Pontifícia Universidade Católica
do Rio de Janeiro

Felipe Vieira Cortes

An alternative formalization of reference typing

Tese de Doutorado

Thesis presented to the Programa de Pós–graduação em Informática, do Departamento de Informática da PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Informática.

Advisor: Prof. Roberto Ierusalimschy

Rio de Janeiro
March 2023

Pontifícia Universidade Católica
do Rio de Janeiro

Felipe Vieira Cortes

An alternative formalization of reference typing

Thesis presented to the Programa de Pós–graduação em Informática da PUC-Rio in partial fulfillment of the requirements for the degree of Doutor em Informática. Approved by the Examination Committee:

Prof. Roberto Ierusalimschy
Advisor
Departamento de Informática – PUC-Rio

Prof. Roberto Ierusalimschy
Departamento de Informática – PUC-Rio

Rio de Janeiro, March 11th, 2023

Felipe Vieira Cortes

Graduated in computer science by the Pontifícia Universidade Católica do Rio de Janeiro.

Bibliographic data

Cortes,Felipe V.

An alternative formalization of reference typing / Felipe Vieira Cortes; advisor: Roberto Ierusalimschy. – 2023.

?? f: il. color. ; 30 cm

Tese (doutorado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2023.

Inclui bibliografia

1. Informática – Teses. 2. Type References. 3. Programming Languages. 4. Formalization. I. Ierusalimschy, Roberto. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

Acknowledgments

Abstract

Cortes,Felipe V.; Ierusalimschy, Roberto (Advisor). **An alternative formalization of reference typing**. Rio de Janeiro, 2023. **??**p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Reference Typing is a ...

Keywords

Type References; Programming Languages; Formalization.

Resumo

Cortes,Felipe V.; Ierusalimschy, Roberto. **An alternative formalization of reference typing**. Rio de Janeiro, 2023. **??**p. Tese de Doutorado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

A tipagem de referencia eh ...

Palavras-chave

Type References; Programming Languages; Formalization.

Table of contents

List of figures

List of tables

List of algorithms

List of codes

List of Abreviations

ADI – Análise Digital de Imagens

BIF – *Banded Iron Formation*

My beautifull epigraph

Wassily Kandinsky, *Regards sur le passé.*

1
Introduction

So, to start with, could you provide me with more details about your research idea? What specific area in the field of programming languages are you interested in? What problem or question would you like your research to address?

Sure. I'll provide you with more details about our research idea. I will present you some background information so we can formulate an enthralling laconic research proposal for our target audience.

I am interested in Typing Mutable References. The main reference for the problem my research adresses is the Software Foundations series by Benjamin C. Pierce. The first two volumes, "Logical Foundations" and "Programming Language Foundations" gives a solid understanding of "functional programming, basic concepts of logic, computer-assisted theorem proving, and Coq." and also "the theory of programming languages, including operational semantics and static type systems." respectively. More specifically, the chapter "References" contains a very interesting implementation of mutable references in a Simply Typed Lambda Calculus (STLC) language that we take as an inspiration for condcting our study aroud typing mutable references.

The main idea behind our study is to - propose a different approach for the syntax definition of terms inside a Simply Typed Lambda Calculus (STLC) language. - redefine the operational semantics comprising our new term definition - Check if the standard type safety properties still hold (progress and preservation theorems). - Analyse both solutions (canonical and research) comparing simplicity, reasoning, understanding and proof implementations.

Naturally, when introducing mutable references in a language definition, we would consider the following terms (as described in the book Programming Language Foundations, chapter References):

"'coq (** *** Terms *)

(** Besides variables, abstractions, applications, natural-number-related terms, and [unit], we need four more sorts of terms in order to handle mutable references:

t ::= ... Terms | ref t allocation | !t dereference | t := t assignment | l location *)

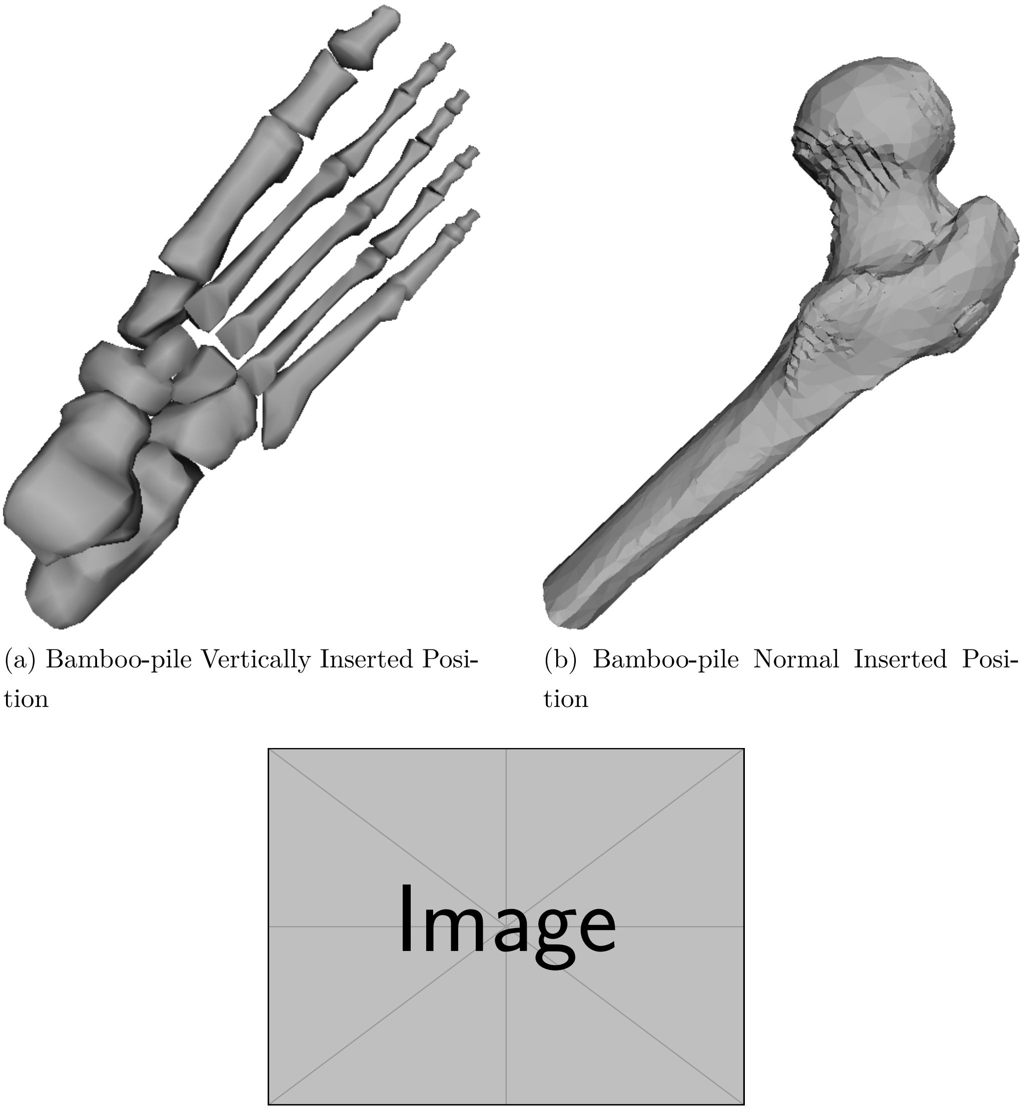
Inductive tm : Type := (* STLC with numbers: *) | tm$_{v}ar : string- > tm|tm_{a}pp : tm- > tm- > tm|tm_{a}bs : string- > ty- > tm- > tm|tm_{c}onst : nat- > tm|tm_{s}ucc : tm- > tm|tm_{p}red : tm- > tm|tm_{m}ult : tm- > tm- >

$tm|tm_{i}f0 : tm- > tm- > tm- > tm(*Newterms : *)|tm_{u}nit : tm|tm_{r}ef : tm- > tm|tm_{d}eref : tm- > tm|tm_{a}ssign : tm- > tm- > tm|tm_{l}oc : nat- > tm.$“‘

This document is structured as follows. In Chapter **??** we present some previous work relevant to our problem. In Chapter **??** we explain our proposal. In Chapter **??** we show our results. Finally, in Chapter **??** we present our conclusion and future work.

2
Previous Work

Early smoothing methods tried to minimize... In the figure **??** we see...

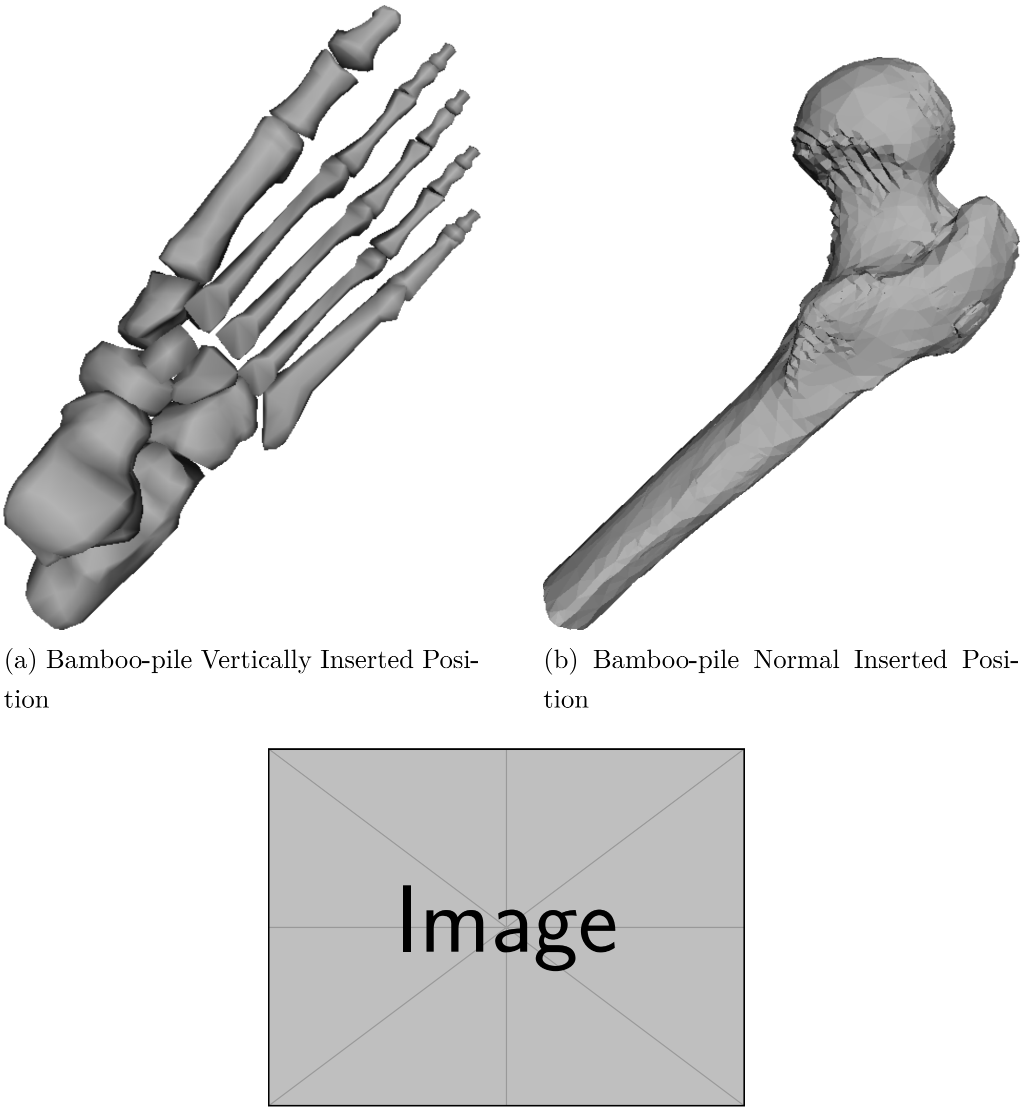


(a) Bamboo-pile Vertically Inserted Position

(b) Bamboo-pile Normal Inserted Position

(c) bamboo-pile Inserted 45° angle

Figure 2.1: A set of three subfigures: (a) describes the first subfigure; (b) describes the second subfigure; (c) describes the third subfigure.

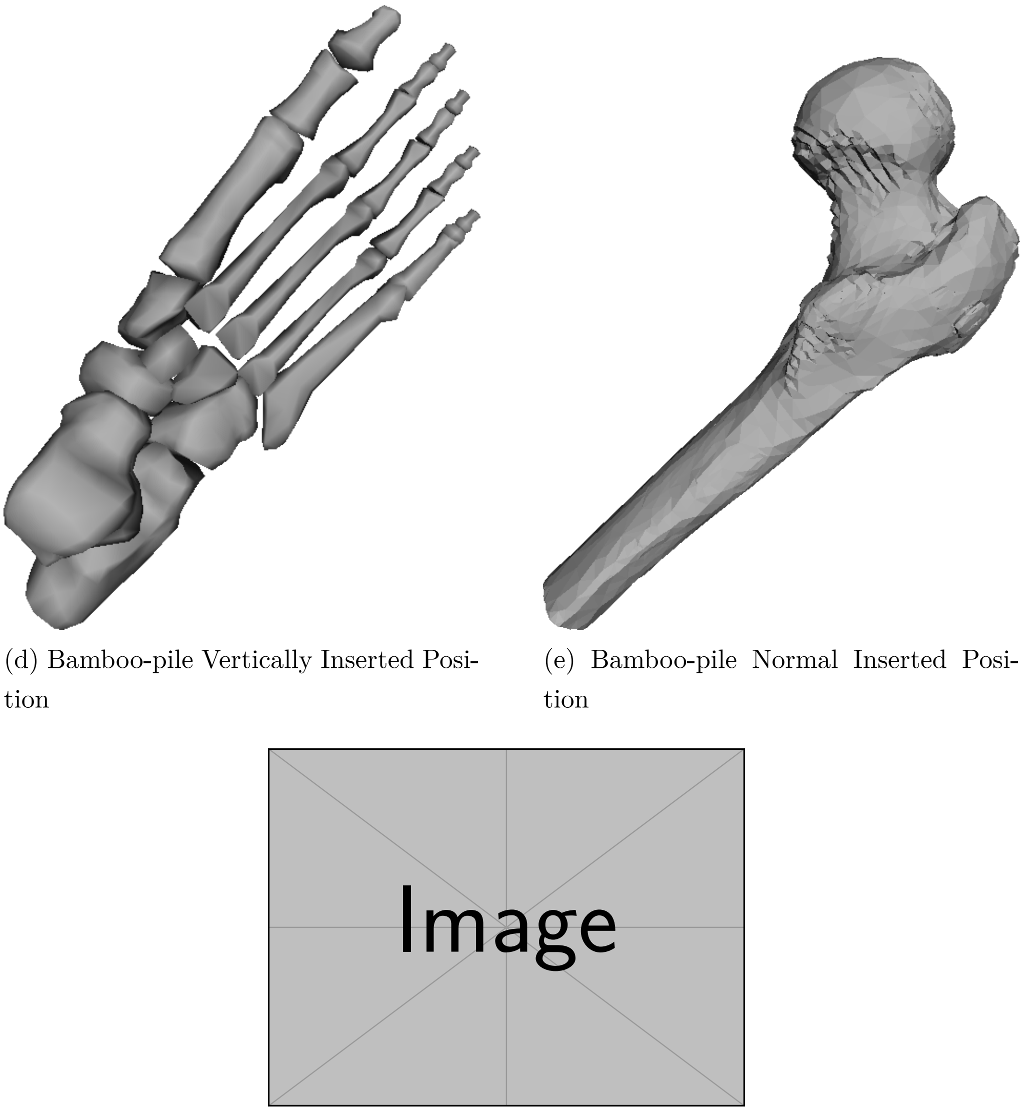


(a) Bamboo-pile Vertically Inserted Position

(b) Bamboo-pile Normal Inserted Position

(c) bamboo-pile Inserted 45° angle

Figure 2.2: A set of six subfigures in two pages.



(d) Bamboo-pile Vertically Inserted Position

(e) Bamboo-pile Normal Inserted Position

(f) bamboo-pile Inserted 45° angle

Figure 2.2: A set of six subfigures in two pages.(Continuation)

3
Proposal

Equation example 1:

$$\begin{aligned} \min_{u} \int_{x_i \in X} \int_{x_j \in X} q_{ij} u_i u_j dada + \int_{x_i \in X} ||x' - x_i|| u_i da \\ s.t. \quad u \in [0,1] \quad \wedge \quad \int_{x_i \in X} uda = a_0, \end{aligned} \tag{3-1}$$

Equation exmaple 2:

$$\begin{aligned} \min_{\mathbf{u}} \alpha \mathbf{u}^T \mathbf{A}^T \mathbf{Q} \mathbf{A} \mathbf{u} + \beta \mathbf{d}^T a' \mathbf{A} \mathbf{u} + \gamma \mathbf{u}^T \mathbf{G}^T \mathbf{G} \mathbf{u} + \delta \mathbf{f}^T a' \mathbf{A} \mathbf{u} \\ s.t. \quad \mathbf{0} \leq \mathbf{u} \leq \mathbf{1} \wedge \mathbf{a}^T \mathbf{u} = a_0. \end{aligned} \tag{3-2}$$

Equation example 3:

$$\mathbf{G} = (g_{ij}) = \begin{cases} \sum_{f_k \in N_f(f_i)} l_{ik} & i = j \\ -l_{ij} & e_{ij} \in E \\ 0 & \text{otherwise} \end{cases} \tag{3-3}$$

Code 1: Mean Filter

```
# --------------------------------------------------------------------#
# Create filter function
# l is the width of window
# --------------------------------------------------------------------#
meanfilter <- function( l, imagem ) {
  if( l%%2 == 0 )
    print("Please, type an odd number!")
  imagem.result <- imagem
  lp1d2 <- (l-1)/2
  L <- dim(imagem)[1]
  C <- dim(imagem)[2]
  for( j in as.integer(lp1d2+1) : as.integer(C-lp1d2)) {
    for( i in as.integer(lp1d2+1) : as.integer(L-lp1d2)) {
      imagem.result[i,j] <- mean(imagem[as.integer(i-lp1d2):as.integer(i+lp1d2), as.integer(j-lp1d2):as.integer(j+lp1d2)])
    }
  }
  print("Image filtred with success!")
  return(imagem.result)
}
#
    ----------------------------------------------------------------------#
# End of Script.
#
    ----------------------------------------------------------------------#
```

Algorithm 1: Escolha das amostras inicias

```
Input: Malha e quantidade de pontos a ser amostrado
Output: Pontos amostrados na malha
Crie um vetor de números randômicos entre $[0,1]$ com a quantidade de pontos a ser amostrada e ordene-o
Calcule a área total dos triângulos da malha
for $i = 0$ to numeroDePontos do
    Navegue entre as faces acumulando a sua $\frac{area}{areaTotal}$ até achar a face com valor acumulado $\geqslant$ numerosRandomicos[i]
    Pegue um ponto randômico dentro da face utilizando o método de Turk e adicione no vetor do resultado
```

4
Results

Table example. Table **??** shows results.

Table 4.1: Results for devil mesh

	Mean Vertex Distance	L2 Vertex Based	Mean Quadric	MSAE	L2 Normal Based	Tangential	Mean Discrete Curvature	Area Error	Volume Error
(**??**)	0.061277	0.110973	0.236219	19.697900	0.055170	0.047678	0.090284	0.051443	0.045645
(**??**)	0.001293	0.002800	0.002289	21.237300	0.021589	0.013026	0.087991	0.000364	0.002621
(**??**)	0.001439	0.002880	0.003540	14.043200	0.012654	0.008911	0.055849	0.007806	0.000582
(**??**)	0.000713	0.001537	0.001824	12.171400	0.009654	0.005781	0.054567	0.005617	0.000425
(**??**)	0.002531	0.004560	0.007108	13.830100	0.017459	0.010314	0.114528	0.001686	0.001786
(**??**)	0.001623	0.003079	0.005048	10.454200	0.015233	0.008054	0.094668	0.002629	0.001326
(**??**)	0.000737	0.001548	0.001493	16.880800	0.014129	0.006974	0.079952	0.000209	0.002375
Ours	0.000987	0.001902	0.002686	11.574200	0.010632	0.006796	0.075106	0.003970	0.000722

4.1
Comparison

5
Conclusion and future work

We proposed an algorithm for triangular mesh denoising with detail preservation...

Code 2: Mean Filter

```
# ---------------------------------------------------------------------#
# Create filter function
# l is the width of window
# ---------------------------------------------------------------------#
meanfilter <- function( l, imagem ) {
  if( l%%2 == 0 )
    print("Please, type an odd number!")
  imagem.result <- imagem
  lp1d2 <- (l-1)/2
  L <- dim(imagem)[1]
  C <- dim(imagem)[2]
  for( j in as.integer(lp1d2+1) : as.integer(C-lp1d2)) {
    for( i in as.integer(lp1d2+1) : as.integer(L-lp1d2)) {
      imagem.result[i,j] <- mean(imagem[as.integer(i-lp1d2):as.integer(i+lp1d2), as.integer(j-lp1d2):as.integer(j+lp1d2)])
    }
  }
  print("Image filtred with success!")
  return(imagem.result)
}
# ---------------------------------------------------------------------#
# End of Script.
# ---------------------------------------------------------------------#
```

6
Bibliography

FLEISHMAN, S.; DRORI, I.; COHEN-OR, D. Bilateral mesh denoising. In: ACM. **ACM transactions on graphics (TOG)**. [S.l.], 2003. v. 22, n. 3, p. 950–953. Cited in page 21.

HE, L.; SCHAEFER, S. Mesh denoising via l 0 minimization. **ACM Transactions on Graphics (TOG)**, ACM, v. 32, n. 4, p. 64, 2013. Cited in page 21.

JONES, T. R.; DURAND, F.; DESBRUN, M. Non-iterative, feature-preserving mesh smoothing. In: ACM. **ACM Transactions on Graphics (TOG)**. [S.l.], 2003. v. 22, n. 3, p. 943–949. Cited in page 21.

PIERCE, B. C. et al. **Logical Foundations**. [S.l.]: Electronic textbook, 2022. v. 1. (Software Foundations, v. 1). Version 6.2, URLhttp://softwarefoundations.cis.upenn.edu. No citation in the text.

SUN, X. et al. Fast and effective feature-preserving mesh denoising. **IEEE transactions on visualization and computer graphics**, IEEE, v. 13, n. 5, p. 925–938, 2007. Cited in page 21.

YADAV, S.; REITEBUCH, U.; POLTHIER, K. Mesh denoising based on normal voting tensor and binary optimization. **arXiv preprint arXiv:1607.07427**, 2016. Cited in page 21.

ZHANG, W. et al. Guided mesh normal filtering. In: WILEY ONLINE LIBRARY. **Computer Graphics Forum**. [S.l.], 2015. v. 34, n. 7, p. 23–34. Cited in page 21.

ZHENG, Y. et al. Bilateral normal filtering for mesh denoising. **IEEE Transactions on Visualization and Computer Graphics**, IEEE, v. 17, n. 10, p. 1521–1530, 2011. Cited in page 21.