Cedille2: A proof theoretic redesign of the Calculus of Dependent Lambda Eliminations

by

Andrew Marmaduke

A thesis submitted in partial fulfillment
of the requirements for the Doctor of Philosophy degree
in Computer Science
in the Graduate College
of The University of Iowa

May 2024

Thesis Committee: Aaron Stump, Thesis Supervisor
Cesare Tinelli
J. Garrett Morris
Sriram Pemmaraju
William J. Bowman

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

– Some wise dude

ACKNOWLEDGMENTS

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

ABSTRACT

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

PUBLIC ABSTRACT

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

Chapter 1

INTRODUCTION

Undergraduate-level description of System F Omega Type Theory as a discipline is a difficult subject to thoroughly introduce because it in essence captures a wide variety of programming languages (if not all programming languages currently defined). To trim the fat this thesis will focus on a particular type theory, System F^ω, and the various bits of machinery that are required to describe it. However, even with this focus there are different equivalent methods of presenting the theory: Pure Type Systems, Martin-Löf style presentations, Bidirectional systems, etc. An extrinsic bidirectional presentation will be used with only summary remarks, if any, for the other styles. This introduction is far from complete and is instead focused on providing the reader with enough background to understand the later chapters.

Describe the syntax including opening/closing/substitution, note that variable bureaucracy is going to be taken for granted in exposition

Describe reduction, multistep reduction (for any predicate), conversion (for any predicate), and how reduction is confluent and transitive, and how reduction is strongly normalizing

Describe the typing rules using a bidirectional system, note that type checking is decidable

Describe Church encodings of data in System F Omega note that they cannot be inductive

Carve out the relevant subsystems, connect them to known notions of logic

Detour about what a proof will mean for us

Describe extension to CC

Add an irrelevant equality and demonstrate how it breaks decidability

Describe, briefly, how Cedille enables inductive encodings through a quotient construction

List the goals of Cedille2 and the remaining structure of the thesis

Chapter 2

THEORY DESCRIPTION AND BASIC METATHEORY

The theory described in this chapter is a variation of the core theory of Cedille [3]. It is closely related with the significant differences occurring with the equality type. This variation has two primary goals. First, to have decidable type checking (and thus decidable conversion checking). Second, to retain as many constructions as possible from Cedille. This chapter focuses on the description of the theory and some basic metatheory. By basic, we mean properties that are provable by induction on the various derivations or are otherwise provable using straightforward methods.

Syntax for the theory is described in Figure 2.1. Unlike other presentations a generic syntax tree is used with a tag to indicate different syntactic forms. There are three basic syntactic constructs: variables, binders, and constructors. A generic presentation enables occasional economic benefits in presenting other derivations. However, a more standard syntax is defined in terms of the generic one. The specific syntactic forms and the generic forms are used interchangeably whichever is more convenient.

Formally, syntax is worked with as a locally nameless set following the axioms of Pitts [1]. For the sake of presentation these details are elided. This means that freshness of variables and capture avoiding substitution are largely taken for granted in the exposition of the theory. Moreover, identity of syntactic terms is assumed to be alpha equivalence. Meaning that, again, bureaucracy around variables is taken for granted. Thus, substitution is defined simply:

$$[x := v]y = v \text{ if } x = y$$

$$[x := v]y = y \text{ if } x \neq y$$

$$[x := v]\mathfrak{b}(\kappa_1, x : t_1, t_2) = \mathfrak{b}(\kappa_1, x : [x := v]t_1, [x := v]t_2)$$

$$[x := v]\mathfrak{c}(\kappa_2, t_1, \dots, t_{\mathfrak{a}(\kappa_2)}) = \mathfrak{c}(\kappa_2, [x := v]t_1, \dots, [x := v]t_{\mathfrak{a}(\kappa_2)})$$

$$
\begin{aligned}
t &::= x \mid \mathfrak{b}(\kappa_1, x : t_1, t_2) \mid \mathfrak{c}(\kappa_2, t_1, \dots, t_{\mathfrak{a}(\kappa_2)}) \\
\kappa_1 &::= \lambda_m \mid \Pi_m \mid \cap \\
\kappa_2 &::= \star \mid \square \mid \bullet_m \mid \mathrm{pair} \mid \mathrm{proj}_1 \mid \mathrm{proj}_2 \mid \mathrm{eq} \mid \mathrm{refl} \mid J \mid \vartheta \mid \delta \mid \phi \\
m &::= \omega \mid 0 \mid \tau \\
&\mathfrak{a}(\star) = \mathfrak{a}(\square) = 0 \\
&\mathfrak{a}(\mathrm{proj}_1) = \mathfrak{a}(\mathrm{proj}_2) = \mathfrak{a}(\mathrm{refl}) = \mathfrak{a}(\vartheta) = \mathfrak{a}(\delta) = 1 \\
&\mathfrak{a}(\bullet_m) = 2 \\
&\mathfrak{a}(\mathrm{pair}) = \mathfrak{a}(\mathrm{eq}) = \mathfrak{a}(\varphi) = 3 \\
&\mathfrak{a}(J) = 6
\end{aligned}
$$

$$
\begin{aligned}
\star &:= \mathfrak{c}(\star) & t.1 &:= \mathfrak{c}(\mathrm{proj}_1, t) \\
\square &:= \mathfrak{c}(\square) & t.2 &:= \mathfrak{c}(\mathrm{proj}_2, t) \\
\lambda_m\, x : t_1.\, t_2 &:= \mathfrak{b}(\lambda_m, x : t_1, t_2) & t_1 =_{t_2} t_3 &:= \mathfrak{c}(\mathrm{eq}, t_1, t_2, t_3) \\
(x : t_1) \to_m t_2 &:= \mathfrak{b}(\Pi_m, x : t_1, t_2) & \mathrm{refl}(t) &:= \mathfrak{c}(\mathrm{refl}, t) \\
(x : t_1) \cap t_2 &:= \mathfrak{b}(\cap, x : t_1, t_2) & \vartheta(t) &:= \mathfrak{c}(\vartheta, t) \\
t_1 \bullet_m t_2 &:= \mathfrak{c}(\bullet_m, t_1, t_2) & \delta(t) &:= \mathfrak{c}(\delta, t) \\
[t_1, t_2, t_3] &:= \mathfrak{c}(\mathrm{pair}, t_1, t_2, t_3) & \varphi(t) &:= \mathfrak{c}(\varphi, t)
\end{aligned}
$$

$$
J(t_1, t_2, t_3, t_4, t_5, t_6) := \mathfrak{c}(J, t_1, t_2, t_3, t_4, t_5, t_6)
$$

Figure 2.1: Generic syntax, there are three constructors, variables, a generic binder, and a generic non-binder. Each are parameterized with a constant tag to specialize to a particular syntactic consruct. The non-binder constructor has a vector of subterms determined by an arity function computed on tags. Standard syntactic constructors are defined in terms of the generic forms.

$$
\begin{aligned}
|x| &= x & |f \bullet_\tau a| &= |f| \bullet_\tau |a| \\
|\star| &= \star & |[t_1, t_2, T]| &= |t_1| \\
|\square| &= \square & |t.1| &= |t| \\
|\lambda_0\, x\!:\!A.\, t| &= |t| & |t.2| &= |t| \\
|\lambda_\omega\, x\!:\!A.\, t| &= \lambda_\omega\, x.\, |t| & |x =_A y| &= |x| =_{|A|} |y| \\
|\lambda_\tau\, x\!:\!A.\, t| &= \lambda_\tau\, x\!:\!|A|.\, |t| & |\mathrm{refl}(t)| &= \lambda_\omega\, x.\, x \\
|(x:A) \to_m B| &= (x:|A|) \to_m |B| & |J(A,P,x,y,e,w)| &= |e| \bullet_\omega |w| \\
|(x:A) \cap B| &= (x:|A|) \cap |B| & |\vartheta(e)| &= |e| \\
|f \bullet_0 a| &= |f| & |\delta(e)| &= |e| \\
|f \bullet_\omega a| &= |f| \bullet_\omega |a| & |\varphi(a,f,e)| &= |a|
\end{aligned}
$$

Figure 2.2: Erasure of syntax, for type-like and kind-like syntax erasure is homomorphic, for term-like syntax erasure reduces to the untyped lambda calculus.

$$
\frac{t_1 \rightsquigarrow_\beta t_1'}{\mathfrak{b}(\kappa, x : t_1, t_2) \rightsquigarrow_\beta \mathfrak{b}(\kappa, x : t_1', t_2)}
\qquad
\frac{t_2 \rightsquigarrow_\beta t_2'}{\mathfrak{b}(\kappa, x : t_1, t_2) \rightsquigarrow_\beta \mathfrak{b}(\kappa, x : t_1, t_2')}
$$

$$
\frac{t_i \rightsquigarrow_\beta t_i' \qquad i \in 1, \ldots, \mathfrak{a}(\kappa)}{\mathfrak{c}(\kappa, t_1, \ldots t_i, \ldots t_{\mathfrak{a}(\kappa)}) \rightsquigarrow_\beta \mathfrak{c}(\kappa, t_1, \ldots t_i', \ldots t_{\mathfrak{a}(\kappa)})}
$$

$$
\begin{aligned}
(\lambda_m\, x\!:\!A.\, b) \bullet_m t &\rightsquigarrow_\beta [x := t]b \\
[t_1, t_2, A].1 &\rightsquigarrow_\beta t_1 \\
[t_1, t_2, A].2 &\rightsquigarrow_\beta t_2 \\
J(A, P, x, y, \mathrm{refl}(z), w) &\rightsquigarrow_\beta w \bullet_0 z \\
\vartheta(\mathrm{refl}(t.1)) &\rightsquigarrow_\beta \mathrm{refl}(t) \\
\vartheta(\mathrm{refl}(t.2)) &\rightsquigarrow_\beta \mathrm{refl}(t) \\
\varphi(a, f, e).1 &\rightsquigarrow_\beta a
\end{aligned}
$$

Figure 2.3: Reduction rules for arbitrary syntax.

$$\begin{aligned} \mathrm{dom}_\Pi(\omega, K) &= \star & \qquad \mathrm{codom}_\Pi(\omega) &= \star \\ \mathrm{dom}_\Pi(\tau, K) &= K & \qquad \mathrm{codom}_\Pi(\tau) &= \Box \\ \mathrm{dom}_\Pi(0, K) &= K & \qquad \mathrm{codom}_\Pi(0) &= \star \end{aligned}$$

Figure 2.4: Domain and codomains for function types. The variable K is either $\star$ or $\Box$.

$$\frac{\vdash \Gamma}{\Gamma \vdash \star \rhd \Box}\ \textsc{Axiom} \qquad \frac{\vdash \Gamma \qquad (x : A) \in \Gamma}{\Gamma \vdash x \rhd A}\ \textsc{Var}$$

$$\frac{\Gamma \vdash t \rhd A \qquad A \leadsto_\beta^* B}{\Gamma \vdash t \mathrel{|\!\!\rhd} B}\ \textsc{HdInf} \qquad \frac{\begin{array}{c}\Gamma \vdash t \rhd A \\ \Gamma \vdash B \mathrel{|\!\!\rhd} K \qquad A \equiv B\end{array}}{\Gamma \vdash t \lhd B}\ \textsc{Chk}$$

$$\frac{}{\vdash \varepsilon}\ \textsc{CtxEm} \qquad \frac{\begin{array}{c}x \notin \mathrm{FV}(\Gamma) \\ \vdash \Gamma \qquad \Gamma \vdash A \mathrel{|\!\!\rhd} K\end{array}}{\vdash \Gamma, x : A}\ \textsc{CtxApp}$$

$$\frac{\Gamma \vdash A \mathrel{|\!\!\rhd} \mathrm{dom}_\Pi(m, K) \qquad \Gamma, x : A \vdash B \mathrel{|\!\!\rhd} \mathrm{codom}_\Pi(m)}{\Gamma \vdash (x : A) \to_m B \rhd \mathrm{codom}_\Pi(m)}\ \textsc{Pi}$$

$$\frac{\Gamma \vdash A \mathrel{|\!\!\rhd} \mathrm{dom}_\Pi(m, K) \qquad \Gamma, x : A \vdash t \rhd B \qquad x \notin \mathrm{FV}(|t|) \text{ if } m = 0}{\Gamma \vdash \lambda_m\, x{:}A.\, t : (x : A) \to_m B}\ \textsc{Lam}$$

$$\frac{\Gamma \vdash f \mathrel{|\!\!\rhd} (x : A) \to_m B \qquad \Gamma \vdash a \lhd A}{\Gamma \vdash f \bullet_m a \rhd [x := a]B}\ \textsc{App}$$

Figure 2.5: Inference rules for function types, including erased functions. The variable K is either $\star$ or $\Box$.

$$\frac{\Gamma \vdash A \mathrel{\triangleright\!\!\!\triangleright} \star \qquad \Gamma, x : A \vdash B \mathrel{\triangleright\!\!\!\triangleright} \star}{\Gamma \vdash (x : A) \cap B \triangleright \star}\ \textsc{Int}$$

$$\frac{\begin{array}{c}\Gamma \vdash T \mathrel{\triangleright\!\!\!\triangleright} (x : A) \to_\tau B \\ \Gamma \vdash t \triangleleft A \\ \Gamma \vdash s \triangleleft [x := t]B \qquad |t| =_\beta |s|\end{array}}{\Gamma \vdash [t, s; T] \triangleright (x : A) \cap B}\ \textsc{Pair}$$

$$\frac{\Gamma \vdash t \mathrel{\triangleright\!\!\!\triangleright} (x : A) \cap B}{\Gamma \vdash t.1 \triangleright A}\ \textsc{Fst}$$

$$\frac{\Gamma \vdash t \mathrel{\triangleright\!\!\!\triangleright} (x : A) \cap B}{\Gamma \vdash t.2 \triangleright [x := t.1]B}\ \textsc{Snd}$$

Figure 2.6: Inference rules for intersection types.

$$\frac{\begin{array}{c}\Gamma \vdash A \mathrel{\triangleright\!\!\!\triangleright} \star \\ \Gamma \vdash a \triangleleft A \qquad \Gamma \vdash b \triangleleft A\end{array}}{\Gamma \vdash a =_A b \triangleright \star}\ \textsc{Eq}$$

$$\frac{\Gamma \vdash t \triangleright A}{\Gamma \vdash \mathrm{refl}(t) \triangleright t =_A t}\ \textsc{Refl}$$

$$\frac{\begin{array}{c}\Gamma \vdash A \mathrel{\triangleright\!\!\!\triangleright} \star \\ \Gamma \vdash P \triangleleft (x\ y : A) \to_\tau (e : x =_A y) \to_\tau \star \qquad \Gamma \vdash x \triangleleft A \qquad \Gamma \vdash y \triangleleft A \\ \Gamma \vdash e \triangleleft x =_A y \qquad \Gamma \vdash w \triangleleft (a : A) \to_0 P \bullet_\tau a \bullet_\tau a \bullet_\tau \mathrm{refl}(a)\end{array}}{\Gamma \vdash J(A, P, x, y, e, w) \triangleright P \bullet_\tau x \bullet_\tau y \bullet_\tau e}\ \mathrm{J}$$

$$\frac{\Gamma \vdash e \mathrel{\triangleright\!\!\!\triangleright} a.i =_T b.j \qquad \Gamma \vdash a \mathrel{\triangleright\!\!\!\triangleright} (x : A) \cap B \qquad \Gamma \vdash b \triangleleft (x : A) \cap B}{\Gamma \vdash \vartheta(e) \triangleright a =_{(x:A)\cap B} b}\ \textsc{Prm}$$

$$\frac{\begin{array}{c}\Gamma \vdash f \mathrel{\triangleright\!\!\!\triangleright} (a : A) \to_\omega (x : A) \cap B \\ \Gamma \vdash a \triangleleft A \qquad \Gamma \vdash e \triangleleft (a : A) \to_\omega a =_A (f \bullet_\omega a).1 \qquad \mathrm{FV}(|e|) = \varnothing\end{array}}{\Gamma \vdash \varphi(a, f, e) \triangleright (x : A) \cap B}\ \textsc{Cast}$$

$$\frac{\Gamma \vdash e \triangleleft \mathrm{ctt} =_{\mathrm{cBool}} \mathrm{cff}}{\Gamma \vdash \delta(e) \triangleright (X : \star) \to_0 X}\ \textsc{Sep}$$

Figure 2.7: Inference rules for equality types where cBool $:= (X : \star) \to_0 (x : X) \to_\omega (y : X) \to_\omega X$; ctt $:= \lambda_0\, X : \star.\, \lambda_\omega\, x : X.\, \lambda_\omega\, y : X.\, x$; and cff $:= \lambda_0\, X : \star.\, \lambda_\omega\, x : X.\, \lambda_\omega\, y : X.\, y$. Also, $i, j \in \{1, 2\}$

Chapter 3

PROOF NORMALIZATION AND RELATIONSHIP TO SYSTEM $\mathbf{F}^{\omega}$

Chapter 4

CONSISTENCY AND RELATIONSHIP TO CDLE

Chapter 5

OBJECT NORMALIZATION

A φ_i-proof is a proof that allows i nested φ syntactic constructs. For example, a φ_0-proof allows no φ subterms, a φ_1-proof allows φ subterms but no nested φ subterms, and a φ_2-proof allows φ_1 subterms. Defined inductively, a φ_0-proof is a proof with no φ syntactic constructs and a φ_{i+1}-proof is a proof with φ_i-proof subterms.

For any φ_i-proof p there is a strictification $s(p)$ that is a φ_0-proof in Figure 5.1.

Lemma 1 (Strictification Preserves Inference)**.** *Given* $\Gamma \vdash t \rhd A$ *then* $\Gamma \vdash s(t) \rhd A$

Proof. By induction on the typing rule, the φ rule is the only one of interest:

Case:
$$\frac{\overset{\mathcal{D}_1}{\Gamma \vdash f \mathrel{|\!\!\rhd} (a : A) \to_\omega (x : A) \cap B} \qquad \overset{\mathcal{D}_2}{\Gamma \vdash a \lhd A} \quad \overset{\mathcal{D}_3}{\Gamma \vdash e \lhd (a : A) \to_\omega a =_A (f \bullet_\omega a).1} \quad \overset{\mathcal{D}_4}{\mathrm{FV}(|e|) = \varnothing}}{\Gamma \vdash \varphi(a, f, e) \rhd (x : A) \cap B}$$

Need to show that $\Gamma \vdash s(\varphi(a, f, e)) \rhd (x : A) \cap B$ which reduces to: $\Gamma \vdash s(f) \bullet_\omega s(a) \rhd (x : A) \cap B$. By the IH we know that $s(f)$ infers the same function type, and that $s(a)$ infers the same argument type, therefore the application rule concludes the proof.

□

Lemma 2 (Strict Proofs are Normalizing)**.** *Given* $\Gamma \vdash t \rhd A$ *then* $s(t)$ *is strongly normalizing*

Proof. Direct consequence of strong normalization of proofs □

$$
\begin{aligned}
s(x) &= x & s([s,t,T]) &= [s(s), s(t), s(T)] \\
s(\star) &= \star & s(t.1) &= s(t).1 \\
s(\square) &= \square & s(t.2) &= s(t).2 \\
s(\lambda_m\, x\!:\!A.\, t) &= \lambda_m\, x\!:\!s(A).\, s(t) & s(x =_A y) &= s(x) =_{s(A)} s(y) \\
s((x:A) \to_m B) &= (x:s(A)) \to_m s(B) & s(\mathrm{refl}(t)) &= \mathrm{refl}(s(t)) \\
s((x:A) \cap B) &= (x:s(A)) \cap s(B) & s(\vartheta(e)) &= \vartheta(s(e)) \\
s(f \bullet_m a) &= s(f) \bullet_m s(a) & s(\delta(e)) &= \delta(s(e))
\end{aligned}
$$

$$
\begin{aligned}
s(J(A,P,x,y,r,w)) &= J(s(A), s(P), s(x), s(y), s(r), s(w)) \\
s(\varphi(a,f,e)) &= s(f) \bullet_\omega s(a)
\end{aligned}
$$

Figure 5.1: Strictification of a proof.

Lemma 3 (Strict Objects are Normalizing)**.** *Given* $\Gamma \vdash t \rhd A$ *then* $|s(t)|$ *is strongly normalizing*

Proof. Proof Idea:

Proof reduction tracks object reduction in the absence of φ constructs. Thus, the normalization of a proof provides an upper-bound on the number of reductions an object can take to reach a normal form. $\square$

A proof, $\Gamma \vdash t_1 \rhd A$, is contextually equivalent to another proof, $\Gamma \vdash t_2 \rhd A$, if there is no context with hole of type A whose object reduction diverges for t_1 but not t_2. In other words, if a context can be constructed that distinguishes the terms based on their object reduction.

Lemma 4. *A* φ_1*-proof,* p*, is contextually equivalent to its strictification,* $s(p)$

Proof. Proof by induction on the typing rule for p, focus on the application rule:

Case: $$\frac{\overset{\mathcal{D}_1}{\Gamma \vdash f \mathrel{\unrhd\!\!\rhd} (x:A) \to_m B} \qquad \overset{\mathcal{D}_2}{\Gamma \vdash a \lhd A}}{\Gamma \vdash f \bullet_m a \rhd [x := a]B}$$

In particular, we care about when $f = \varphi(v, b, e).2$ and $m = \omega$. Note that the first projection has a proof-reduction that yields a which makes it unproblematic.

We know that $s(v) = v$ because f is a φ_1-proof. Let v_n be the normal form of v and note that $|v_n|$ is also normal. Likewise, we have e_n and $|e_n|$ normal.

Suppose there is a context $C[\cdot]$ where $|p|$ diverges but $|s(p)|$ normalizes. (Note that the opposite assumption is impossible). If $|v_n|$ is a variable, then reduction in $|p|$ is blocked (contradiction). Otherwise $|v_n| = \lambda\, x.\, x\ t_1\ \cdots\ t_n$ where t_i are normal.

Now it must be the case that $|e \bullet_\omega v| = |e_n| \bullet_\omega |v_n|$ is normalizing. Thus, we have a refl proof that $v_n = (f \bullet_\omega v_n).1$. (Note, this proof *must* be refl because $\mathrm{FV}(|e|) = \varnothing$). But, this implies convertibility, thus $|v_n| =_\beta |f| \bullet_\omega |v_n|$, but this must mean more concretely that $|f| \bullet_\omega |v_n| \rightsquigarrow_\beta |v_n|$. Yet $|f| \bullet_\omega |v_n| \bullet_\omega a$ is strongly normalizing because it is $s(p)$. Therefore, p in this case is strongly normalizing which refutes the assumption yielding a contradiction.

□

Lemma 5. *If t_1 is strongly normalizing and contextually equivalent to t_2 then t_2 is strongly normalizing*

Proof. Immediate by the definition of contextual equivalence. □

Theorem 1. *A φ_i-proof p is strongly normalizing for all i*

Proof. By induction on i.

Case: $i = 0$

Immediate because $s(p) = p$ and strict proofs are strongly normalizing.

Inductive Case:

Suppose that φ_i-proof is strongly normalizing. Goal: show that φ_{i+1}-proof is strongly normalizing.

$\square$

Chapter 6

CEDILLE2: SYSTEM IMPLEMENTATION

Chapter 7

CEDILLE2: INTERNALLY DERIVABLE CONCEPTS

Chapter 8

CONCLUSION AND FUTURE WORK

Appendix A

PROOFS OF CHAPTER 1

Appendix B

UNDECIDED

Hello!

BIBLIOGRAPHY

[1] Andrew M. Pitts. "Locally Nameless Sets". In: *Proc. ACM Program. Lang.* 7.POPL (Jan. 2023). DOI: 10.1145/3571210. URL: https://doi.org/10.1145/3571210.

[2] Aaron Stump. "The calculus of dependent lambda eliminations". In: *Journal of Functional Programming* 27 (2017), e14.

[3] Aaron Stump and Christopher Jenkins. *Syntax and Semantics of Cedille.* 2021. arXiv: 1806.04709 [cs.PL].