measurements	6

4	Discussion	7
4.1	Implementation	7
4.1.1	Too many result branches	7
4.1.2	LexGen – Alex discrepancy	7
4.2	Improvements	7
4.3	Future work	7

Chapter 1

Background

1.1 Introduction

What is the topic? Describe each word in the title. Something like a motivation on why this is interesting at all.

1.1.1 Incrementality

1.1.2 Parallelism

1.1.3 Parsing

1.1.4 Motivation

1.2 Lexing

Shortly describe LexGen and its relevance.

1.3 Context-free grammars

1.3.1 Chomsky Normal Form

1.4 Parsing

1.4.1 CYK algorithm

1.4.2 Improvement by Bernardy & Claessen

1.5 Dependently typed programming

1.5.1 Kinds, Types and Values

What is dependently typed programming, and how can it be used in Haskell.

Chapter 2

Implementation

2.1 Measuring

2.1.1 Pipeline of measures

An illustration would be good here

2.2 Lexing

2.3 Parsing

2.3.1 BNFC

2.3.2 Dependently typed programming with charts

2.3.3 Oracle and unsafePerformIO

2.4 Testing

2.5 Final product

Chapter 3

Results

3.1 Measurements

How fast is it? What is the complexity?

Chapter 4

Discussion

4.1 Implementation

4.1.1 Too many result branches

4.1.2 LexGen – Alex discrepancy

4.2 Improvements

4.3 Future work