Bachelor's Thesis Computing Science

Radboud University Nijmegen

Thesis Title

Subtitle if you like

Author:
Floris Reuvers
s1096976

First supervisor/assessor:
Dr. Niels van der Weide

Second assessor:
Dr. Engelbert Hubbers

February 16, 2025

Abstract

Brief outline of research questions, results. (The preferred size of an abstract is one paragraph or one page of text.)

Contents

Chapter 1

Introduction

The introduction of your bachelor thesis introduces the research area, the research hypothesis, and the scientific contributions of your work. A good narrative structure is the one suggested by Simon Peyton Jones [1]:

- describe the problem / research question
- motivate why this problem must be solved
- demonstrate that a (new) solution is needed
- explain the intuition behind your solution
- motivate why / how your solution solves the problem (this is technical)
- explain how it compares with related work

Close the introduction with a paragraph in which the content of the next chapters is briefly mentioned (one sentence per chapter).

Starting a new paragraph is done by inserting an empty line like this.

Chapter 2

Preliminaries

This *optional* chapter contains the stuff that your reader needs to know in order to understand your work. Your "audience" consists of fellow third year computing science bachelor students who have done the same core courses as you have, but not necessarily the same specialization, minor, or free electives.

2.1 Rules

2.1.1 Rules Intuitionistic Logic

$$\frac{}{A \vdash A}\ \text{Id} \qquad \frac{\Gamma, \Delta \vdash A}{\Delta, \Gamma \vdash A}\ \text{Exchange}$$

$$\frac{\Gamma, A, A \vdash B}{\Gamma, A \vdash B}\ \text{Contraction} \qquad \frac{\Gamma \vdash B}{\Gamma, A \vdash B}\ \text{Weakening}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B} \to I \qquad \frac{\Gamma \vdash A \to B \qquad \Delta \vdash A}{\Gamma, \Delta \vdash B} \to E$$

$$\frac{\Gamma \vdash A \qquad \Delta \vdash B}{\Gamma, \Delta \vdash A \times B} \times I \qquad \frac{\Gamma \vdash A \times B \qquad \Delta, A, B \vdash C}{\Gamma, \Delta \vdash C} \times E$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A + B} +I_1 \quad \frac{\Gamma \vdash B}{\Gamma \vdash A + B} +I_2 \quad \frac{\Gamma \vdash A + B \qquad \Delta, A \vdash C \qquad \Delta, B \vdash C}{\Gamma, \Delta \vdash C} +I$$

Figure 2.1: Natural Deduction rules for Intuitionistic Logic

2.1.2 Intuitionistic Logic with types

$$\frac{}{x : A \vdash x : A}\ \mathrm{Id} \qquad \frac{\Gamma, \Delta \vdash x : A}{\Delta, \Gamma \vdash x : A}\ \mathrm{Exchange}$$

$$\frac{\Gamma, y : A, z : A \vdash u : B}{\Gamma, x : A \vdash u[y \mapsto x][z \mapsto x] : B}\ \mathrm{Contraction} \qquad \frac{\Gamma \vdash y : B}{\Gamma, x : A \vdash y : B}\ \mathrm{Weakening}$$

$$\frac{\Gamma, x : A \vdash u : B}{\Gamma \vdash \lambda x.u : A \to B} \to I \qquad \frac{\Gamma \vdash f : A \to B \qquad \Delta \vdash x : A}{\Gamma, \Delta \vdash s(x) : B} \to E$$

$$\frac{\Gamma \vdash x : A \qquad \Delta \vdash y : B}{\Gamma, \Delta \vdash (x, y) : A \times B}\ \times I \qquad \frac{\Gamma \vdash s : A \times B \qquad \Delta, x : A, y : B \vdash u : C}{\Gamma, \Delta \vdash \text{case } s \text{ of } (x, y) \to u : C}\ \times E$$

$$\frac{\Gamma \vdash x : A}{\Gamma \vdash \mathrm{inl}(x) : A + B}\ +I_1 \qquad \frac{\Gamma \vdash x : B}{\Gamma \vdash \mathrm{inr}(x) : A + B}\ +I_2$$

$$\frac{\Gamma \vdash s : A + B \qquad \Delta, x : A \vdash v : C \qquad \Delta, y : B \vdash w : C}{\Gamma, \Delta \vdash \text{case } s \text{ of } \mathrm{inl}(x) \to v;\ \mathrm{inr}(y) \to w : C}\ +I$$

Figure 2.2: Natural Deduction rules for Intuitionistic Types

Chapter 3

Lambda Calculus

This sections provides a detailed and formal description of the λ-calculus. We define a formal grammar of the λ-calculus and give examples of the λ-calculus. Then we explain β-reduction, after which we define call-by-name evaluation and call-by-value evaluation.

3.1 Introduction to the λ-calculus

The simplest and most basic grammer of the lambda calculus is as follows:

$$M, N, P, Q \quad ::= \quad x \mid \ \lambda x.M \mid \ MN$$

So, terms are denoted by M, N, P or Q and they can either be of the form x, $\lambda x.M$ or MN:

- x is a variable, which is a symbol that represents an input or a value.
- $\lambda x.M$ is an abstraction. An abstraction is an anonymous function, where x is the parameter and M is the body of the function.
- MN is a function application, where M and N are terms.

Here, we give an example of an abstraction and of a function application. $\lambda x.x$ is an abstraction. This specific abstraction is called the identity function and has one input parameter, namely x, and returns the input x. This function is often denoted as $\mathbf{I}$. $(\lambda x.x)y$ is a function application. The anonymous function is applied to the variable y and the λ-term reduces to the variable y.

We now discuss bound and free variables. Let us consider the following two functions: $\lambda x.x$ and $\lambda y.y$. These functions will do the exact same. $(\lambda x.x)z$ returns the same value as $(\lambda y.y)z$, namely the variable z. Therfore the name of the variables x and y do not matter. These variables are called bound variables. A variable is considered a bound variable if it occurs in an abstraction and the variable name is a parameter in that abstraction. A variable is free if/iff?? it is not bound. In the abstraction $(\lambda x.xy)z$, the variable x is bound, while y and z are free.

With the definition of free variables, we can define substitution. Let us consider the last example $(\lambda x.xy)z$ again. z is a free variable and all occurrences of the varaible x in the body of the function will be substituted by the variable z. We denote this as follows: $(xy)[x \mapsto z]$. The function application $(\lambda x.(\lambda x.xx)x)y$ should reduce to $(\lambda x.xx)y$ and $((\lambda x.xx)x)[x \mapsto y] = (\lambda x.xx)y$. Therefore, the substitution of x by y in lambda term M is defined by replacing all free occurences of x by y.

3.2 β-Reduction

3.3 Values

In the λ-calculus, there is a distinction between terms that are values and terms that are not a value. Values are all terms that cannot be reduced. That is, a value is either a variable or an abstraction (function) that is in normal form. The following terms are values:

$$x \qquad \lambda x.x + 37 \qquad \lambda x.\lambda y(x+y)$$

The following two terms are not values:

$$(\lambda x.x)z \qquad (\lambda x.\lambda y(x+y))37$$

If the letter V or W is used for a term, it is assumed that that term is a value. Terms, indicated with the letter M, N, P or Q, can be any kind of term. This distinciton is important for defining beta reduction rules for cbn and cbv lambda calculus. Beta reduction of the cbn and cbv lamda calculus is indicated by β_n and β_v respectively. β_n and β_v can be defined as follows:

$$\beta_n : \quad (\lambda x.M)N \to M[x \mapsto N] \qquad \beta_v : \quad (\lambda x.M)V \to M[x \mapsto V]$$

These reductions are quite straightforward. With β_n, we replace the variable in the function by the argument without eveluating it. With β_v, the argument should be a value, indicated by V. Only if the argument is a value, the variable of the function will be replaced by the argument. We define $M[x \mapsto N]$ as the term M where all free occurrences of x are replaced by N. Explain what 'free' occurrences are???

Now consider the following lambda term:

$$(\lambda x.x + ((\lambda y.y)7))((\lambda x.x)30)$$

If we do not define extra closure rules, we can only beta reduce this term with β_n, since $(\lambda x.x)30$ is not a value. It would be nice if we could reduce

$(\lambda x.x)30$ or $(\lambda x.x+((\lambda y.y)7))$ in the bigger term. This is made possible with the following closure rules. The rules are specified with a horizontal line. If we can proof that everything above the line holds, then we can conclude everything below the line.

$$\frac{M \to M'}{MN \to M'N}\,\mu \qquad \frac{N \to N'}{MN \to MN'}\,\nu \qquad \frac{M \to M'}{\lambda x.M \to \lambda x.M'}\,\xi$$

Chapter 4

Research

This chapter, or series of chapters, delves into all technical details that are required to *prove* your scientific hypothesis. It should be sufficiently detailed and precise in order for any fellow computing scientist student to be able to *repeat* your research and therewith establish the same results / conclusions that you have obtained. Please note that, in order to improve readability of your thesis, you can put a part of this information also in one or more appendices (see Appendix A).

Chapter 5

Related Work

In this chapter you demonstrate that you are sufficiently aware of the state-of-art knowledge of the problem domain that you have investigated as well as demonstrating that you have found a *new* solution / approach / method.

Chapter 6

Conclusions

In this chapter you present all conclusions that can be drawn from the preceding chapters. It should not introduce new experiments, theories, investigations, etc.: these should have been written down earlier in the thesis. Therefore, conclusions can be brief and to the point.

Bibliography

[1] Simon Peyton Jones. How to write a good research paper, 2004. Presentation at Technical University of Vienna, `http://research.microsoft.com/en-us/um/people/simonpj/papers/giving-a-talk/writing-a-paper-slides.pdf`.

Appendix A

Appendix

Appendices are *optional* chapters in which you cover additional material that is required to support your hypothesis, experiments, measurements, conclusions, etc. that would otherwise clutter the presentation of your research.