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. $X$ 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 constructors, the term is matched against destructors.

| | | | |
|---|---|---|---|
| $e$ | ::= | $x$ | Variable |
| | \| | $K(\overline{e})$ | Constructor |
| | \| | $e.d(\overline{e})$ | Destructor |
| | \| | $e.\mathbf{case}\{\overline{K(\overline{x}) \Rightarrow e}\}$ | Pattern match |
| | \| | $\mathbf{cocase}\{\overline{d(\overline{x}) \Rightarrow e}\}$ | Copattern match |

**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, Substitutions, Contexts

**Definition 2.2** (Free Variables)**.** The set of free variables of a term $e$ is FV($e$). A term is closed if this set is empty. Free Variables are defined recursively over

the structure of terms as follows:

$$\begin{aligned}
\mathrm{FV}(x) &:= \{x\} \\
\mathrm{FV}(K(e_1, ...e_n)) &:= \mathrm{FV}(e_1) \cup \cdots \cup \mathrm{FV}(e_n) \\
\mathrm{FV}(e.d(e_1, ...e_n)) &:= \mathrm{FV}(e) \cup FV(e_1) \cup \cdots \cup \mathrm{FV}(e_n) \\
\mathrm{FV}(e.\mathbf{case}\{\overline{K(\overline{x}) \Rightarrow e}\}) &:= \mathrm{FV}(e) \cup (\mathrm{FV}(e_1)\backslash\overline{x}) \cup \cdots \cup (\mathrm{FV}(e_n)\backslash\overline{x}) \\
\mathrm{FV}(\mathbf{cocase}\{\overline{d(\overline{x}) \Rightarrow e}\}) &:= (\mathrm{FV}(e_1)\backslash\overline{x}) \cup \cdots \cup (\mathrm{FV}(e_n)\backslash\overline{x})
\end{aligned}$$

**Definition 2.3** (Substitution)**.** A simultaneous substitution $\sigma$ of the terms $e_1, ...e_n$ for the distinct variables $x_1, ...x_n$ is defined as follows:

$$\sigma ::= [e_1, ...e_n\backslash x_1, ...x_n]$$

The set of variables for which the substitution is defined is called the domain. The set of free variables which appear in the substitution is called the range.

**Definition 2.4** (Domain and Range of a Substitution)**.** The definitions of Domain and Range of a Substitution are as follows:

$$\begin{aligned}
\mathtt{dom}([e_1, ...e_n/x_1, ...x_n]) &:= \{x_1, ...x_n\} \\
\mathtt{rng}([e_1, ...e_n/x_1, ...x_n]) &:= \mathrm{FV}(e_1) \cup \cdots \cup \mathrm{FV}(e_n)
\end{aligned}$$

What is actually interesting is what happens when we appy a substitution to an expression

**Definition 2.5** (Action of a Substitution)**.** The action of a substitution $\sigma$ on a term $e$, written as $e\sigma$ and is defined as follows:

$$\begin{aligned}
x[e_1, ...e_n/x_1, ...x_n] &:= e_i \quad (\text{if } x = x_i) \\
y\sigma &:= y \quad (\text{if } y \notin \mathtt{dom}(\sigma)) \\
(K(e_1, ...e_n))\sigma &:= K(e_1\sigma, \ldots, e_n\sigma) \\
(e.d(e_1, ...e_n))\sigma &:= (e\sigma).d(e_1\sigma, \ldots, e_n\sigma) \\
(e.\mathbf{case}\{\overline{K(\overline{x}) \Rightarrow e}\})\sigma &:= (e\sigma).\mathbf{case}\{\overline{K(\overline{y} \Rightarrow (e\sigma')\sigma)}\} \\
(\mathbf{cocase}\{\overline{d(\overline{x}) \Rightarrow e}\})\sigma &:= \mathbf{cocase}\{\overline{(d(\overline{y}) \Rightarrow (e\sigma')\sigma)}\}
\end{aligned}$$

Where $\sigma'$ is a substitution that ensures that we don't bind new variables: $\sigma'$ has the form $[y_1, \ldots, y_n/x_1, ...x_n]$ and all $y_i$ are fresh for both the domain and the range of $\sigma$.

The composition of two substitutions $\sigma_2 \circ \sigma_1$ which is equivalent to first applying the substitution $\sigma_1$, then the substitution $\sigma_2$.

**Definition 2.6** (Composition of Subsitutions)**.** Given two substitutions

$$\sigma_1 := [e_1, ...e_n/x_1, ...x_n], \qquad \sigma_2 := [t_1, \ldots, t_m/y_1, \ldots, y_m]$$

Composition is defined as:

$$\sigma_2 \circ \sigma_1 := [e_1\sigma_2, \ldots, e_n\sigma_2, t_j, \ldots, t_k/x_1, ...x_n, y_j, \ldots, y_k]$$

Where $j, \ldots, k$ ist the greatest sub-range of indices $1, \ldots, m$ such that none of the variables $y_j$ to $y_k$ is in the domain of $\sigma_1$

## 2.3 Conversion

**Definition 2.7** (Beta-Conversion)**.** A single step of beta-conversion $e_1 \equiv^1_\beta e_2$ is defined as follows:

$$K(\overline{e}).\mathbf{case}\{\ldots, K(\overline{x} \Rightarrow e)\} \equiv^1_\beta e[\overline{e}/x] \qquad (\beta\text{-Data})$$
$$\mathbf{cocase}\{\ldots, d(\overline{x} \Rightarrow e, \ldots)\}.d(\overline{e}) \equiv^1_\beta e[\overline{e}/x] \qquad (\beta\text{-Codata})$$

**Definition 2.8** (Eta-Conversion for Codata)**.**

# 3 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.

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

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