Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik

Bachelor Thesis Bioinformatics

Title of thesis

Name

Date

Reviewer

Name Reviewer
Department of Computer Science
University of Tübingen

Supervisor

Name Supervisor
Address
University of Tübingen

Name, first name:
Title of thesis
Bachelor Thesis Bioinformatics
Eberhard Karls Universität Tübingen
Period: from-till

Abstract

Write here your abstract.

Acknowledgements

Write here your acknowledgements.

Contents

List of Theorems

1 Introduction

2 The Untyped Calculus ND

I will be introducing the Untyped Calculus ND, based on ...

2.1 Syntax of the Untyped Calculus ND

Some knowledge of notation is necessary to familiarize oneself with the syntax of the Untyped Calculus. $\overline{X}$ represents a (possibly empty) sequence $X_1, ...X_i, ...X_n$.
A pattern match $e.\mathbf{case}\{\overline{K(\overline{x}) \Rightarrow e}\}$ matches a term e against a sequence of clauses, each clause consisting of a constructor and an expression. The expression associated with first constructor that matches the term is the result of the pattern match. For a copattern match $\mathbf{cocase}\{\overline{d(\overline{x}) \Rightarrow e}\}$ the same rules apply, but instead of constructers, the term is matched against destructors.

$$
\begin{array}{rcll}
e & ::= & x & \text{Variable} \\
 & | & K(\overline{e}) & \text{Constructor} \\
 & | & e.d(\overline{e}) & \text{Destructor} \\
 & | & e.\mathbf{case}\{\overline{K(\overline{x}) \Rightarrow e}\} & \text{Pattern match} \\
 & | & \mathbf{cocase}\{\overline{d(\overline{x}) \Rightarrow e}\} & \text{Copattern match}
\end{array}
$$

Definition 2.1 (Terms of the Calculus ND).

In pattern and copattern matches, every con- or destructor may occur no more than once.

Let's look at some examples to ... All rudimentary datatypes can be constructed through constructors:
The Booleans `True` and `False` are constructors on an empty sequence.

Abstract data types like lists, arrays, records etc. can similarly be defined through constructors:
`Cons(True, Cons(False, Nil))` and `Date(...)`

We can use Pattern matching to represent conditionals: Lik this if statement: "$\mathbf{if} e_1 \mathbf{then} e_2 \mathbf{else} e_3$" which is analogous to: $e_1.\mathbf{case}\{\texttt{True} \Rightarrow e_2, \texttt{False} \Rightarrow e_3\}$ Or the expression which tests whether a given list contains the number 5: $e.\mathbf{case}\{\}$

2.2 Free Variables, Subsitutions, Contexts

Definition 2.2 (Free Variables).

Definition 2.3 (Substitution).

Definition 2.4 (Domain and Range of a Substitution).

Definition 2.5 (Action of a Substitution).

Definition 2.6 (Composition of Subsitutions).

2.3 Conversion

Definition 2.7 (Beta-Conversion).

Definition 2.8 (Eta-Conversion for Codata).

2.4 The unification problem

The Unification Problem is described by a set of equations with expressions on each side $e = e$ with unknown unification variables $\alpha^?$, where our goal is to find a number of pairs/mappings $\alpha^? \mapsto e$ consisting of unification variables and expressions, such that when substituting the expressions for the unification variables both sides of the given equations are the same.
To describe this formally, we need to expand our syntax to be able to represent unification variables.

$$
\begin{array}{rcll}
e & ::= & \alpha^? & \text{Unification variable} \\
& | & x & \text{Variable} \\
& | & K(\overline{e}) & \text{Constructor} \\
& | & e.d(\overline{e}) & \text{Destructor} \\
& | & e.\mathbf{case}\{\overline{K(\overline{x}) \Rightarrow e}\} & \text{Pattern match} \\
& | & \mathbf{cocase}\{\overline{d(\overline{x}) \Rightarrow e}\} & \text{Copattern match}
\end{array}
$$

For some unification problems, there exists a solution and it is obvious: The solution to $\alpha^? = True$ is $\alpha^? \mapsto True$

For some unification problems, it is rather obvious that there is no solution on the other hand: There is no mapping of unification variables that make both sides of $True = False$ the same.

Solutions aren't necessarily unique, either. The problem $\alpha^? 5 = 5$ has multiple solutions: $\alpha^? \mapsto \lambda x.x$ and $\alpha^? \mapsto \lambda x.5$.

The interesting unification problems are those where it is not clear at first sight whether there exists a solution.

Selbständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe und dass alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entnommen sind, durch Angaben von Quellen als Entlehnung kenntlich gemacht worden sind. Diese Bachelorarbeit wurde in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt.

Ort, Datum Unterschrift