THE UNIVERSITY OF CHICAGO

WEAK AND STRONG NORMALIZATION OF TIERED PURE TYPE SYSTEMS VIA TYPE-PRESERVING TRANSLATION

A DISSERTATION SUBMITTED TO
THE FACULTY OF THE DIVISION OF THE PHYSICAL SCIENCES
IN CANDIDACY FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

BY

NATHAN MULL

CHICAGO, ILLINOIS

JUNE 16, 2023

To Mom and Dad

You may forget but

let me tell you
this: someone in
some future time
will think of us

-Sappho

TABLE OF CONTENTS

ACKNOWLEDGMENTS

Forthcoming.

ABSTRACT

A type system is weakly normalizing if every typable expression has a normal form, and is strongly normalizing if no typable expression appears in an infinite reduction sequence. The Barendregt-Geuvers-Klop conjecture asks if weak normalization implies strong normalization for all *pure type systems*, a class of systems which generalizes the λ-cube. There are two natural techniques based on type-preserving translations for proving strong normalization from weak normalization. The first is to define a translation from expressions to I-expressions (*i.e.*, expressions in which λ-bound variables *must* be used). Strong normalization then readily follows from weak normalization by a conservation result along the lines of the one proved by Church [1941] for the untyped λ-calculus. This was first done by Xi [1996] and Sørensen [1997] for a subset of the λ-cube, and was generalized to a class of pure type systems by Barthe et al. [2001]. The second technique is to define an infinite-reduction-path preserving translation to a weak sub-system for which the conjecture is known to hold. By a straightforward boot-strapping argument, the conjecture can be shown to hold for the full system. Translations of this form were first presented by Geuvers and Nederhof [1991] and Harper et al. [1993] for systems in the λ-cube, and similar techniques were used by Roux and Doorn [2014] for a class of pure type systems.

These syntactic techniques have the further benefit that they are proof theoretically weak, and so can be used to address the stronger form of the conjecture noted by Xi [1997], which asks if the proof of strong normalization from weak normalization can be carried out in Peano arithmetic (PA), or even Heyting arithmetic (HA). This is notable insofar as Girard [1972] proved that the strong normalization of the very simple pure type system $\lambda 2$ implies the consistency of PA within PA, so these results imply a proof-theoretic impossibility results.

In what follows, I present three translations, one of the first form and two of the second form. The first is a generalization of the thunkification translation of Xi [1997], which maps expressions to I-expressions using a more fine-tuned padding scheme than the one used

by Barthe et al. [2001], leading to fewer technical restrictions on the class of systems to which it applies. The second is a generalization of the dependency-eliminating translation of Geuvers and Nederhof [1991] to a class of pure type systems which are known to be strongly normalizing, but for which the stronger form of the conjecture is demonstrated for the first time. The third is a novel application of ideas from Roux and Doorn [2014] (and, consequently, from Geuvers and Nederhof [1991]) for a class of pure type systems with "irrelevant" structure. These systems have properties like dependent types (albeit, a very weak notion of dependent types) which have not appeared in any of the classes of systems previously considered.

All of this work is done in the framework of *tiered pure type systems*, a simple class of persistent pure type systems which are concretely specified but sufficient to study in most cases regarding questions of normalization. I consider the introduction of this class of systems one of the more important contributions of this work, as their combinatorial nature makes them easier to study than previously considered classes of systems, and allows for better classification of systems for which current techniques do not apply. They encompass in some sense the simplest extension of the λ-cube which have the desired meta-theoretic properties, while also leaving much to be further developed, and I believe the conjecture restricted to this class of systems strikes well the balance between challenging and approachable.

CHAPTER 1

INTRODUCTION

This dissertation is concerned with the relationship between weak and strong normalization of pure type systems, a class of typed λ-calculi. I'd like to begin by motivating said concern with a very brief (read: caricatured) historical outline of the relevant material. As much as I would like the following material to be based in first principles, the pedagogical problem of discussing type theory without presenting a course-worth of material so far eludes me, and thus I will be assuming general familiarity with basic logical concepts.

1.0.1 The Untyped λ-Calculus

Church [1941] introduced what is now called the *untyped λ-calculus* as a theory of "functions as rules"[1] (*e.g.*, the function "given x, add 1 to x") as opposed to functions as graphs in the set-theoretic tradition (*e.g.*, the "function" $\{(x, 1+x) : x \in \mathbb{N}\}$). It is an equational theory over expressions from T, defined inductively as follows:

- $\mathsf{V} \subset \mathsf{T}$;
- $x \in \mathsf{V}$ and $M \in \mathsf{T}$ implies $\lambda x.\ M$;
- $M \in \mathsf{T}$ and $N \in \mathsf{T}$ implies $MN \in \mathsf{T}$;

where V is a fixed set of variable symbols.[2] The construct "$\lambda x.\ M$" is referred to as *abstraction* over the variable x, and is thought of as yielding a function on x (*e.g.*, abstracting over x in the expression "add 1 to x" yields the function "given x, add 1 to x"). The construct "MN" is referred to as *application* of one expression to another, and it has been noted by

1. This phrase is quoted by Barendregt [1984], who does not give a citation, but refers to it as "old fashioned," and so must have pulled it from the rarefied air of those times.

2. I take the usual liberty of being agnostic regarding the ontological status of "symbols"

many (Barendregt [1984] included) that this is a mild misnomer, as application in this setting is a syntactic notion and should be thought of as yielding the application itself, not the value of applying a function to its argument (*e.g.*, "add 1 to x, applied to 2" as opposed to "3"). In subsequent exposition, I will instead use Backus-Naur form grammars to describe inductively defined expressions, *e.g.*

$$\mathsf{T} ::= \mathsf{V} \mid \lambda \mathsf{V}.\ \mathsf{T} \mid \mathsf{T}\mathsf{T}$$

The theory has at its core an interest in *computation*—the notion of rules invoking more computational associations than graphs[3]—and describes a convertibility relation via a single computational rule which, for historical reasons, is called β:

$$(\lambda x.\ M)N = M[N/x]$$

where $M[N/x]$ is the result of substituting N for x in M (eliding the nasty details of substitution with which seasoned readers are keenly aware). This relation is lifted compatibly through the structure of expressions, *i.e.*, β-reductions can be done anywhere in the expression, not just at the top level; pictorially speaking:

$$(\dots(\lambda x.\ M)N\dots) = (\dots M[N/x]\dots)$$

But formally speaking, the theory is the equational presentation of the compatible equivalence closure of β; viewed as a relation,

$$\beta = \{((\lambda x.\ M)N, M[N/x]) \mid M, N \in \mathsf{T}\}$$

3. It seems worth noting that students are so often confused when we refer to graphs as functions, since they look nothing like the functions of our imaginations, at least for the seeming majority of us.

and its compatibility closure $\to_\beta$ is the smallest relation containing β such that $M \to_\beta N$ implies

$$\lambda x.\ M \to_\beta \lambda x.\ N$$

$$MP \to_\beta NP$$

$$PM \to_\beta PN$$

I have avoided to this point discussing any tricky technicalities in the definition of this theory, but I will note now that expressions are considered equal *up to α-conversion*, or the renaming of bound variables (so, for example, $\lambda x.\ \lambda y.\ x = \lambda z.\ \lambda w.\ z$) and I will observe the *Barendregt variable convention*, which states that when writing down an expression, bound variables are chosen to be distinct from free variables and from each other (a standard *proviso* for logical syntax).

There is a rich line of work regarding the λ-calculus as a logical theory (see, *e.g.*, the encyclopedic text by Barendregt [1984]), but it is sufficient for the purposes of this dissertation to consider the λ-calculus as an *abstract reduction system*. And for the other systems considered in this introduction, I will ignore the associated equational theory altogether, and focus strictly on the associated reduction system as well.

1.0.2 Abstract Reduction Systems

An **abstract reduction system** is simply a set X together with a binary relation R on X, with the "attitude"[4] of viewing R as a notion of reduction, *i.e.*, xRy means x *reduces to* y.

Notation and Terminology. Fix a reduction system (X, R).

- $\to_R$ stands for R.

4. I recently discovered the nLab [2023] page for the notion of a "concept with an attitude" which is used to describe the added psychological structure implicit in a mathematical object I quite like this term.

- $\twoheadrightarrow_R^+$ stands for the transitive closure of R.
- $\twoheadrightarrow_R$ stands for the reflexive transitive closure of R, and $x \twoheadrightarrow_R y$ also reads as x **reduces to** y.
- $=_R$ stands for the equivalence closure of R.
- A **reduction sequence** is a potentially infinite sequence of elements $x_1, x_2, \ldots$ such that

  $$x_1 \to_R x_2 \to_R \cdots$$

  It is straightforward to verify that $x \twoheadrightarrow_R y$ if there is a finite reduction sequence starting at x and ending in y, and that $x \twoheadrightarrow_R^+$ if there is a non-empty such sequence.
- An element y is an R**-normal form** (or is in R-normal form) if it is irreducible, *i.e.*, there is no element z such that yRz. It is an R-normal form of x if $x \twoheadrightarrow_R y$.

Properties. There are multitudes one may wish to verify of a given reduction system, but I restrict myself to the core four of interest here (and really, only the last two will play a substantial role outside of this introduction).

- **Church-Rosser.** For every element x and y, if $x =_R y$, then there is an element z such that $x \twoheadrightarrow_R z$ and $y \twoheadrightarrow_R z$.
- **Unique Normal Forms.** For every element x, if y and z are normal forms of x then $y = z$.
- **Weak Normalization.** Every element of X has an R-normal form. A particular element is (weakly) normalizing if it has an R-normal form.
- **Strong Normalization.** No element of X appears in an infinite reduction sequence. A particular element x is strongly normalizing if there is no reduction sequence starting at x.

The study of abstract reduction systems postdates the study of the λ-calculus, with $(\mathsf{T}, \rightarrow_\beta)$ being an oft cited example of a prototypical reduction system (see, *e.g.*, the exposition of Bezem et al. [2003]). Church and Rosser [1936] proved that this system satisfies the first eponymous property, as well as the second (up to α-equivalence). From this, it is possible to show that normalizing β-equivalent expressions reduce to the same normal form, which indicates that β-reduction captures a reasonable notion of computation. The last two properties—which are the focus of this dissertation—do not hold of $(\mathsf{T}, \rightarrow_\beta)$. The expression

$$\Omega \triangleq (\lambda x.\ xx)(\lambda x.\ xx)$$

has no normal form since $\Omega \rightarrow_\beta \Omega$ and this is the only reduction that can be performed on Ω. Even the sub-system restricted to the weakly normalizing expressions is not strongly normalizing; given

$$I \triangleq \lambda x.\ x$$

$$K \triangleq \lambda x.\ \lambda y.\ x$$

$KI\Omega \rightarrow_\beta I$ and I is a normal form but $KI\Omega \rightarrow_\beta KI\Omega \rightarrow_\beta KI\Omega \ldots$

1.0.3 The λI-Calculus

The previous remark is perhaps unsurprising; it seems a mere platitude to note that strong normalization is a stronger notion than weak normalization. But it is natural to consider under what conditions it is not.

The λI-calculus is the fragment of the λ-calculus in which every expression of the form $\lambda x.\ M$ is required to have x appearing in M. I will use $\mathsf{T}_{\lambda I}$ to denote the set of such expressions. The expression K, for example, is not in $\mathsf{T}_{\lambda I}$. It can then be seen that the reason $KI\Omega$ is weakly normalizing but not strongly normalizing is that K can "throw away"

its non-normalizing argument, *i.e.*, that K is, in fact, not in $\mathsf{T}_{\lambda I}$. Church [1941] proves that the inability to do this is a sufficient condition for weak and strong normalization to coincide.

Theorem 1. *(Conservation for λI, Church [1941]) For any λI-expression M, if M is weakly normalizing then M is strongly normalizing.*

This is, in a loose sense, a reverse mathematical result. It may be equivalently stated that every λI-expression is either non-normalizing or strongly normalizing, but the more interesting reading (at least to me) is that weak normalizing is a sufficient assumption for proving strong normalization. Or, with a more constructive bent, a single reduction path may be used to reason about all reduction paths. So from the perspective of reduction systems, and with an eye towards the next section, in order to show that a sub-system of $(T_{\lambda I}, \rightarrow_\beta)$[5] is strongly normalizing, it is sufficient to show that it is weakly normalizing.

It may be of some interest to the historically inclined that λI-calculus was the first calculus presented by Church [1941], but the calculus as we known it now (sometimes referred to as the λK-calculus) became the dominant form. Barendregt [1984] (pp. 38) cites some potential reasons for the shift.

1.0.4 The Simply Typed λ-Calculus

Church [1940] introduced the simply typed λ-calculus—here on out denoted by $\lambda_\rightarrow$—as an extension of the simple types of Russell and Whitehead [1910, 1912, 1913] to include notions from the λ-calculus. In its modern form, as presented here, the grammar of types is given by

$$\mathsf{Ty} ::= \mathsf{B} \mid \mathsf{Ty} \rightarrow \mathsf{Ty}$$

where B is a set of base types. The construct $A \rightarrow B$ is a *function type* and is used to describe expressions which behave like functions. In the tradition of Church-style typing, we

5. Here and in future iterations, I allow the reduction relation to be defined on a super-set, so as not to require notation for restricting the relation.

work over a slightly different expressions grammar

$$\mathsf{T} ::= \mathsf{V} \mid \lambda \mathsf{V}^{\mathsf{Ty}}.\ \mathsf{T} \mid \mathsf{TT}$$

in which abstracted variables are annotated with types. Typing judgment of the form

$$x_1 : A_1, \ldots, x_k : A_k \vdash M : A$$

read as

"x_1 is of type A_1, ..., x_k is of type A_k implies M is of type A"

are then used to delineate a class of "well-behaved" expressions. Roughly speaking, type acts as a simple syntactic description of their semantic behavior. The rules for building typing judgments are as one might expect if we understand function types to delineate the expressions we presume to behave like functions.

Typing Judgments. A variable x is **fresh** if it does not appear in on the left side of the turnstile.

- **Start.** For any type A (in Ty) and any variable x

$$x : A \vdash x : A$$

- **Variable Introduction.** For any type B and variable x which is fresh with respect to Γ

$$\frac{\Gamma \vdash M : A}{\Gamma, x : B \vdash x : B}$$

- **Weakening.** For any type B and variables x which is fresh with respect to Γ

$$\frac{\Gamma \vdash M : A}{\Gamma, x : B \vdash M : A}$$

- **Abstraction.**

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x^A.\ M : A \to B}$$

- **Application.**

$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B}$$

The rules as presented are non-standard, but better mirror the shape of the rules for general pure type systems considered in the next section. Despite this, I hope that they represent natural intuitions about how function types should behave. For example, if M is a function of type $A \to B$, and N is of type A, then it is well-defined to apply M to N and we should expect the result to be of type B. For a more concrete example, if $\mathsf{Nat} \in \mathsf{B}$, then it is possible to derive

$$Z : \mathsf{Nat}, S : \mathsf{Nat} \to \mathsf{Nat}, A : \mathsf{Nat} \to (\mathsf{Nat} \to \mathsf{Nat}) \vdash \lambda x^{\mathsf{Nat}}.\ A(SZ)x : \mathsf{Nat} \to \mathsf{Nat}$$

a slightly obfuscated but potentially familiar example.

Given these typing rules, we can then consider the rewrite system $(\mathsf{T}_{\lambda\to}, \to_\beta)$ where $\mathsf{T}_{\lambda\to}$ is the subset of expressions which are typable according to the above rules. Turing, in the 1940s, was the first (as recorded by Gandy [1980]) to prove that this system is weakly normalizing via a measure on expressions which is monotonically decreasing in a particular reduction strategy. It would take several decades before Tait [1967] proved it is strongly normalizing via a more complicated proof which, in essence, uses logical relations. Xi [1996] and Sørensen [1997] recognized that, since the proof of weak normalization is a fair bit simpler, it would be reasonable to try to derive strong normalization from weak normalization, and both achieve this via a translation into a system which has a conservation theorem like the one for the λI-calculus. Again, we have a reverse mathematical question: is weak normalization of $(\mathsf{T}_{\lambda\to}, \to_\beta)$ a sufficient condition to more easily prove its strong normalization?

1.0.5 Beyond Simple Types

In the unlikely event that I have piqued the interest of a non-expert, I recommend instead reading any of the great introductory texts in the field (*Type Theory and Formal Proof* by Nederpelt and Geuvers [2014] for example). What remains of this introduction will be informal and cursory.

The decades after the introduction of simple types saw an explosion of type theories for logic, linguistics and computer science. Type systems may be fancified with a number of new constructions, shapes, annotations, etc. but there were three features of particular importance, which were noted by Barendregt [1991] to share the property that they blurred the line between types and terms.

- **Polymorphism.** Or abstraction over type variables in terms, which allows for functions which are, in a sense, agnostic to the type of their argument. The expression $\lambda A^{\mathsf{Type}}.\ \lambda x^{A}.\ x$ can be applied to different types to yield different typed identity functions.

- **Type Constructors.** Or abstraction over type variables in types, which allows for functions that can return types themselves. Given an empty type $\bot$, it may be useful to write the function $\lambda A^{\mathsf{Type}}.\ A \to \bot$, which returns the negated form the type.

- **Dependent Types.** Or abstraction over term variables in types. Given dependent types, it is possible to define predicates of types like $\mathsf{Nat} \to \mathsf{Type}$, for which inhabiting terms are evidence that the predicate holds on its argument.

These three features form the "basis" of a class of systems called the λ-cube, introduced by Barendregt [1991], which uniformly describes these three features in single type system. There are a number of proofs of strong normalization for every system in the λ-cube, so the relationship between weak normalization and strong normalization is determined. I chose to not present these systems, in part because they are so similar to *pure types systems* which

are presented in the next section, and are a natural extension of the λ-cube. I instead refer the reader to the survey by Barendregt [1993]. Pure type systems including the inconsistent system λU of Girard [1972], leaving the relationship between weak normalization and strong normalization unclear. For any pure type system, we can play the same game as above by considering the reduction system associated with typable expressions in the system. This is the core of the Barendregt-Geuvers-Klop conjecture:

Weak normalization implies strong normalization for all pure types systems.

Given this conjecture remains unsolved, I consider it as being truly motivated by two questions.

- First, the same reverse mathematical question as above: is weak normalization a sufficiently strong assumption to prove strong normalization? This question takes on a more interesting role here in that it is known that not all pure type systems are weakly normalizing. Furthermore, Girard [1972] proved that strong normalization of $\lambda 2$ implies the consistency of Peano Arithmetic, so the reverse mathematical strength can be better calibrated, as is done in the *strong form* of the conjecture, which asks if the proof of strong normalization from weak normalization can be carried out in Peano arithmetic, or even Heyting arithmetic.

- Falsifying this conjecture would in essence require finding a system with an expression that is weakly normalizing but not strongly normalizing. What would such an expression look like? The expression $KI\Omega$ in the untyped λ-calculus hides a non-normalizing expression which could not be typable in a normalizing type system. There are examples of more complicated λ-expressions using fixed-points which have the property that every sub-expression is weakly normalizing, but it is unknown if such fixed-points are typable.

At this point, I transition into more formal definitions. I will conclude the introduction with an outline of the contributions of this dissertation to this question.

1.1 Definitions

A pure type system is specified by a triple of sets $(\mathcal{S}, \mathcal{A}, \mathcal{R})$ satisfying $\mathcal{A} \subset \mathcal{S} \times \mathcal{S}$ and $\mathcal{R} \subset \mathcal{S} \times \mathcal{S} \times \mathcal{S}$. The elements of $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{R}$ are called sorts, axioms and rules, respectively. I use s and t as meta-variables for sorts.[6]

For each sort s, fix a $\mathbb{Z}^+$-indexed set of expression variables V_s. Let ${}^s v_i$ denote the ith expression variable in V_s and let V denote $\bigcup_{s \in \mathcal{S}} \mathsf{V}_s$. I use x, y, and z as meta-variables for expression variables. The choice to annotate variables with sorts is one of convenience. The annotations can be dropped for the systems I consider, and are selectively included in the exposition.

The set of expressions of a pure type system with sorts $\mathcal{S}$ is described by the grammar

$$\mathsf{T} ::= \mathcal{S} \mid \mathsf{V} \mid \Pi \mathsf{V}^{\mathsf{T}}.\ \mathsf{T} \mid \lambda \mathsf{V}^{\mathsf{T}}.\ \mathsf{T} \mid \mathsf{T}\mathsf{T}$$

I use capital Modern English letters like M, N, P, Q, A, B, and C as meta-variables for expressions. Free variables, bound variables, α-congruence, β-reduction, substitution, sub-expressions, *etc.* are defined as usual (see, for example, Barendregt's presentation Barendregt [1993]). Substitution of x with N in M is denoted $M[N/x]$, and I write $N \subset M$ for "N is a sub-expression of M."

A **statement** is a pair of expressions, denoted $M : A$. The first expression is called the **subject** and the second is called the **predicate**. A **proto-context** is a sequence of statements whose subjects are expression variables. The statements appearing in proto-

6. For any subsequent meta-variables, I use positive integer subscripts and tick marks, *e.g.*, s_1, s_2, and s'. Note, however, that in later sections, s_i will refer to a particular sort in tiered systems. I will try to be as clear as possible when distinguishing between these two cases of notation.

contexts are called **declarations**. I use capital Greek letters Γ, Δ, Φ, and Υ as meta-variables for contexts. Often the sequence braces of contexts are dropped and concatenation of contexts is denoted by comma-separation. The β-equality relation and substitution extend to contexts element-wise. For a context Γ and statement $(x : A)$ I write $(x : A) \in \Gamma$ if that statement appears in Γ, and $\Gamma \subset \Delta$ if $(x : A) \in \Gamma$ implies $(x : A) \in \Delta$.

A **proto-judgment** is a proto-context together with statement, denoted $\Gamma \vdash M : N$. The designation "judgment" is reserved for proto-judgments that are derivable according to the rules below. Likewise, the designation "context" is reserved for proto-contexts that appear in some (derivable) judgment.[7]

Definition 1. *The pure type system λS specified by $(\mathcal{S}, \mathcal{A}, \mathcal{R})$ has the following rules for deriving judgments. In what follows, the meta-variables s and s' range over all sorts in $\mathcal{S}$ when unspecified. A variable ${}^s x$ is **fresh** with respect to a context Γ if it does not appear anywhere in Γ.*

- ***Axioms.*** *For any axiom (s, s')*

$$\vdash_{\lambda S} s : s'$$

- ***Variable Introduction.*** *For a variable ${}^s x$ which is fresh with respect to Γ*

$$\frac{\Gamma \vdash_{\lambda S} A : s}{\Gamma, {}^s x : A \vdash_{\lambda S} {}^s x : A}$$

- ***Weakening.*** *For a variable ${}^s x$ which is fresh with respect to Γ*

$$\frac{\Gamma \vdash_{\lambda S} M : A \qquad \Gamma \vdash_{\lambda S} B : s}{\Gamma, {}^s x : B \vdash_{\lambda S} M : A}$$

- ***Product Type Formation/Generalization.*** *For any rule (s, s', s'')*

7. Alternatively, in any non-trivial pure type system λS, a proto-context Γ is a context if $\Gamma \vdash s : s'$ for any axiom (s, s').

$$\frac{\Gamma \vdash_{\lambda S} A : s \qquad \Gamma, {}^s x : A \vdash_{\lambda S} B : s'}{\Gamma \vdash_{\lambda S} \Pi^s x^A.\ B : s''}$$

- ***Abstraction.***

$$\frac{\Gamma, {}^s x : A \vdash_{\lambda S} M : B \qquad \Gamma \vdash_{\lambda S} \Pi^s x^A.\ B : s'}{\Gamma \vdash_{\lambda S} \lambda^s x^A.\ M : \Pi^s x^A.\ B}$$

- ***Application.***

$$\frac{\Gamma \vdash_{\lambda S} M : \Pi^s x^A.\ B \qquad \Gamma \vdash_{\lambda S} N : A}{\Gamma \vdash_{\lambda S} MN : B[N/{}^s x]}$$

- ***Conversion.*** *For any terms A and B such that $A =_\beta B$*

$$\frac{\Gamma \vdash_{\lambda S} M : A \qquad \Gamma \vdash_{\lambda S} B : s}{\Gamma \vdash_{\lambda S} M : B}$$

The subscript on the turnstile is dropped when there is no fear of ambiguity. The annotations on variables in Π-expressions and λ-expressions are non-standard, and will in most cases be dropped, but they are occasionally useful to maintain (*e.g.*, see Lemma 1). It is also standard to write $A \to B$ for $\Pi x^A.\ B$ in the case that x does not appear free in B, and to use the derived inference rule

$$\frac{\Gamma \vdash_{\lambda S} A : s \qquad \Gamma \vdash_{\lambda S} B : s'}{\Gamma \vdash_{\lambda S} A \to B : s''}$$

An expression M is said to be **derivable** in λS if there is some context Γ and expression A such that $\Gamma \vdash_{\lambda S} M : A$. Although there is no distinction between terms and types, it is useful to call a judgment a **type judgment** if it is of the form $\Gamma \vdash A : s$ where $s \in \mathcal{S}$, and a **term judgment** if it is of the form $\Gamma \vdash M : A$ where $\Gamma \vdash A : s$ for some sort s. I also write that M is a term and A is a type in this case. By type correctness (Lemma 3), a judgment that is not a type judgment is a term judgment, though some judgments are both type and term judgments. In the system specified by $(\{s_1, s_2\}, \{(s_1, s_2)\}, \emptyset)$, for example,

- $\vdash s_1 : s_2$ is a type judgements but not a term judgment,

- $x : s_1 \vdash x : s_1$ is a type judgment and a term judgment, and
- $x : s_1, y : x \vdash y : x$ is a term judgment but not a type judgment.

1.1.1 Meta-Theory

I collect here the meta-theoretic lemmas necessary for the subsequent results. I choose not to present any proofs, and instead refer the reader to any of the great resources on pure type systems (Barendregt [1993], Barthe et al. [2001], Kamareddine et al. [2004], among others). For the remainder of the section, fix a pure type system λS.

Lemma 1. *(Generation) For any context Γ and expression A, the following hold.*

- *Sort. For any sort s, if $\Gamma \vdash s : A$, then there is a sort s' such that $A =_\beta s'$ and $(s, s') \in \mathcal{A}$.*
- *Variable. For any sort s and variable ${}^s x$, if $\Gamma \vdash {}^s x : A$, then there is an type B such that $\Gamma \vdash B : s$ and $({}^s x : B)$ appears in Γ and $A =_\beta B$.*
- *Π-expression. For any sort s and expressions B and C, if*

$$\Gamma \vdash \Pi^s x^B.\ C : A$$

then there are sorts s', and s'' such that

$$\Gamma \vdash B : s \qquad \text{and} \qquad \Gamma, {}^s x : B \vdash C : s'$$

and $(s, s', s'') \in \mathcal{R}$ and $A =_\beta s''$.

- *λ-expression. For any sort s and expressions B and M, if*

$$\Gamma \vdash \lambda^s x^B.\ M : A$$

then there is a type C and sort s' such that such that

$$\Gamma \vdash \Pi^s x^B.\ C : s' \qquad \textit{and} \qquad \Gamma, {}^s x : B \vdash M : C$$

and $A =_\beta \Pi^s x^B.\ C$.

- *Application. For expressions M and N, if $\Gamma \vdash MN : A$, then there is a sort s and types B and C such that $\Gamma \vdash M : \Pi^s x^B.\ C$ and $\Gamma \vdash N : B$ and $A =_\beta C[N/{}^s x]$.*

Lemma 2. *(Substitution) For contexts Γ and Δ and expressions M, N, A and B, if*

$$\Gamma, x : A, \Delta \vdash M : B \qquad \textit{and} \qquad \Gamma \vdash N : A$$

then

$$\Gamma, \Delta[N/x] \vdash M[N/x] : B[N/x]$$

Lemma 3. *(Type Correctness) For any context Γ and expressions M and A, if $\Gamma \vdash M : A$ then $A \in \mathcal{S}$ or there is a sort s such that $\Gamma \vdash A : s$.*

Lemma 4. *(Thinning) For contexts Γ and Δ and expressions M and A, if $\Gamma \subset \Delta$ and $\Gamma \vdash M : A$, then $\Delta \vdash M : A$.*

Lemma 5. *(Permutation) For contexts Γ and Δ, variables x and y, and expressions A, B, M, and C, if x does not appear free in B and*

$$\Gamma, x : A, y : B, \Delta \vdash M : C$$

then

$$\Gamma, y : B, x : A, \Delta \vdash M : C$$

Definition 2. *A pure type system is **functional** if*

- *for all axioms* (s, s') *and* (t, t') *if* $s = t$ *then* $s' = t'$*;*
- *for all rules* (s, s', s'') *and* (t, t', t'') *if* $s = t$ *and* $s' = t'$ *then* $s'' = t''$*.*

Lemma 6. *(Type Unicity) If* λS *is functional then for any context* Γ *and expressions* M*,* A*, and* B*, if* $\Gamma \vdash M : A$ *and* $\Gamma \vdash M : B$*, then* $A =_\beta B$*.*

Definition 3. *A sort* s *is a* ***top-sort*** *if there is no sort* s' *such that* $(s, s') \in \mathcal{A}$*.*

Lemma 7. *(Top-Sort Lemma) For any context* Γ*, variable* x*, expressions* A *and* B*, and top-sort* s *the following hold.*

1. $\Gamma \not\vdash s : A$
2. $\Gamma \not\vdash x : s$
3. $\Gamma \not\vdash AB : s$
4. $\Gamma \not\vdash \lambda x^A.\ B : s$.

1.1.2 Tiered Pure Type Systems

General pure type systems are notoriously difficult to work with so it is typical to consider classes of pure type systems satisfying certain properties (*e.g.*, functionality). Here I choose to work with a simple class of systems I call *tiered* pure type systems, which have a very concrete description.

Definition 4. *Let* n *be a positive integer. A pure type system is* ***n-tiered*** *if it has the form*

$$\mathcal{S} = \{s_i \mid i \in [n]\}$$
$$\mathcal{A} = \{(s_i, s_{i+1}) \mid i \in [n-1]\}$$
$$\mathcal{R} \subset \{(s, s', s') \mid (s, s') \in \mathcal{S} \times \mathcal{S}\}$$

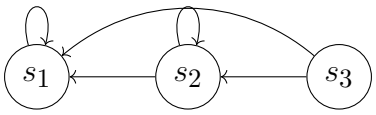


Figure 1.1: A visual representation of the system λU

From this point forward, I freely use the notation (s, s') for the rule (s, s', s'). A couple remarks about these systems:

- these systems can be envisioned as graphs as in Figure 1.1, which is a visual representation of the 3-tiered system λU. In such representations, an arrow (s_i, s_j) indicates the presence of the rule (s_i, s_j). Axioms are not represented in the graph except in the ordered the nodes are presented;

- the 2-tiered systems are the lambda cube together with all systems in the lambda cube minus the rule (s_1, s_1);

- the n-tiered systems are considered in passing by Barthes *et al.* (Remark 2.39, Barthe et al. [2001]). They include natural subsystems of ECC^n (as defined in Luo [1990]) with only the 2-sorted rules;

- we can also naturally consider $\mathbb{Z}^+$-tiered, $\mathbb{Z}^-$-tiered, and $\mathbb{Z}$-tiered systems. These are essentially tiered systems without endpoints.

Working in tiered systems simplifies the arguments in the following section because of their explicit structure, and they are sufficient to consider in so far as their normalization is equivalent to that of a previously considered classes of systems defined in terms of less concrete properties. In particular, we can characterize *separable persistent* pure type systems as disjoint unions of tiered systems (including the infinite tiered systems mentioned in the remark above).

Definition 5. *A pure type system λS is **persistent** if it is functional (Definition 2) and*

- *for all axioms* (s, s') *and* (t, t') *if* $s' = t'$ *then* $s = t$*;*
- $\mathcal{R}_{\lambda S} \subset \{(s, s', s') \mid (s, s') \in \mathcal{S} \times \mathcal{S}\}$.

Let '$\leq_{\mathcal{A}}$' denote the reflexive transitive closure of $\mathcal{A}$, and let '$<_{\mathcal{A}}$' be defined as usual. Furthermore, let '$=_{\mathcal{A}}$' denote the equivalence closure of $\mathcal{A}$. The subscript is dropped when there is no fear of ambiguity.

Definition 6. *A pure type system* λS *is* ***separable*** *if* $(s, s') \in \mathcal{R}_{\lambda S}$ *implies* $s =_{\mathcal{A}_{\lambda S}} s'$.

In order to state the following equivalence, we work in the structural theory of pure type systems of Roux and van Doorn Roux and Doorn [2014].

Definition 7. *For pure type systems* λS *and* $\lambda S'$*, the* ***disjoint union*** $\lambda S \sqcup \lambda S'$ *is specified by*

$$\mathcal{S}_{\lambda S \sqcup \lambda S'} \triangleq \mathcal{S}_{\lambda S} \sqcup \mathcal{S}_{\lambda S'}$$
$$\mathcal{A}_{\lambda S \sqcup \lambda S'} \triangleq \mathcal{A}_{\lambda S} \sqcup \mathcal{A}_{\lambda S'}$$
$$\mathcal{R}_{\lambda S \sqcup \lambda S'} \triangleq \mathcal{R}_{\lambda S} \sqcup \mathcal{R}_{\lambda S'}$$

Lemma 8. *A pure type system is separable and persistent if and only if it is the disjoint union of tiered pure type systems.*

Proof. It is straightforward to verify that tiered systems are persistent and separable, and that the same is true for disjoint unions of such systems, so we focus on the other direction. Let λS be a pure type system that is persistent and separable and consider the partition $\mathcal{P}$ of $\mathcal{S}$ into $=_{\mathcal{A}}$-equivalence classes. Let $\mathcal{S}_p$ be such a $=_{\mathcal{A}}$-equivalence class. Persistence ensures that $\mathcal{S}_p$ is totally ordered by $\leq_{\mathcal{A}}$. Furthermore, it is possible to show that for any two sorts s_i and s_j in $\mathcal{S}_P$, there is a unique finite sequence of sorts $t_1, \ldots, t_k$ such that $s_i = t_1$ and $s_j = t_k$ or $s_i = t_k$ and $s_j = t_1$ and $(t_i, t_{i+1}) \in \mathcal{A}$ for all $i \in [n-1]$. Let λS_p denote the pure

system specified by

$$\mathcal{S}_{\lambda S_p} \triangleq \mathcal{S}_p$$
$$\mathcal{A}_{\lambda S_p} \triangleq \mathcal{A}_{\lambda S} \cap (\mathcal{S}_p \times \mathcal{S}_p)$$
$$\mathcal{R}_{\lambda S_p} \triangleq \mathcal{R}_{\lambda S} \cap (\mathcal{S}_p \times \mathcal{S}_p \times \mathcal{S}_p)$$

Then λS_p is n-, $\mathbb{Z}^+$-, $\mathbb{Z}^-$-, or $\mathbb{Z}$-tiered depending on the existence of a top-sort and a bottom-sort in $\mathcal{S}_p$.

The axioms and rules of the systems for each equivalence class are clearly pairwise disjoint. They also cover all axioms by fiat and rules by separability. Therefore, we can view λS as the system $\bigsqcup_{\mathcal{S}_p \in \mathcal{P}} \lambda S_p$. Formally, they are isomorphic pure type systems.[8] □

This paper is concerned with non-dependent persistent pure type systems, which are separable by definition.

Definition 8. *A tiered pure type system λS is **non-dependent** if its rules are non-dependent, i.e., if $(s, s') \in \mathcal{R}_{\lambda S}$ implies $s \geq_{\mathcal{A}_{\lambda S}} s'$.*

Roux and van Doorn Roux and Doorn [2014] show that the (strong) normalization of a disjoint union of pure type systems is equivalent to the (strong) normalization of each of its individual summands. So on questions of normalization regarding separable persistent systems it suffices to consider tiered systems. Furthermore, we can make the following very simple observation.

Fact 1. *Let $\mathcal{C}$ be a set of tiered pure type systems with the following property: if λS is a $\mathbb{Z}^+$-, $\mathbb{Z}^-$-, or $\mathbb{Z}$-tiered pure type system in $\mathcal{C}$, then for every finite tiered sub-system $\lambda S'$ of λS, there is a finite tiered subsystem $\lambda S''$ such that $\lambda S' \subset \lambda S'' \subset \lambda S$ and $\lambda S'' \in \mathcal{C}$. It then*

8. The definition of a pure type system homomorphism is as one might expect, see Roux and Doorn [2014] for more details.

follows that every system in $\mathcal{C}$ is (strongly) normalizing if and only if every finite system in $\mathcal{C}$ is (strongly) normalizing.

This is simply because derivations are finite, and must be constructable in some finite fragment. So we can restrict our focus to n-tiered systems for well-structured classes of systems.

One of the primary benefits of working in persistent systems in general (and tiered systems in particular) is that derivable expressions can be classified by the *level* in the system at which they are derivable. This property is shown by defining a degree measure on expressions and classifying expressions according to their degree. This result is due to Berardi Berardi [1990], and the presentation here roughly follows the same course.

Definition 9. *The **degree** of an expression is given by the following function* $\deg : \mathsf{T} \to \mathbb{N}$.

$$\begin{aligned}
\deg(s_i) &\triangleq i+1 \\
\deg({}^{s_i}x) &\triangleq i-1 \\
\deg(\Pi x^A.\ B) &\triangleq \deg(B) \\
\deg(\lambda x^A.\ M) &\triangleq \deg(M) \\
\deg(MN) &\triangleq \deg(M)
\end{aligned}$$

I will let T_j denote the set $\{M \in \mathsf{T} \mid \deg(M) = j\}$ and let $\mathsf{T}_{\geq j}$ denote the set $\{M \in \mathsf{T} \mid \deg(M) \geq j\}$.

Lemma 9. *(Classification) Let λS be an n-tiered pure type system. For any expression A, the following hold.*

- $\deg(A) = n+1$ *if and only if* $A = s_n$.
- $\deg(A) = n$ *if and only if* $\Gamma \vdash_{\lambda S} A : s_n$ *for some context* Γ.

- *For* $i \in [n-1]$*, we have* $\deg(A) = i$ *if and only if* $\Gamma \vdash_{\lambda S} A : B$ *and* $\Gamma \vdash_{\lambda S} B : s_{i+1}$ *for some context* Γ *and expression* B*.*

In particular, for context Γ *and expressions* M *and* A*, if* $\Gamma \vdash M : A$ *then* $\deg(A) = \deg(M) + 1$*.*

Finally, some useful facts about degree. See the presentation by Barendregt Barendregt [1993] for proofs in the 2-tiered case.

Lemma 10. *Let* λS *be a tiered pure type system and let* A *and* B *be expressions derivable in* λS*.*

- *If* $\deg(B) = j - 1$ *then*

$$\deg(A[B/^{s_j} x]) = \deg(A)$$

- *If* $A \twoheadrightarrow_\beta B$*, then* $\deg(A) = \deg(B)$*.*

1.2 Outline

With the above context, I can now more easily state the contributions of this dissertation. Beyond weak and strong normalization, this dissertation is concerned with *type-preserving translations*, which are used for type-theoretic conservation theorems. *Forthcoming.*

CHAPTER 2

WEAK AND STRONG NORMALIZATION OF NON-DEPENDENT PERSISTENT PURE TYPE SYSTEMS VIA THUNKIFICATION

This work appears in a standalone paper which is available on Arxiv.

2.1 Introduction

As noted in the previous chapter, one approach to deriving strong normalization from weak normalization is to pass through some form of the λI-calculus, the fragment of the lambda calculus in which lambda-bound variables *must* appear in the bodies of their lambda expressions. In this setting, strong normalization readily follows from weak normalization by a conservation theorem along the lines of the one proved by Church and Rosser Church and Rosser [1936]. The argument is roughly as follows: suppose a system λS is weakly normalizing. Define a translation from expressions to λI-expressions which preserves (1) typability in λS and (2) infinite reduction paths. Translated expressions are weakly normalizing by being typable in λS and, hence, are strongly normalizing by the conservation theorem. And since the translation preserves infinite reduction paths, the untranslated expressions are themselves strongly normalizing.

This technique was originally used by Sørensen Sørensen [1997] and Xi Xi [1996] via a continuation-passing-style (CPS) translation, and Xi Xi [1997] subsequently presented an alternative proof via thunkification that is arguably simpler; rather than passing around continuations, expressions are thunkified and thunks are padded via uninterpreted variables with sub-expressions necessary for translating to λI-expressions. Both are essentially typed versions of Klop's ι-tranlsation, though the thunkification translation is more direct. See the survey in the work of Gortz *et al.* Gørtz et al. [2003] for many more details on the

relationships between these translations and many others.

Barthe *et al.* Barthe et al. [2001] generalized Xi's and Sørensen's result to prove that weak normalization implies strong normalization in *generalized non-dependent clean negatable* pure type systems.[1] In a previous report Mull [2022], I presented the analogous generalization for Xi's thunkification translation to the same class of systems. The primary contribution of this paper is a strengthened version of the BHS result, proved via a non-standard extension of this thunkfication translation. The following points outline the technical contributions, as they pertain to dealing with negatability, cleanliness, and generalized non-dependence, respectively.

- In the informal outline of Sørensen's and Xi's technique above, the first step is doing quite a bit of work; translated expressions are weakly normalizing only because *they are still derivable in* λS which is weakly normalizing by assumption. The endomorphic nature of the translation is what requires negatability. This property ensures the system is expressive enough for thunkified (or CPS-ified) expressions to be typable. But it suffices to show that translated expressions are typable in an extension of λS, and even a *non-normalizing* extension of λS, if it is possible to give a *combinatorial* proof that the translation preserves weak normalization *at the level of terms*. Applying techniques from the study of reduction strategies, I give a combinatorial proof of weak normalization preservation for a variant of the translation which can be applied to non-negatable systems.

- Cleanliness, the second major technical restriction by BHS, ensures the variables mentioned above which are used for padding expressions are, in fact, typable. Roughly speaking, expressions are padded using variables injected in the context of the form $(\mathsf{pad}_A : A \to \bot \to \bot)$, where $\bot$ is the type of the thunk expression $\bullet$ (a distinguised

1. In subsequent exposition, I will refer to this result as the BHS result, or simply BHS, for Barthe, Hatcliff and Sørensen.

variable). Given an expression N of type A, a thunkified expression M of type $\bot \to B$ can be evaluated as $M(\mathsf{pad}_A N \bullet)$, so that the expression N is padded into M. The challenge is that the type A may have variables which appear in the context, and abstraction and generalization (*i.e.*, the formation of λ-terms and Π-types, respectively) both remove variables from the context, potentially making A no longer typable. This is, in some sense, the primary feature of these constructions, the ability to express dependencies at the level of terms instead of at the level of variables in scope. Cleanliness, thus, disallows abstractions and generalizations which remove variables that appear in the types of padding variables. For the translation presented in this paper, I use a different scheme for typing padding variables which allows for more fine-grained control of these dependencies. This technique requires two technical lemmas which may be of independent interest. The first is, in essense, a type-theoretic Skolemization lemma, which allows for contexts to be permuted modulo additional generalizations (Corollary 1). This allows for generalization over the variables that appear in the types of padding variables, eliminating the dependence at the level of variables in scope. The second is a generalization of Corollary 2.46 from BHS (Lemma 12), which characterizes the degree of sub-expressions, and which is also useful in determining when variables can commute with padding variables.

- In the interest of further simplification, I also present a class of basic, concrete pure type systems I call *tiered* pure type systems. Despite their simplicity they are sufficient to consider with regards to questions about normalization. In particular, I characterize non-dependent persistent systems as disjoint unions of possibly infinite tiered systems. And by a simple compactness-style argument, it follows that we can restrict ourselves to the finite tiered systems when considering certain classes of systems.

In what follows I present some preliminary material, which includes some exposition on tiered systems and reduction strategies. I then present the generalized thunkification

translation in two parts: one part for the type-level translation and one part for term-level translation. The term-level translation is further divided into two parts, the negatable and non-negatable cases. For each term-level translation, I include a proof of typability preservation and infinite reduction path preservation, both necessary for the main result (Theorem 6), which is presented in subsection 2.3.3 for completeness.

2.2 Preliminaries

Outside of the preliminary material presented in the section on definitions in the introductory chapter (Section 1.1), I include this short section with an important lemma about non-dependent pure type systems, as well as some information about reductions.

One difficulty with dependencies is that they allow expressions to have sub-expressions of a lower degree than themselves. This makes it very difficult to reverse induct on degree, which is an important proof technique for these systems. In non-dependent systems, the sub-expression of a given expression must have degree at least that of the given expression. Typically, a simplified version of this fact suffices, which says that the degree of variables appearing in an expression must be at least that of the expression itself.

Lemma 11. *Let λS be a non-dependent tiered pure type system. For any expression A, if a variable ${}^{s_j}x$ appears free in A, then $j > \deg(A)$.*

I present a slightly stronger version which is a generalization of Corollary 2.46 from BHS, and for which the previous lemma is a corollary. It captures exactly the possible degrees of sub-expressions, using the following definition for keeping track of potential dependencies. It is worth noting, in future sections, the similarity with cleanliness (Definition 19).

Definition 10. *Let λS be a non-dependent tiered pure type system. Define set $\mathcal{D}_{\lambda S}(i)$ be the smallest set satisfying these properties.*

- $i \in \mathcal{D}_{\lambda S}(i)$

- *if* $(s_j, s_i) \in \mathcal{R}_{\lambda S}$ *then* $\mathcal{D}_{\lambda S}(j) \subset \mathcal{D}_{\lambda S}(i)$
- *if* $(s_j, s_{i+1}) \in \mathcal{R}_{\lambda S}$*, then* $\mathcal{D}_{\lambda S}(j) \subset \mathcal{D}_{\lambda S}(i)$.

These sets are well-defined because they can be constructed by reverse induction on degree.

Lemma 12. *Let* λS *be a non-dependent tiered pure type system. There is an expression* M *derivable in* λS *such that* $\deg(M) = i$ *with a sub-expression* N *such that* $\deg(N) = j$ *if and only if* $j \in \mathcal{D}_{\lambda S}(i)$.

Proof. First, the left direction, by reverse induction on degrees. I prove the stronger fact that M is a derivable type, *i.e.*, there is a context Γ and sort s_i such that $\Gamma \vdash M : s_i$. Note that $\mathcal{D}_{\lambda S}(n) = \{n\}$ so it suffices that $\varnothing \vdash s_{n-1} : s_n$. For arbitrary i, there are three cases to consider. First suppose $j = i$. If $i > 1$, then $\varnothing \vdash s_{i-1} : s_i$ and otherwise $x : s_1 \vdash x : s_1$. Next suppose $j \in \mathcal{D}_{\lambda S}(k)$ where $(s_k, s_i) \in \mathcal{R}_{\lambda S}$. By the inductive hypothesis, there is context Γ and expressions A and B such that $\Gamma \vdash A : s_k$, where $B \subset A$ and $\deg(B) = j$. If $i > 1$, then

$$\Gamma \vdash A \to s_{i-1} : s_i$$

and otherwise

$$\Gamma, x : s_1 \vdash A \to x : s_1.$$

Finally suppose $j \in \mathcal{D}_{\lambda S}(k)$ where $(s_k, s_{i+1}) \in \mathcal{R}_{\lambda S}$. Let Γ, A, and B be as in the previous case. Then $\Gamma \vdash A \to s_i : s_{i+1}$. If $i > 1$, then $\Gamma, y : A, x : A \vdash s_{i-1} : s_i$ so we have

$$\Gamma, y : A \vdash (\lambda x^A.\ s_{i-1})y : s_i$$

This concludes the left direction.

The other direction also follows by reverse induction on i, and then by induction on the structure of derivations. It is straightforward to verify that the sub-expressions of any

derivable expression of degree n is also of degree n. So let M be an derivable expression such that $\deg(M) = i$. It will follow by induction on the structure of derivations that if N is a sub-expression of M then $\deg(N) \in \mathcal{D}_{\lambda S}(i)$. I focus on the cases on which the definition of $\mathcal{D}_{\lambda S}(i)$ depends.

Product Type Formation. Suppose the last inference is of the form

$$\frac{\Gamma \vdash A : s_j \qquad \Gamma, x : A \vdash B : s_i}{\Gamma \vdash \Pi x^A.\ B : s_i}$$

where $(s_j, s_i) \in \mathcal{R}_{\lambda S}$. Let C be a sub-expression of A. If $j > i$, then $C \in \mathcal{D}_{\lambda S}(j)$ by the first inductive hypothesis, which is a subset of $\mathcal{D}_{\lambda S}(i)$ by assumption. If $j = i$, then $C \in \mathcal{D}_{\lambda S}(i)$ by the second inductive hypothesis. The case in which C is a sub-expression of B is similar.

Abstraction. Suppose the last inference is of the form

$$\frac{\Gamma, {}^{s_j} x : A \vdash M : B \qquad \Gamma \vdash \Pi x^A.\ B : s_{i+1}}{\Gamma \vdash \lambda x^A.\ M : \Pi x^A.\ B}$$

where $(s_j, s_{i+1}) \in \mathcal{R}_{\lambda S}$. If $C \subset A$, then $C \in \mathcal{D}_{\lambda S}(j)$ by the first inductive hypothesis. If $C \subset M$, then $C \in \mathcal{D}_{\lambda S}(i)$ by the second inductive hypothesis. □

2.2.1 Reduction and Normalization

Since I am taking a generally more combinatorial approach to proving weak normalization preservation, I include this short section on reductions to collect some useful terminology, notation, and machinery. First, the terminology:

- A **β-redex** of an expression M is a sub-expression of the form $(\lambda x^A.\ P)Q$. In typical abuse of terminology, I will also call an expression of the form $(\lambda x^A.\ P)Q$ a redex, and will call the left expression in a redex the 're' and the right expression the 'dex.' The expression $P[Q/x]$ is the **reduct** of the redex $(\lambda x^A.\ P)Q$.

- An expression M is in **β-normal form** if it does not have any β-redexes, M is **weakly normalizing** if there is a reduction sequence starting at M which ends in a normal

form, and M is **strongly normalizing** if M does not appear in any infinite reduction sequence. I will use the following notation:

$$\mathsf{NF} \triangleq \{M \mid M \text{ is in normal form}\}$$
$$\mathsf{WN} \triangleq \{M \mid M \text{ is weakly normalizing}\}$$
$$\mathsf{SN} \triangleq \{M \mid M \text{ is strongly normalizing}\}.$$

So, for example, we say that a pure type system λS is weakly normalizing if all expressions derivable in λS are also in WN. As an abuse of notation, Finally, I'll write NF also for the function which maps expressions in WN to their normal forms in NF.

Next, the machinery. First is the notion of a single-holed expression, which is typically used to notate the relationship between an expression and one of its sub-expressions. These are called *contexts* in other settings (again, see Barendregt's exposition Barendregt [1984]), but I've used a different name so as not to clash with the above introduced notion of contexts.

Definition 11. *A **single-holed expression**, denoted $C\langle\cdot\rangle$, is defined inductively as follows.*

- $\langle\cdot\rangle$ *is a single-holed expression.*
- *For any variable x, expression M, and single-holed expression $C\langle\cdot\rangle$, the following are all single-holed expressions.*
  - $\Pi x^{C\langle\cdot\rangle}.\ M$
  - $\Pi x^{M}.\ C\langle\cdot\rangle$
  - $\lambda x^{C\langle\cdot\rangle}.\ M$
  - $\lambda x^{M}.\ C\langle\cdot\rangle$
  - $C\langle\cdot\rangle M$
  - $M(C\langle\cdot\rangle)$

Let $C\langle M\rangle$ denote the expression resulting from replacing $\langle\cdot\rangle$ in $C\langle\cdot\rangle$ with M.

Second, a useful result due to Levy Lévy [1978], regarding maximal reductions.

Definition 12. *Let $\mu : \mathsf{T} \to \mathbb{N} \cup \{\infty\}$ be the function which maps an expression M to the length of the longest reduction sequence starting at M (where $\mu(M) = \infty$ if no such sequence exists).*

Lemma 13. *Lévy [1978] For any expressions A, M, $N_1, \ldots, N_k$ and variable*

$$\mu((\lambda x^A.\ M)N_1 \ldots N_k) \leq 1 + \mu(A) + \mu(N_1) + \mu(M[N_1/x]N_2 \ldots N_k)$$

Lemma 13 will be used in subsequent sections to analyze the normalization behavior of translated expressions.

2.3 The Translation

Xi's thunkification translation Xi [1997] was introduced as a simpler approach to deriving strong normalization from weak normalization in typed lambda calculi, as compared to its CPS counterpart Sørensen [1997], Xi [1996]. Roughly speaking, rather than passing continuations, Xi's translation pervasively thunkifies expressions, using uninterpreted padding variables to store sub-expressions in the evaluation of thunkified expressions. This allows for expressions to be mapped to I-expressions, where strong normalization is more readily proved from weak normalization (Lemma 14). The translation is, in a sense, a direct typed implementation of Klop's ι-translation Klop et al. [1993].

The CPS translation was generalized by BHS to *generalized non-dependent, clean, negatable* pure type systems Barthe et al. [2001]. In a research report, I extended Xi's translation by analogy Mull [2022], which yielded a simple alternative proof of the BHS result. In this section, I present a refinement of this translation which eliminates the requirement of negata-

bility and weakens the requirement of cleanliness. Before presenting the translation, I give an overview of the approach.

One of the fundamental features of non-dependent systems is that expressions cannot depend on other expressions of lower degree (Lemma 12). It is, thus, natural to prove properties of derivable expressions by reverse induction on degree. This motivates the following definition.

Definition 13. *A non-dependent n-tiered pure type system is i-**secure** if $deg(M) \geq i$ implies $M \in \mathsf{SN}$ for all derivable expressions M.*

For the remainder of the section, fix an n-tiered pure type system λS and suppose that λS is weakly normalizing. We prove that λS is i-secure for every i, reverse inductively, and in doing so define two families of translations, one for types $\{\rho_i\}_{i=1}^n$ and one for terms $\{\tau_i\}_{i=1}^n$, which is possible exactly because of this fundamental property of non-dependent systems; the degrees of terms are lower than the degree of their types, so there is no need for mutual dependency between the type translations and the term translations.

We then prove the following implications for every expression of degree $i-1$ (that is, terms whose types are degree i).

$$\begin{aligned} M \in \mathsf{WN} &\Rightarrow \tau_i(M) \in \mathsf{WN} \\ &\Rightarrow \tau_i(M) \in \mathsf{SN} \\ &\Rightarrow M \in \mathsf{SN}. \end{aligned}$$

The analysis at each level differs in character according to whether s_i is negatable.

Definition 14. *A sort s_i is **negatable** with if $(s_i, s_i) \in \mathcal{R}_{\lambda S}$. An n-tiered pure type system is i-**negatable** if s_i is negatable in λS.*

In the case that s_i is already negatable, τ_i preserve typability in the same system and the first implication follows by fiat. The second implication follows the fact that τ_i translates

expressions to *I-expressions*, an appropriate generalization of the λI-calculus for this setting, and for which there is an analogous conservation lemma.

Definition 15. *An expression M is an **I-expression at level** j if the following hold.*

- $\deg(M) \geq j$
- *If $\lambda x^A.\ N \subset M$ and $\deg(\lambda x^A.\ N) = j$, then x appears free in N.*

Lemma 14. *(Lemma 5.16, Barthe et al. [2001]) Let λS be a non-dependent tiered pure type system that is i-secure. Then for every derivable I-expression M at level $i-1$, if $M \in \mathsf{WN}$ then $M \in \mathsf{SN}$.*

In the case s_i is not negatable, ρ_i and τ_i are allowed to translate into a negatable extension, but the first implications will need to be proved combinatorially (Lemma 31) and the second implication needs to be shown to still hold (Lemma 25). In particular, the above lemma requires that λS be i-secure, so i-security needs to be preserved in the negatable extension. In both cases, the last implication follows from a standard argument that τ_i preserves infinite reduction paths (Lemma 33).

For the following two translations, **fix a level i and suppose that λS is a weakly normalizing non-dependent i-secure n-tiered pure type system.** The subscript 'i' will be included in definitions, but will typically be dropped in the following exposition.

2.3.1 The Type-Level Translation

We need a distinguished type $\bot_i$ for each sort s_i to stand for the type of thunks. In the case of s_1, this requires an additional type variable in the context.

Definition 16. *If $i > 1$, let $\bot_i$ denote s_{i-1} and, otherwise, let $\bot_1$ be a distinguished variable. Likewise, if $i > 1$ let Δ_i denote the empty context, and otherwise, let Δ_1 denote the context $(\bot_1 : s_1)$.*

The translation ρ gives the types of thunkified terms. This means pervasively replacing each type A where $\deg(A) = i$ with its corresponding thunkified form, *e.g.*, $\bot_i \to \rho_i(A)$.

Definition 17. *Define the functions*

$$\rho_i : \mathsf{T}_{\geq i} \to \mathsf{T}_{\geq i} \qquad \textit{and} \qquad \rho'_i : \mathsf{T}_{\geq i} \to \mathsf{T}_{\geq i}$$

simultaneously as follows.

$$\begin{aligned}
\rho_i(s_j) &\triangleq s_j \qquad \textit{(where } j \geq i-1\textit{)} \\
\rho_i({}^{s_j}x) &\triangleq {}^{s_j}x \qquad \textit{(where } j \geq i+1\textit{)} \\
\rho_i(\Pi^{s_j}x^A.\ B) &\triangleq \Pi^{s_j}x^{\rho'_i(A)}.\ \rho_i(B) \\
\rho_i(\lambda^{s_j}x^A.\ M) &\triangleq \lambda^{s_j}x^{\rho_i(A)}.\ \rho_i(M) \\
\rho_i(MN) &\triangleq \rho_i(M)\rho_i(N) \\
\rho'_i(A) &\triangleq \begin{cases} \bot_i \to \rho_i(A) & \deg(A) = i \\ \rho_i(A) & \textit{otherwise.} \end{cases}
\end{aligned}$$

The function ρ'_i is extended to contexts as follows.

$$\begin{aligned}
\rho'_i(\varnothing) &\triangleq \varnothing \\
\rho'_i(\Gamma, {}^{s_j}x : A) &\triangleq \begin{cases} \rho'_i(\Gamma), {}^{s_j}x : \rho'_i(A) & j \geq i \\ \rho'_i(\Gamma) & \textit{otherwise.} \end{cases}
\end{aligned}$$

The type-level translations are required to commute with substitution and preserve β-equivalence. These two features are necessary for translating application and conversion inferences in the next section. The following two lemmas are standard.

Lemma 15. *(ρ and ρ' commute with substitution) For variable ${}^{s_j}x$ and expressions M and*

N *where* $\deg(N) = j - 1$ *the following hold.*

- $\rho(M[N/^{s_j}x]) = \rho(M)[\rho(N)/^{s_j}x]$
- $\rho'(M[N/^{s_j}x]) = \rho'(M)[\rho(N)/^{s_j}x]$

Proof. We prove both simultaneously by induction on the structure of M. To start, in the case of ρ', if $\deg(M) = i$ then by Lemma 10, $\deg(M[N/x]) = i$ as well. So

$$\begin{aligned}\rho'(M[N/x]) &= \bot \to \rho(M[N/x]) \\ &= \bot \to \rho(M)[\rho(N)/x] \\ &= \rho'(M)[\rho(N)/x]\end{aligned}$$

where the second equality follows from the inductive hypothesis. And if $\deg(M) \neq i$, then

$$\begin{aligned}\rho'(M[N/x]) &= \rho(M[N/x]) \\ &= \rho(M)[\rho(N)/x] \\ &= \rho'(M)[\rho(N)/x]\end{aligned}$$

So we can proceed by induction on M for the case of ρ. The cases in which M is a sort or a variable are straightforward. The following are the cases for Π-expressions and λ-expressions.

$\underline{\Pi\text{-Expression.}}$ If M is of the form $\Pi y^A.\ B$ then

$$\begin{aligned}\rho((\Pi y^A.\ B)[N/x]) &= \rho(\Pi y^{A[N/x]}.\ B[N/x]) \\ &= \Pi y^{\rho'(A[N/x])}.\ \rho(B[N/x]) \\ &= \Pi y^{\rho'(A)[\rho(N)/x]}.\ \rho(B)[\rho(N)/x] \\ &= \rho(\Pi y^A.\ B)[\rho(N)/x]\end{aligned}$$

λ-Expression. If M is of the form $\lambda y^A.\ P$ then

$$\begin{aligned}\rho((\lambda y^A.\ P)[N/x]) &= \rho(\lambda y^{A[N/x]}.\ P[N/x]) \\ &= \lambda y^{\rho(A[N/x])}.\ \rho(P[N/x]) \\ &= \lambda y^{\rho(A)[\rho(N)/x]}.\ \rho(P)[\rho(N)/x] \\ &= \rho(\lambda y^A.\ P)[\rho_i(N)/x]\end{aligned}$$

The case that M is an application is similar. □

Lemma 16. *(ρ and ρ' preserve β-reductions) For derivable expressions M and N, if $M \to_\beta N$ then $\rho(M) \to_\beta \rho(N)$ and $\rho'(M) \to_\beta \rho'(N)$. Furthermore, if $M =_\beta N$ then $\rho(M) =_\beta \rho(N)$ and $\rho'(M) =_\beta \rho'(N)$.*

Proof. The case of β-equality follows immediately from the case of one-step β-reduction. We prove the one-step case for both ρ and ρ' simultaneously by induction on the structure of the one-step β-reduction relation. It is straightforward to show that the lemma holds in the case of ρ' given that it holds inductively for ρ. So we proceed by induction in the case of ρ. In the case of a redex,

$$\begin{aligned}\rho((\lambda x^A.\ M)N) &= (\lambda x^{\rho(A)}.\ \rho(M))\rho(N) \\ &\to_\beta \rho(M)[\rho(N)/x] \\ &= \rho(M[N/x])\end{aligned}$$

It is then straightforward to verify that this property holds up to congruence. □

We also note here the important fact that this translation is simple enough not to affect the normalization behavior of the expression after translation.

Fact 2. *For any expression M, if $M \in \mathsf{NF}$, then $\rho(M) \in \mathsf{NF}$. Furthermore, $M \in \mathsf{WN}$ (resp., SN) if and only if $\rho(M) \in \mathsf{WN}$ (resp., SN) and $\mu(M) = \mu(\rho(M))$.*

Next is typability preservation. This ensures type derivations can be used in proving that the term-level translation preserves typability, *e.g.*, for translating the derivation of Π-type judgments for abstraction. The translation targets the super-system of λS for which s_i is negatable, *i.e.*, where it is possible define types of the form $\bot \to A$ when $\deg(A) = i$.

Definition 18. *For any n-tiered pure type system λS, let $\lambda S^{\neg i}$ be the n-tiered pure type system with the rules $\mathcal{R}_{\lambda S^{\neg i}} \triangleq \mathcal{R}_{\lambda S} \cup \{(s_i, s_i)\}$.*

Lemma 17. *(ρ and ρ' preserve typability) For context Γ and expressions M and A, the following hold.*

1. *If $\Gamma \vdash_{\lambda S} M : A$ and $\deg M \geq i$ then $\Delta, \rho'(\Gamma) \vdash_{\lambda S^{\neg}} \rho(M) : \rho(A)$.*
2. *If $\Gamma \vdash_{\lambda S} A : s_j$ and $\deg A \geq i$ then $\Delta, \rho'(\Gamma) \vdash_{\lambda S^{\neg}} \rho'(A) : s_j$.*

Proof. We prove both simultaneously by induction on the structure of derivations. For item 2, note that by item 1, we have the inference $\Delta, \rho'(\Gamma) \vdash \rho(A) : s_j$. So if $\deg A \neq i$, we're done, and otherwise we have

$$\frac{\Delta, \rho'(\Gamma) \vdash \bot : s_i \qquad \Delta, \rho'(\Gamma) \vdash \rho(A) : s_i}{\Delta, \rho'(\Gamma) \vdash \bot \to \rho(A) : s_i}$$

Note that this judgment is derivable in $\lambda S^{\neg}$ by construction.

For item 1, we proceed with each case. The case in which the derivation is a single axiom is straightforward.

Variable Introduction. Suppose the last inference is of the form

$$\frac{\Gamma \vdash A : s_j}{\Gamma, x : A \vdash x : A}$$

where $j \geq i + 1$. Note that $\rho'(A) = \rho(A)$ since $\deg(A) > i$. So by the inductive hypothesis, we have

$$\frac{\Delta, \rho'(\Gamma) \vdash \rho(A) : s_j}{\Delta, \rho'(\Gamma), x : \rho(A) \vdash x : \rho(A)}$$

Weakening. Suppose the last inference is of the form

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s_j}{\Gamma, x : B \vdash M : A}$$

By the inductive hypothesis, $\Delta, \rho'(\Gamma) \vdash \rho(M) : \rho(A)$, so if $\deg(B) \leq i$, then we're done. Otherwise, we have

$$\frac{\Delta, \rho'(\Gamma) \vdash \rho(M) : \rho(A) \qquad \Delta, \rho'(\Gamma) \vdash \rho'(B) : s_j}{\Delta, \rho'(\Gamma), x : \rho'(B) \vdash \rho(M) : \rho(A)}$$

where the right antecedent judgment also follows from the inductive hypothesis.

Product Type Formation. If the last inference is of the form

$$\frac{\Gamma \vdash A : s_j \qquad \Gamma, x : A \vdash B : s_k}{\Gamma \vdash \Pi x^A.\ B : s_k}$$

then by the inductive hypothesis, we have

$$\frac{\Delta, \rho'(\Gamma) \vdash \rho'(A) : s_j \qquad \Delta, \rho'(\Gamma), x : \rho'(A) \vdash \rho(B) : s_k}{\Delta, \rho'(\Gamma) \vdash \Pi x^{\rho'(A)}.\ \rho(B) : s_k}$$

where $k \geq i$. Note we can apply the inductive hypothesis to the left antecedent judgment since $\deg(A) \geq \deg(B)$ by non-dependence and so

$$\deg(A) \geq \deg(\Pi x^A.\ B) \geq i.$$

Abstraction. Suppose the last inference is of the form

$$\frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash \Pi x^A.\ B : s_j}{\Gamma \vdash \lambda x^A.\ M : \Pi x^A.\ B}$$

Since $\deg(M) = \deg(\lambda x^A.\ M) \geq i$, we have $\deg(B) > i$, and by non-dependence, we have $\deg(A) \geq \deg(B) > i$. In particular, $\rho'(A) = \rho(A)$. Therefore, we have

$$\frac{\Delta, \rho'(\Gamma), x : \rho(A) \vdash \rho(M) : \rho(B) \qquad \Delta, \rho'(\Gamma) \vdash \Pi x^{\rho(A)}.\ \rho(B) : s_j}{\Delta, \rho'(\Gamma) \vdash \lambda x^{\rho(A)}.\ \rho(M) : \Pi x^{\rho(A)}.\ \rho(B)}$$

Application. Suppose the last inference is of the form

$$\frac{\Gamma \vdash M : \Pi x^A.\ B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B[N/x]}$$

By non-dependence, $\deg A \geq \deg B > \deg M \geq i$ and $\rho'(A) = \rho(A)$. So by the inductive hypothesis we have

$$\frac{\Delta, \rho'(\Gamma) \vdash \rho(M) : \Pi x^{\rho(A)}.\ \rho(B) \qquad \Delta, \rho'(\Gamma) \vdash \rho(N) : \rho(A)}{\Delta, \rho'(\Gamma) \vdash \rho(M)\rho(N) : \rho(B)[\rho(N)/x]}$$

where $\rho(B)[\rho(N)/x] = \rho(B[N/x])$ by Lemma 15.

Conversion. Suppose the last inference is of the form

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s_j}{\Gamma \vdash M : B}$$

where $j \geq i$ and $A =_\beta B$. By the inductive hypothesis, we have

$$\frac{\Delta, \rho'(\Gamma) \vdash \rho(M) : \rho(A) \qquad \Delta, \rho'(\Gamma) \vdash \rho(B) : s_j}{\Delta, \rho'(\Gamma) \vdash \rho(M) : \rho(B)}$$

where $\rho(A) =_\beta \rho(B)$ by Lemma 16. □

2.3.2 The Term-Level Translation

Next we define the translation of terms. At a high level, we need to add uninterpreted padding variables to our context for collecting expressions in the bodies of λ-expressions so that the translation maps expressions to I-expressions. There are two issues that need to be addressed.

1. The system λS may not be expressive enough to carry out the translation.
2. The types of the padding variables may depend on variables in the context, which restricts our ability to generalize or abstract over those variables.

BHS handles these two problems by requiring λS to satisfy two technical restrictions. The first is *negatability*, which was covered in the previous section, and is necessary to derive the types of translated terms. The second is *cleanliness*, which disallows generalizing or abstracting over variables on which the types of padding variables depend.

Definition 19.

- *A rule* (s_i, s_j) *is* ***generalizable*** *if* $(s_{i+1}, s_j) \in \mathcal{R}_{\lambda S}$.
- *A rule* (s_i, s_j) *is* ***harmless*** *if neither* (s_k, s_j) *nor* (s_k, s_{j-1}) *are in* $\mathcal{R}_{\lambda S}$ *when* $k > j$.

An n-tiered pure type system λS *is* ***clean*** *if all its rules are generalizable or harmless.*

BHS uses the notion of cleanliness to characterize systems in which padding variables are definable. These padding variables are polymorphic when possible, in which case they are typed as $\Pi A^{s_j}.\ A \to \bot \to \bot$. But if the rules for defining this type are not available (*i.e.*, the rule (s_j, s_i) is not generalizable) the padding variables are typed as $\rho(A)' \to \bot \to \bot$, with the understanding that $\rho'(A)$ may depend on variables in the context. For an expression M derivable in the context Γ, its translation $\tau(M)$ is derived in a context of the form

$$\Delta, \rho'(\Gamma), \Upsilon$$

where Υ contains padding variables depending on variables in $\Delta, \rho'(\Gamma)$. The challenge is then translating abstractions and generalizations (though the latter is significantly more problematic). To illustrate this, consider that the left antecedent judgment of abstraction is translated roughly to

$$\Delta, \rho'(\Gamma), x : \rho'(A), \Upsilon \vdash \tau(M) : \rho(A)$$

so in order to abstract over the variable x, it must commute with the entire context Υ of padding variables, *i.e.*, it must be that x does not appear free any of the types in Υ. By Lemma 11, the only types which can depend on the variable x are those with higher degree than that of $\rho'(A)$, so cleanliness requires that either

- *(generalizability)* all types of degree larger than $\deg(\rho'(A))$ have polymorphic padding variables, or

- *(harmlessness)* there are no rules for abstracting or generalizing over variables whose types have larger degree than $\deg(\rho'(A))$.

In the next two sections, I present two variants of the term-level translation τ, depending on whether s_i is negatable. In the non-negatable case, τ targets an extension of λS, but is otherwise an application of the ideas in the BHS paper. The deviation is in the new requirement that weak normalization preservation, i-security, and infinite reduction path preservation must be proved explicitly, as we can no longer assume that the target system of τ is weakly normalizing. In the negatable case, τ is a novel variant of the thunkification translation which uses a more complex padding scheme that reduces the structural restrictions on λS, the details of which will be covered in Section 2.3.2.

The Non-Negatable Case

BHS handles the case of irrelevant non-negatable sorts by an argument leveraging the sparse inhabitation of types at irrelevant sorts.

Definition 20. *A sort s_j is **irrelevant** with respect to rules $\mathcal{R}$ if there is no sort s_k such that $(s_k, s_j) \in \mathcal{R}$. A rule (s_k, s_j) is said to be j-**relevant**.*

Lemma 18. *(Lemma 5.19, Barthe et al. [2001]) Let λS be a non-dependent, weakly normalizing, i-secure tiered pure type system. If s_i is irrelevant, then every derivable expression M with $\deg(M) = i - 1$ is strongly normalizing. That is, λS is $(i-1)$-secure.*

The translation below handles non-negatable sorts by translating into a negatable extension of λS. This was also done in the previous section to ensure that translated types were derivable.

Because of Lemma 12, we can work with a slightly weaker notion of cleanliness for this translation.

Definition 21. *A rule (s_j, s_i) is **weakly harmless** if for $k > j$, it follows that $(s_k, s_i) \in \mathcal{R}_{\lambda S}$ or $(s_k, s_{i-1}) \in \mathcal{R}_{\lambda S}$ implies $k \notin \mathcal{D}_{\lambda S}(j)$. A non-dependent n-tiered pure type system λS is i-**weakly clean** if all i-relevant rules are either generalizable or weakly harmless.*

As noted in the introduction to this section, the translation needs access to padding variables so that they can be used to hide terms inside thunks, and these padding functions will be polymorphic when possible; if (s_j, s_i) is generalizable, then it is possible to add a variable pad of type $\Pi A^{s_j}.\ A \to \bot \to \bot$ so that if we need to pad a term N of type $\rho'(A)$ of degree j into a thunkified term M of type $\bot \to \tau(B)$, we can type the term $\lambda \bullet^{\bot} M(\mathsf{pad} A N \bullet)$ as $\bot \to \tau(B)$ as well. If (s_j, s_i) is not generalizable, then the padding variables are typed simply as $\rho'(A) \to \bot \to \bot$ with the caveat that $\rho'(A)$ may have free variables that appear in the context.

Definition 22. *For each index i, define $\Upsilon_{i,M}$ inductively on the structure of M as follows.*

$$\begin{aligned}
\Upsilon_{i,s_{i-2}} &\triangleq \varnothing \\
\Upsilon_{i,^{s_i}x} &\triangleq \varnothing \\
\Upsilon_{i,\Pi x^A.\ B} &\triangleq \Upsilon_{i,A}, \Upsilon_{i,B} \\
\Upsilon_{i,\lambda x^A.\ M} &\triangleq \mathsf{pad}_x : \alpha_{i,A}, \Upsilon_{i,M} \\
\Upsilon_{i,MN} &\triangleq \Upsilon_{i,M}
\end{aligned}$$

where

$$\alpha_{i,A} = \begin{cases} \Pi t^{s_{\deg(A)}}.\ t \to \bot_i \to \bot_i & (\deg(A), i) \text{ is generalizable} \\ \rho'_i(A) \to \bot_i \to \bot_i & (\deg(A), i) \text{ is weakly harmless} \end{cases}$$

I will write $\langle M, N \rangle_{\rho(A)}$ for both $\mathsf{pad}_x \rho(A) M N$ when $(\deg(A), i)$ is generalizable and $\mathsf{pad}_x M N$ when $(\deg(A), i)$ is weakly harmless, if there is no fear of ambiguity.

Fact 3. *For any derivable expression M, the context Δ, Υ_M is well-formed in $\lambda S^{\neg}$.*

The context of padding variables appears *after* the translation of a given context, as it may depend on some the declarations in it, so to translate abstractions and generalizations, we need some declarations to commute with the context of padding variables. The following ensures this is possible.

Lemma 19. *If* (s_j, s_i) *or* (s_j, s_{i-1}) *are in* $\mathcal{R}_{\lambda S}$ *then no variables (other than* $\bot$*) of degree* $j-1$ *(i.e., of the form* ${}^{s_j}x$*) can appear free in* Υ_M *for any derivable expression* M *where* $\deg(M) = i-1$.

Proof. By induction on the structure of M. The only interesting case is if M is of the form $\lambda x^A.\, N$ where $(\deg(A), i)$ is weakly harmless. If, for some k, the variable ${}^{s_k}x$ appears free in $\rho'(A)$, then it must also appear free in A (this is straightforward to check), and so by Lemma 12, it must be that $k > \deg(A)$ and $k \in \mathcal{D}_{\lambda S}(\deg(A))$. But by weak harmlessness, $j \notin \mathcal{D}_{\lambda S}(\deg(A))$, so must be that $j \neq k$. □

As for the translation itself, it is quite natural. The core of the definition is in the case of λ-expressions, where the padding variables are used to ensure our translation maps expressions to I-expressions.

Definition 23. *Define the functions*

$$\tau_i : \mathsf{T}_{i-1} \to \mathsf{T}_{i-1} \qquad \textit{and} \qquad \tau'_i : \mathsf{T}_{i-1} \to \mathsf{T}_{i-1}$$

simultaneously as follows.

$$\begin{aligned}
\tau_i(s_{i-2}) &\triangleq \bullet_i \\
\tau_i({}^{s_i}x) &\triangleq {}^{s_i}x\bullet_i \\
\tau_i(\Pi x^A.\ B) &\triangleq \begin{cases} \tau_i(A) \to \tau_i(B) & \deg(A) = i-1 \\ \Pi x^{\rho_i'(A)}.\ \tau_i(B) & \textit{otherwise} \end{cases} \\
\tau_i(\lambda x^A.\ M) &\triangleq \lambda x^{\rho_i'(A)}.\ \tau_i'(M)\langle x, \bullet_i\rangle_{\rho_i'(A)} \\
\tau_i(MN) &\triangleq \tau_i(M)\rho_i(N) \\
\tau_i'(M) &\triangleq \lambda\, \bullet_i^{\perp_i}\ .\ \tau_i(M)
\end{aligned}$$

where $j \geq i-1$ and $\bullet_i$ is a distinguished variable.

This translation must first be shown to be well-defined. In particular, it must be that if MN is derivable, then $\deg(N) \geq i$, so that ρ may be applied. This is the first place where we use non-negatability. By generation, M must be be given a Π-type $\Pi x^A.\ B$ where $\deg(B) = i$ by assumption and $\deg(A) \geq i$ by non-dependence. By non-negatability, it must in fact be that $\deg(A) \geq i+1$, and so $\deg(N) \geq i$.

Note that, for sorts, $\perp_i = s_{i-1}$, so $\tau(s_{i-1})$ will be given the correct type. This trick is used to ensure that the translation maps to I-expressions even for abstractions over thunks.

Fact 4. *For any derivable expression M such that $\deg(M) = i-1$, the variable $\bullet$ appears free in $\tau(M)$.*

Now we verify that this translation preserves typability. This is fairly mechanical. Again, recall that we are checking typability *in a stronger system.*

Lemma 20. *(τ and τ' preserve typability) For any context Γ and expressions M and A, if $\Gamma \vdash_{\lambda S} M : A$ and $\Gamma \vdash_{\lambda S} A : s_i$, then*

1. $\Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_M \vdash_{\lambda S^\neg} \tau(M) : \rho(A)$

2. $\Delta, \rho'(\Gamma), \Upsilon_M \vdash_{\lambda S^\neg} \tau'(M) : \rho'(A)$

Proof. We prove both simultaneously by induction on the structure of derivations. For item 2, suppose that

$$\Delta, \bullet : \bot, \Upsilon_M, \rho'(\Gamma), \vdash \tau(M) : \rho(A)$$

By typability preservation of ρ (Lemma 17) and thinning (Lemma 4) we can derive

$$\Delta, \rho'(\Gamma), \Upsilon_M \vdash \bot \to \rho(A) : s_i$$

and by permutation (Lemma 5) we can derive

$$\Delta, \rho'(\Gamma), \Upsilon_M, \bullet : \bot \vdash \tau(M) : \rho(A)$$

which means by abstraction we can derive

$$\Delta, \Upsilon, \rho'(\Gamma) \vdash \lambda\, \bullet^{\bot}\, .\, \tau(M) : \bot \to \rho(A)$$

For item 1, we consider each case.

Axiom. If the derivation is of the form

$$\vdash s_{i-2} : s_{i-1}$$

then of course

$$\Delta, \bullet_i : \bot_i \vdash \bullet_i : \bot_i$$

Recall that $\bot_i = s_{i-1} = \rho(s_{i-1})$.

Variable Introduction. Suppose the last derivation is of the form

$$\frac{\Gamma \vdash A : s_i}{\Gamma, x : A \vdash x : A}$$

By ρ typability preservation and thinning, we can derive

$$\frac{\Delta, \bullet : \bot, \rho'(\Gamma) \vdash \rho'(A) : s_i}{\Delta, \bullet : \bot, \rho'(\Gamma), x : \rho'(A) \vdash x : \rho'(A)}$$

and then we can use weakening and application to derive

$$\Delta, \bullet : \bot, \rho'(\Gamma), x : \rho'(A) \vdash x\bullet : \rho(A)$$

Weakening. Suppose the last inference is of the form

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s_j}{\Gamma, x : B \vdash M : A}$$

If $j < i$, then we have

$$\Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_M \vdash \tau(M) : \rho(A)$$

directly by the inductive hypothesis. Otherwise, B is in the domain of ρ' and by the inductive hypothesis, ρ typability preservation, and thinning, we have

$$\frac{\begin{array}{c}\Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_M \vdash \tau(M) : \rho(A) \\ \Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_M \vdash \rho'(B) : s_j\end{array}}{\Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_M, x : \rho'(B) \vdash \tau(M) : \rho(A)}$$

Since none of the padding variables in Υ_M appear in $\rho'(B)$ it follows by permutation that we can also derive

$$\Delta, \bullet : \bot, \rho'(\Gamma), x : \rho'(B), \Upsilon_M \vdash \tau(M) : \rho(A)$$

Product Type Formation. Suppose the last inference is of the form

$$\frac{\Gamma \vdash A : s_j \qquad \Gamma, {}^{s_j}x : A \vdash B : s_{i-1}}{\Gamma \vdash \Pi x^A.\ B : s_{i-1}}$$

If $j = i - 1$, then by the inductive hypothesis, and thinning we have

$$\frac{\begin{array}{c}\Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_A, \Upsilon_B \vdash \tau(A) : s_{i-1} \\ \Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_A, \Upsilon_B \vdash \tau(B) : s_{i-1}\end{array}}{\Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_A, \Upsilon_B \vdash \tau(A) \to \tau(B) : s_{i-1}}$$

Otherwise, A is in the domain of ρ' and the inductive hypothesis gives us

$$\Delta, \bullet : \bot, \rho'(\Gamma), x : \rho'(A), \Upsilon_B \vdash \tau(B) : s_{i-1}$$

By Lemma 19, x does not appear free in Υ_B, so by permutation, ρ typability preservation and thinning, we can derive

$$\frac{\Delta, \bullet : \bot\rho'(\Gamma), \Upsilon_A, \Upsilon_B \vdash \rho'(A) : s_j \qquad \Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_A, \Upsilon_B, {}^{s_j}x : \rho'(A) \vdash \tau(B) : s_{i-1}}{\Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_A, \Upsilon_B \vdash \Pi x^{\rho'(A)}.\ \tau(B) : s_{i-1}}$$

Abstraction. Suppose the last inference is of the form

$$\frac{\Gamma, {}^{s_j}x : A \vdash M : B \qquad \Gamma \vdash \Pi x^A.\ B : s_i}{\Gamma \vdash \lambda x^A.\ M : \Pi x^A.\ B}$$

By the inductive hypothesis,

$$\Delta, \rho'(\Gamma), x : \rho'(A), \Upsilon_M \vdash \tau'(M) : \bot \to \rho(B)$$

and by Lemma 19 and permutation, in fact we have

$$\Delta, \rho'(\Gamma), \Upsilon_M, x : \rho'(A) \vdash \tau'(M) : \bot \to \rho(B)$$

By assumption, (s_j, s_i) is generalizable or weakly harmless. In the first case, we have

$$\Delta \vdash \Pi A^{s_j}.\ A \to \bot \to \bot : s_i$$

Otherwise, by ρ typability preservation and a couple additional steps (including a use of generation), we can derive

$$\Delta, \rho'(\Gamma) \vdash \rho'(A) \to \bot \to \bot : s_i$$

so in both cases, we can apply thinning to derive.

$$\Delta, \rho'(\Gamma), \Upsilon_{\lambda x^A.\ M}, x : \rho'(A) \vdash \tau'(M) : \bot \to \rho(B)$$

Therefore, in a few steps we can derive

$$\Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_{\lambda x^A.\ M}, x : \rho'(A) \vdash \langle x, \bullet \rangle_{\rho'(A)} : \bot$$

which can be used with application to derive

$$\Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_{\lambda x^A.\ M}, x : \rho'(A) \vdash \tau'(M)\langle x, \bullet \rangle_{\rho'(A)} : \rho(B)$$

Finally, this can be used with abstraction to derive

$$\Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_{\lambda x^A.\ M} \vdash \lambda x^{\rho'(A)}.\ \tau'(M)\langle x, \bullet \rangle_{\rho'(A)} : \Pi x^{\rho'(A)}.\ \rho(B)$$

Application. Suppose the last inference is of the form

$$\frac{\Gamma \vdash M : \Pi x^A.\ B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B[N/x]}$$

As noted above, by non-negatability, it must be that $\deg(N) \geq i$, which implies N is in the domain of ρ, so with additional thinning we have

$$\frac{\begin{array}{c}\Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_M, \vdash \tau(M) : \Pi x^{\rho(A)}.\ \rho(B) \\ \Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_M \vdash \rho(N) : \rho(A)\end{array}}{\Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_M \vdash \tau(M)\rho(N) : \rho(B)[\rho(N)/x]}$$

and $\rho(B)[\rho(N)/x] = \rho(B[N/x])$ by Lemma 15.

Conversion. Suppose the last inference is of the form.

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s_j}{\Gamma \vdash M : B}$$

where $A =_\beta B$. By the inductive hypothesis, ρ-typability preservation, thinning, and ρ β-preservation (Lemma 16), we have

$$\frac{\Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_M \vdash \tau(M) : \rho(A) \quad \Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_M \vdash \rho(B) : s_j}{\Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon_M \vdash \tau(M) : \rho(B)}$$

□

The focus of the remainder of the section is to prove the key lemmas below which state that

- (Lemma 31) τ preserves weak normalization *of terms* (but not necessarily of the system as a whole).
- (Lemma 25 i-security is preserved from λS to $\lambda S^\neg$, and hence, the conservation lemma (Lemma 14) holds for to $\lambda S^\neg$.
- (Lemma 33) τ preserves infinite reduction sequences.

These encompass the three steps of the outline given at the beginning of the section. I start with the second result since it fairly general and useful in proving the remaining to lemmas.

i-Security Preservation. The basic problem is the following: λS and $\lambda S^\neg$ derive the same expressions of degree greater than i, but there is a small class of degree i expressions that are derivable with the introduction of the new rule (s_i, s_i), namely function types of the form $A \to B$ where $\deg(A) = \deg(B) = i$ (there can be no dependence because all variables of B must have types of degree greater than i). Without loss of generality we may annotate these types as $A \to^i B$.

It is necessary to prove that the introduction of these types does not allow for non-strongly-normalizing expressions. Morally speaking, in degree i expressions, these function types are equivalent to pair types; any substitution that happens in Π-types only occurs to the right of the turnstile, when it is treated as a type. So the real challenge is to prove

that pair types for degree i types do not allow for non-strongly-normalizing expressions. I emphasize here that this only refers to *types*. We don't need to worry about how these types are inhabited, since such an expression would be of degree $i - 1$.

Let M be an expression of degree i derivable in $\lambda S^{\neg}$ and suppose that M has an infinite reduction sequence. The rough idea is to think of $A \to^i B$ appearing in M as a non-deterministic choice between A and B in the derivation of M. Considering all non-deterministic choices, we construct a set of *projections* of M, which are derivable without an uses of $\to^i$, and hence derivable in λS. It will then be possible to track the development of reductions in these projections alongside the infinite reduction sequence of M and show that there is always some projection which can make progress for every reduction in the sequence. Since the set of candidates for M is finite, some candidate expression must make infinite progress.

Definition 24. *The set of* ***projections*** *of a derivable expression M of $\lambda S^{\neg}$ is defined inductively as follows.*

$$
\begin{aligned}
\mathsf{Proj}(s_j) &\triangleq \{s_j\} \\
\mathsf{Proj}(x) &\triangleq \{x\} \\
\mathsf{Proj}(\Pi x^A.\ B) &\triangleq \{\Pi x^{A'}.\ B' \mid A' \in \mathsf{Proj}(A) \text{ and } B' \in \mathsf{Proj}(B)\} \\
\mathsf{Proj}(\lambda x^A.\ M) &\triangleq \{\lambda x^{A'}.\ M' \mid A' \in \mathsf{Proj}(A) \text{ and } M' \in \mathsf{Proj}(M)\} \\
\mathsf{Proj}(MN) &\triangleq \{M'N' \mid M' \in \mathsf{Proj}(M) \text{ and } N' \in \mathsf{Proj}(N)\} \\
\mathsf{Proj}(A \to^i B) &\triangleq \mathsf{Proj}(A) \cup \mathsf{Proj}(B)
\end{aligned}
$$

Lemma 21. *If M is derivable in $\lambda S^{\neg}$, then every projection of M is derivable in λS. Furthermore, if $\deg(M) = i$, then every projection is of degree i as well.*

Lemma 22. *For derivable expression M and N such that* $\deg(M) = i$,

$$\mathsf{Proj}(M[N/x]) = \{M'[N'/x] \mid M' \in \mathsf{Proj}(M) \text{ and } N' \in \mathsf{Proj}(N)\}$$

The proofs of the above lemmas are straighforward. Unfortnately, projections do not behave very well with respect to β-reduction. There are two issues to address.

- There may be redexes in an expression that are not represented in its projections. In the simplest case, a redex appearing in the left hand side of the expression $A \to^i B$ does not have an analogous redex in any of the projections of B appearing in $\mathsf{Proj}(A \to^i B)$.
- Even when every projection of M is reduced along the analogous redex when it exists, the resulting set of projections may not be a full set of projections of the reduct of M. Reducing any redex which creates multiple copies of its argument will yield an expression M' whose projections contains expressions that are not the result of reducing projections of M.

The first problem is addressed by inductively defining a β-reduction operation for sets of projections, which may act as the identity on some projections. The second problem is addressed by defining the notion of *coverage*; even if a set of projections does not contain all of $\mathsf{Proj}(M)$, there may be sufficiently many of them that they capture enough information about M. Roughly speaking, a set of projections covers M all non-deterministic branches in the expression tree are considered, even if not all combinations are considered.

Definition 25. *Let M and N be a derivable expressions in $\lambda S^\neg$ such that $M \to_\beta N$ along the redex R, and let M' be a projection of M. Define $\beta_R^M(M')$ by induction on the structure*

of M as follows. Note that M cannot be a sort or a variable.

$$\beta_R^{\Pi x^A.\ B}(\Pi x^{A'}.\ B') \triangleq \begin{cases} \Pi x^{\beta_R^A(A')}.\ B' & R \subset A \\ \Pi x^{A'}.\ \beta_R^B(B') & R \subset B \end{cases}$$

$$\beta_R^{\lambda x^A.\ M}(\lambda x^{A'}.\ M') \triangleq \begin{cases} \lambda x^{\beta_R^A(A')}.\ M' & R \subset A \\ \lambda x^{A'}.\ \beta_R^M(M') & R \subset M \end{cases}$$

$$\beta_R^{MN}(M'N') \triangleq \begin{cases} \beta_R^M(M')N' & R \subset M \\ M'\beta_R^N(N') & R \subset N \\ P[Q/x] & R = MN = (\lambda x^A.\ P)Q \end{cases}$$

$$\beta_R^{A \to^i B}(M) \triangleq \begin{cases} \beta_R^A(M) & M \in \mathsf{Proj}(A), R \subset A \\ \beta_R^B(M) & M \in \mathsf{Proj}(B), R \subset B \\ M & \textit{otherwise} \end{cases}$$

For a subset $\mathcal{P}$ of $\mathsf{Proj}(M)$, define $\beta_R^M(\mathcal{P}) \triangleq \{\beta_R^M(P) \mid P \in \mathcal{P}\}$.

Lemma 23. *For derivable expressions M and N such that $M \to_\beta N$ along R, if $\mathcal{P} \subset \mathsf{Proj}(M)$, then $\beta_R^M(\mathcal{P}) \subset \mathsf{Proj}(N)$.*

This lemma is proved by induction on the structural definition of a redex and then using Lemma 22 in the case of redex. Next, the notion of coverage.

Definition 26. *For an expression of the form $\Pi x^A.\ B$ and a subset $\mathcal{P}$ of $\mathsf{Proj}(\Pi x^A.\ B)$ let $\mathcal{P}_A$ denote the set $\{A' \mid \Pi x^{A'}.\ B' \in \mathcal{P}\}$ and let $\mathcal{P}_B$ denote the set $\{B' \mid \Pi x^{A'}.\ B' \in \mathcal{P}\}$. I will use the analogous notation for λ-expressions and applications. In the case of $\to^i$, let $\mathcal{P}_A$ denote the set $\mathcal{P} \cap \mathsf{Proj}(A)$ and let $\mathcal{P}_B$ denote the set $\mathcal{P} \cap \mathsf{Proj}(B)$. For any expression M, a set of projections $\mathcal{P}$ from $\mathsf{Proj}(M)$ **covers** M if the following hold.*

- *$\mathcal{P}$ covers s_j if $\mathcal{P} = \{s_j\}$.*

- $\mathcal{P}$ *covers* ${}^{s}{}_{j}x$ *if* $\mathcal{P} = \{{}^{s}{}_{j}x\}$.
- $\mathcal{P}$ *covers* $\Pi x^{A}.\ B$ *if* $\mathcal{P}_A$ *covers* A *and* $\mathcal{P}_B$ *covers* B.
- $\mathcal{P}$ *covers* $\lambda x^{A}.\ M$ *if* $\mathcal{P}_A$ *covers* A *and* $\mathcal{P}_M$ *covers* M.
- $\mathcal{P}$ *covers* MN *if* $\mathcal{P}_M$ *covers* M *and* $\mathcal{P}_N$ *covers* N.
- $\mathcal{P}$ *covers* $A \to^{i} B$ *if* $\mathcal{P}_A$ *covers* A *and* $\mathcal{P}_B$ *covers* B.

The fundamental lemma of coverage is that it is preserved by β-reduction.

Lemma 24. *Let M and N be expressions such that $M \to_\beta N$ along the redex R and let $\mathcal{P}$ be a set of projections of M. If $\mathcal{P}$ covers M then $\beta^M_R(\mathcal{P})$ covers N. Furthermore, there is some projection P of $\mathcal{P}$ such that $P \to_\beta \beta^M_R(P)$ (in particular, $\beta^M_R(P) \neq P$).*

Proof. (Sketch) This follows by induction on notion of coverage, using Lemma 22 in the case of a redex. In this process, it is possible to build inductively a witness to the fact that some element in $\mathcal{P}$ is β-reduced. □

Lemma 25. *If λS is i-secure, then $\lambda S^{\neg}$ is i-secure.*

Proof. Suppose that $\lambda S^{\neg}$ is not i-secure. Let M be an expression such that $\deg(M) \geq i$ and $M \notin \mathsf{SN}$. Since λS and $\lambda S^{\neg}$ derive the same expressions of degree greater than i, it must be that $\deg(M) = i$. Given an infinite reduction sequence starting at M, Lemma 24 implies there is an infinite sequences of β reductions on sets of projections starting at $\mathsf{Proj}(M)$ (which trivially covers M). Since $\mathsf{Proj}(M)$ is finite, there some expression in $\mathsf{Proj}(M)$ which has an infinite reduction sequence. Since all expression in $\mathsf{Proj}(M)$ are derivable in λS and are degree i, it follows that λS is not i-secure. □

Weak Normalization Preservation. First, a substitution commutation lemma akin to Lemma 15 but for τ.

Lemma 26. *(τ commutes with substitution) For variable ${}^{s_k}x$ and derivable expressions M and N where* $\deg(M) = i - 1$ *and* $\deg(N) \geq i$*,*

$$\tau(M[N/{}^{s_k}x]) = \tau(M)[\rho(N)/{}^{s_k}x]$$

Proof. By induction on the structure of M. The cases in which M is a sort or a variable are straightforward.

Π-Expression. Suppose M is of the form $\Pi y^A.\ B$. If $\deg(A) = i - 1$, then

$$\begin{aligned}\tau((\Pi y^A.\ B)[N/x]) &= \tau(A[N/x]) \to \tau(B[N/x]) \\ &= \tau(A)[\rho(N)/x] \to \tau(B)[\rho(N)/x] \\ &= \tau(\Pi y^A.\ B)[\rho(N)/x]\end{aligned}$$

Otherwise,

$$\begin{aligned}\tau((\Pi y^A.\ B)[N/x]) &= \Pi y^{\rho'(A[N/x])}.\ \tau(B[N/x]) \\ &= \Pi y^{\rho'(A)[\rho(N)/x]}.\ \tau(B)[\rho(N)/x] \\ &= \Pi y^{\rho'(A)[\rho(N)/x]}.\ \tau(B)[\rho(N)/x] \\ &= \tau(\Pi y^A.\ B)[\rho(N)/x]\end{aligned}$$

The cases that M is a λ-terms or an application are similar. □

Unfortunately, unlike ρ, the translation τ does not preserve β-reductions. This is by design, as it maintains information about previously performed reductions in the padded expressions. So, to aid the proof below, we use an auxiliary translation θ, which is simply τ without padding. This translation *does* preserve β-reductions, and will act as a convenient intermediate translation.

Definition 27. *Define the functions*

$$\theta_i : \mathsf{T}_{i-1} \to \mathsf{T}_{i-1} \qquad \textit{and} \qquad \theta'_i : \mathsf{T}_{i-1} \to \mathsf{T}_{i-1}$$

simultaneously as follows.

$$\begin{aligned}
\theta_i(s_{i-2}) &\triangleq \bullet_i \\
\theta_i({}^{s_i}x) &\triangleq {}^{s_i}x\bullet_i \\
\theta_i(\Pi x^A.\ B) &\triangleq \begin{cases} \theta_i(A) \to \theta_i(B) & \deg(A) = i-1 \\ \Pi x^{\rho'_i(A)}.\ \theta_i(B) & \textit{otherwise} \end{cases} \\
\theta_i(\lambda x^A.\ M) &\triangleq \lambda x^{\rho'_i(A)}.\ \theta'_i(M)\bullet \\
\theta_i(MN) &\triangleq \theta_i(M)\rho_i(N) \\
\theta'_i(M) &\triangleq \lambda\, \bullet_i^{\perp_i}.\ \theta_i(M)
\end{aligned}$$

where $j \geq i-1$ and $\bullet_i$ is a distinguished variable.

Again, the definition of θ is exactly the same as that of τ, except in the case of λ-expressions, where no padding occurs. In particular, θ does not map expressions to I-expression. But it does have the same sort of substitution commutation and β-reduction preservation lemmas as ρ, and it can further be proved that θ preserves weak normalization.

Lemma 27. *(θ commutes with substitution) For variable ${}^{s_k}x$ and derivable expressions M and N where $\deg(M) = i-1$ and $\deg(N) \geq i$,*

$$\theta(M[N/{}^{s_k}x]) = \theta(M)[\rho(N)/{}^{s_k}x]$$

The proof is nearly identical to that of Lemma 26 above.

Lemma 28. *For derivable expression M and N of degree $i-1$, the following hold.*

1. *If* $M \to_\beta N$, *then* $\theta(M) \twoheadrightarrow_\beta \theta(N)$.

2. $\theta(\mathsf{NF}(M)) \twoheadrightarrow_\beta \mathsf{NF}(\theta(M))$.

Proof. The proof of item 1 is straightforward, modulo Lemma 27. For the proof of item 2 it suffices to note that performing a reduction of the form $\theta'(N)\bullet \to_\beta \theta(N)$ in $\theta(M)$ does not produce any new redexes if $M \in \mathsf{NF}$. □

The translations τ and θ are then connected via an erasure function.

Definition 28. *Define the function* $|\cdot| : \mathsf{T}_{i-1} \to \mathsf{T}_{i-1}$ *which erases padding expressions shallowly, i.e.,*

$$
\begin{aligned}
|\langle P, Q\rangle_A| &\triangleq \bullet_i \\
|s_{i-2}| &\triangleq s_{i-2} \\
|^{s_i}x| &\triangleq {}^{s_i}x \\
|\Pi x^A.\ B| &\triangleq \Pi x^{|A|}.\ |B| \\
|\lambda x^A.\ B| &\triangleq \lambda x^{|A|}.\ |B| \\
|MN| &\triangleq |M||N|
\end{aligned}
$$

Fact 5. $|\tau(M)| = \theta(M)$ *for any expression* M.

Naturally, this erasure function also commutes with substitution.

Fact 6. $|M[N/x]| = |M|[|N|/x]$ *for variable* ${}^{s_k}x$ *and expressions* M *and* N *where* $\deg(M) = i - 1$ *and* $\deg(N) = k - 1$.

The key idea of the following lemma is to build a reduction sequence starting at $\tau(M)$ which follows a reduction sequence starting at M in lockstep, and then argue that the padded information in the reductions of $\tau(M)$ is sufficiently simple so as not to induce any potentially

infinite reduction sequences. The intermediate translation θ is used as a lens to view reduced forms of $\tau(M)$ without the additional padding information.

A padded expression is sufficiently simple it's only non-thunk expressions are of degree at least i. Since λS, and hence $\lambda S^{\neg}$, is i-secure, the padded expressions must be strongly normalizing. Note that $\bullet$, and $\langle M, N\rangle_A$ are both degree $i-1$, so this is not immediate, but it is not difficult to show in the case that λS is not i-negatable.

Definition 29. *An expression M is i-**padded** if $\langle P, Q\rangle_A \subset M$ implies it is of the from*

$$\langle P_1, \langle P_2, \ldots, \langle P_k, \bullet\rangle_{A_k} \cdots \rangle_{A_2}\rangle_{A_1}$$

*where $P_1, \ldots, P_k$ (and hence, $A_1, \ldots, A_k$) are degree at least i. I will call such an expression an i-**padding term**.*

The above idea is captured by the following commutative diagram. Note that there are two different kinds of arrows in this diagram; horizontal arrows are for β-reductions and vertical arrows are for translations. In essense, the lemma says that, although τ does not preserve β-reduction in the same way that θ does, it can be made to preserve it up to ρ-padding, which is sufficient since $\tau(M)$ is ρ-padded by design.

Lemma 29. *Let M, M' and N be derivable expressions such that $M \to_\beta M'$ and $|N| = \theta(M)$ and N is i-padded. Then there is a derivable expression N' such that N' is i-padded and the following diagram commutes.*

$$\begin{array}{ccc} M & \xrightarrow{\beta} & M' \\ \downarrow{\scriptstyle\theta} & & \downarrow{\scriptstyle\theta} \\ \theta(M) & \xrightarrow{\beta}\!\!\!\!\!\rightarrow & \theta(M') \\ \uparrow{\scriptstyle|\cdot|} & & \uparrow{\scriptstyle|\cdot|} \\ N & \overset{\beta}{\cdots\cdots\!\!\!\!\!\rightarrow} & N' \end{array}$$

Proof. Let C be a single-holed expression such that $M = C\langle(\lambda x^A.\ P)Q\rangle)$ and $M' = C\langle P[Q/x]\rangle$, *i.e.*, $(\lambda x^A.\ P)Q$ is the redex along which M is reduced to M'. By definition of

θ, there is a single-holed expression C' such that

$$\theta(M) = C'\langle(\lambda x^{\rho'(A)}.\ (\lambda\ \bullet^{\perp}\ .\ \theta(P))\bullet\rho(Q)\rangle$$

By definition of $|\cdot|$, there is some context C'' such that

$$N = C''\langle(\lambda x^{\rho'(A)}.\ (\lambda\ \bullet^{\perp}\ .\ P')Z)\rho(Q)\rangle$$

where $|P'| = \theta(P)$ and $|Z| = \bullet$. Note that, since $|\cdot|$ only removes padding information, we know that this redex must appear in N (*i.e.*, it is not erased by $|\cdot|$) and that it does not affect ρ-translated expressions. Since N is i-padded, it must be that

$$Z = \langle T_1, \langle T_2, \ldots, \langle T_k, \bullet\rangle_{A_k} \cdots\rangle_{A_2}\rangle_{A_1}$$

and Z is an i-padding term. Then

$$\begin{aligned} C''\langle(\lambda x^{\rho'(A)}.\ (\lambda\ \bullet^{\perp}\ .\ P')Z)\rho(Q)\rangle &\to_\beta C''\langle(\lambda\ \bullet^{\perp}\ .\ P'[\rho(Q)/x])Z[\rho(Q)/x]\rangle \\ &\to_\beta C''\langle P'[\rho(Q)/x][Z[\rho(Q)/x]/\bullet]\rangle \end{aligned}$$

Take N' to be this last expression, which is i-padded because degree is preserved by substitution (which handles the first reduction) and substitution $\bullet$ in an i-padding expression with an i-padding expression clearly preserving i-padding. Finally,

$$\begin{aligned} |C''\langle P'[\rho(Q)/x])[Z[\rho(Q)/x]/\bullet]\rangle| &= C'\langle\theta(P[Q/x])[|Z[\rho(Q)/x]|/\bullet]\rangle \\ &= C'\langle\theta(P[Q/x])[\bullet/\bullet]\rangle \\ &= C'\langle\theta(P[Q/x])\rangle \\ &= \theta(C\langle P[Q/x]\rangle) \end{aligned}$$

$$
\begin{aligned}
|C''\langle P'[\rho(Q)/x]\rangle| &= C'\langle |P'[\rho(Q)/x]|\rangle \\
&= C'\langle \theta(P)[\rho(Q)/x]\rangle \\
&= C'\langle \theta(P[Q/x])\rangle \\
&= \theta(C\langle P[Q/x]\rangle)
\end{aligned}
$$

□

Lemma 30. *Let M and M' be derivable expressions such that $M \twoheadrightarrow_\beta M'$. Then there is an expression i-padded N such that the following diagram commutes.*

$$
\begin{array}{ccc}
M & \xrightarrow{\beta} & M' \\
\big\downarrow{\scriptstyle \theta} & & \big\downarrow{\scriptstyle \theta} \\
\theta(M) & \xrightarrow{\beta} & \theta(M') \\
\big\uparrow{\scriptstyle |\cdot|} & & \big\uparrow{\scriptstyle |\cdot|} \\
\tau(M) & \overset{\beta}{\cdots\!\!\twoheadrightarrow} & N
\end{array}
$$

Proof. By a diagram chase with Lemma 29. Note that $\tau(M)$ is i-padded by construction. □

All together, this gives us the next key lemma.

Lemma 31. *For any derivable expression M such that $\deg(M) = i - 1$, if $M \in \mathsf{WN}$ then $\tau(M) \in \mathsf{WN}$.*

Proof. Consider a reduction sequence from M to $\mathsf{NF}(M)$. By Lemma 30, there is an reduct N of $\tau(M)$ such that $|N| = \theta(\mathsf{NF}(M))$. Following the reduction sequence $\theta(\mathsf{NF}(M)) \twoheadrightarrow_{\mathsf{NF}} (\theta(M))$ given by Lemma 57 analogously in N yields and expression N' such that $|N'| = \mathsf{NF}(\theta(M))$. Thus, the only sub-expressions of N' which have redexes are i-padding expressions, which are strongly normalizing by Lemma 25. □

Infinite Reduction Sequence Preservation. Finally, we prove $\mu(M) \leq \mu(\tau(M))$. Note that I've already stated the upper bound for ρ (Fact 2). We take advantage of a standard structural lemma which is also proved by BHS (Lemma 5.7).

Lemma 32. *If M is derivable then M is a sort, a Π-expressions, or of the form*

$$NM_1 \dots M_k$$

where $k \geq 0$ and N is a variable or a λ-expression.

Lemma 33. *For any derivable expression M with $\deg(M) = i - 1$,*

$$\mu(M) \leq \mu(\tau(M)) = \mu(\tau'(M))$$

Proof. By induction on $\mu(\tau(M))$ and the structure of M, lexicographically ordered. The cases in which M is a sort, a variable, or a Π-expression are straightforward.

λ-Expression. Suppose M is of the form $\lambda x^A.\ N$. By Fact 2, $\mu(A) \leq \mu(\rho'(A))$ and by the inductive hypothesis, $\mu(N) \leq \mu(\tau(N)) = \mu(\tau'(N))$, so

$$\begin{aligned}\mu(\lambda x^A.\ N) &= \mu(A) + \mu(N) \\ &\leq \mu(\rho'(A)) + \mu(\tau(N)) = \mu(\rho'(A)) + \mu(\tau'(N))\end{aligned}$$

And since $\mu(\tau'(N)) \leq \mu(\tau'(N)\langle x, \bullet\rangle_{\rho'(A)})$ we also have that $\mu(\lambda x^A.\ N) \leq \mu(\tau(\lambda x^A.\ N))$.

Application. We consider two cases. First suppose that M is of the form $xN_1 \dots, N_k$ where $k \geq 1$. Then

$$\begin{aligned}\mu(xN_1 \dots N_k) &= \sum_{j=1}^{k} \mu(N_j) \\ &\leq \sum_{j=1}^{k} \mu(\rho(N_j)) \\ &= \mu(\tau(xN_1 \dots N_k)) = \mu(\tau'(xN_1 \dots N_k))\end{aligned}$$

Otherwise, M is of the form $(\lambda x^A.\ P)N_1 \dots N_k$ with $\deg(P) = i - 1$. Consider the reduction

sequence

$$
\begin{aligned}
&\tau((\lambda x^{A}.\ P)N_1 \dots N_k) \\
&= (\lambda x^{\rho'(A)}.\ \tau'(P)\langle x, \bullet\rangle_{\rho'(A)})\rho(N_1), \dots \rho(N_k) \\
&\twoheadrightarrow_\beta (\lambda x^{\mathsf{NF}(\rho'(A))}.\ \tau'(P)\langle x, \bullet\rangle_{\rho'(A)})\rho(N_1), \dots \rho(N_k) \\
&\to_\beta \tau'(P)[\rho(N_1)/x]\langle \rho(N_1), \bullet\rangle_{\rho'(A)}\rho(N_2) \dots \rho(N_k) \\
&= \tau'(P[N_1/x])\langle \rho(N_1), \bullet\rangle_{\rho'(A)}\rho(N_2) \dots \rho(N_k) \\
&\twoheadrightarrow_\beta \tau'(P[N_1/x])\langle \mathsf{NF}(\rho(N_1)), \bullet\rangle_{\rho'(A)}\rho(N_2) \dots \rho(N_k) \\
&\to_\beta \tau(P[N_1/x])[\langle \mathsf{NF}(\rho(N_1)), \bullet\rangle_{\rho'(A)}/\bullet]\rho(N_2) \dots \rho(N_k)
\end{aligned}
$$

where the above normal forms are guaranteed to exist by the assumption of i-security and Lemma 25. Note that the fifth line follows from Lemma 26. This reduction has length at least

$$
\begin{aligned}
&2 + \mu(\rho'(A)) \\
&\quad + \mu(\rho(N_1)) \\
&\quad + \mu(\tau(P[N_1/x][\langle \mathsf{NF}(N_1), \bullet\rangle_{\rho'(A)}/\bullet])\rho(N_2) \dots \rho(N_k))
\end{aligned}
$$

and is upper bounded by $\mu(\tau(M))$. Furthermore,

$$
\begin{aligned}
&\mu(\tau((P[N_1/x])N_2 \dots N_k)) \\
&= \mu(\tau(P[N_1/x])\rho(N_2) \dots \rho(N_k)) \\
&\leq \mu(\tau(P[N_1/x][\langle \mathsf{NF}(N_1), \bullet\rangle_{\rho'(A)}/\bullet])\rho(N_2) \dots \rho(N_k))
\end{aligned}
$$

since $\bullet$ can be replaced with $\langle \mathsf{NF}(N_1), \bullet\rangle_{\rho'(A)}$ in any reduction sequence starting at $\tau(P[N_1/x])\pi(N_2) \dots \pi(N$

So applying the inductive hypothesis, the above expression is lower bounded by

$$1 + \mu(A) + \mu(N_1) + \mu((P[N_1/x])N_2 \dots N_k)$$

Note that this is why we cannot simply induct over the structure of M, as we need to be able to say that

$$\mu((P[N_1/x])N_2 \dots N_k) \leq \mu(\tau((P[N_1/x])N_2 \dots N_k))$$

Finally, by Lemma 13, this implies $\mu(M) \leq \mu(\tau(M))$. □

I conclude with the main lemma of this section, which derives $(i-1)$-security from i-security.

Lemma 34. *Let λS be an i-secure, weakly normalizing n-tiered pure type system which is i-weakly clean and not i-negatable. Then every derivable expression M such that $\deg(M) = i - 1$ is strongly normalizing. That is, λS is $(i-1)$-secure.*

The Negatable Case

At this point, we can prove that weak normalization implies strong normalization for all non-dependent weakly clean n-tiered pure type systems. In the previous report I have been alluding to Mull [2022], I use a very close variant of the translation from the previous section to given the same result in the case that s_i is negatable. In addition, for negatable sorts I give a different closure property using a novel padding technique based roughly on the notion of Skolem functions. This translation preserves typability in λS, so will not require the combinatorial-style proofs as in the previous section.

I begin with the technical lemma that makes possible the padding technique. It is based on the observation that if it is possible to abstract over a variable, then it is also possi-

ble to generalize over that variable in the types of padding variables. So a variable can commute with the padding variables *modulo generalization over that variable*. The generalization allows padding variables to instantiate their dependencies at the level of terms, which eliminates the dependence in the types.

This idea can almost eliminate the need for any notion of cleanliness, but unfortunately, the situation is complicated by products types. On the one hand, product types are simpler because they cannot be "reduced" at the level of terms except at their individual components; substitution only occurs at the type level with application steps. The difficulty is that a type of degree $i-1$ may include generalization using the rule (s_j, s_{i-1}), and it is not possible to generalize over padding variables with this rule (the types of padding variables are degree i), so we may not be able to use the Skolemization-style argument referenced above. So the translation I present still requires a cleanliness-like restriction on the rules in the system, but it is weaker.

Definition 30. *A non-dependent n-tiered pure type system λS is i-**upwards clean** if it satisfies the following closure property: For any j (where $j \geq i$) in $[n]$, if $(s_j, s_{i-1}) \in \mathcal{R}_{\lambda S}$, then $(s_j, s_i) \in \mathcal{R}_{\lambda S}$ or $j \notin \mathcal{D}_{\lambda S}(i)$.*

The first condition allows for the same technique for abstraction to be used for product type formation, *i.e.*, the Skolemization-style argument. The second condition is based Lemma 12, and is essentially a stronger fall-back notion of full cleanliness which captures when variables commute by virtue of the structure of the rules, as was done in the previous section. Also note that cleanliness implies i-very weak cleanliness in the case that s_i is negatable, as the existence of both the rules (s_j, s_{i-1}) and (s_i, s_i) imply the generalizability of rules (s_k, s_i) for $i \leq k \leq i-1$. This not the case for weak-cleanliness, which also shows the strength of the notion of weak harmlessness.

I continue to the above mentioned technical lemma. The following is a useful definition and notation for generalizing over many variables at a time.

Definition 31.

- *A context* Γ *is* ***generalizable over a sort*** s_j *in* λS *if* $(s_k, s_j) \in \mathcal{R}_{\lambda S}$ *for each statement* $({}^{s_k}x : A)$ *appearing in* Γ. *In abuse of notation, for a context* Γ *of the form* $(x_1 : A_1, \ldots, x_k : A_k)$, *I write* $\Pi\Gamma.\ B$ *for*

$$\Pi {x_2}^{A_1}.\ \ldots \Pi {x_k}^{A_k}.\ B$$

  and I write $M\Gamma$ *for* $Mx_1 \ldots x_k$.

- *A context* Γ *is* ***generalizable over a context*** Φ *if* Γ *is generalizable over each* s_j *such that* $({}^{s_j}x : A)$ *appears in* Φ. *In further abuse of notation, for a context* Φ *of the form* $(x_1 : B_1, \ldots, x_k : B_k)$, *I write* $\Pi\Gamma.\ \Phi$ *for*

$$(x_1 : \Pi\Gamma.\ B_1, \ldots, x_k : \Pi\Gamma.\ B_k).$$

The basic idea is the following. Suppose

$$\Gamma, x : A, y : B \vdash M(x, y) : C(x, y)^{2}$$

and $\Pi x^A.\ B$ is a derivable type. It is then possible to derive

$$\Gamma, y : \Pi x^A.\ B, x : A \vdash M(x, yx) : C(x, yx).$$

The variable y is replaced with yx to recover its type in the first derivation. This is a fairly simple syntactic trick, but it has some very nice consequences. The dependence on x in the type of y is represented at *the level of terms* instead of types. So if it is possible to abstract

2. I am using this notion to emphasize the dependence on x and y.

over x, then we can derive

$$\Gamma, y : \Pi x^A.\ B \vdash \lambda x^A.\ M(x, yx) : \Pi x^A.\ C(x, yx).$$

And if this λ-expression is applied to another expression N and then β-reduced, the subexpression yx is replaced with yN, maintaining the coupling of y and x without needing it to be expressed in the context.

What follows is the natural generalization of this idea. The translation below replaces variables whose types may have changed with function applications to recover their original type.

Definition 32. *For contexts Γ and Ψ, define the function $\xi^\Gamma_\Psi : \mathsf{T} \to \mathsf{T}$ as follows.*

$$\xi^\Gamma_\Psi(s_j) \triangleq s_j$$

$$\xi^\Gamma_\Psi(x) \triangleq \begin{cases} x\Gamma & x \in \Psi \\ x & \textit{otherwise} \end{cases}$$

$$\xi^\Gamma_\Psi(\Pi x^A.\ B) \triangleq \Pi x^{\xi^\Gamma_\Psi(A)}.\ \xi^\Gamma_\Psi(B)$$

$$\xi^\Gamma_\Psi(\lambda x^A.\ M) \triangleq \Pi x^{\xi^\Gamma_\Psi(A)}.\ \xi^\Gamma_\Psi(M)$$

$$\xi^\Gamma_\Psi(MN) \triangleq \xi^\Gamma_\Psi(M)\xi^\Gamma_\Psi(N)$$

Note that the in Ψ are irrelevant, but are included for convenience. I will write Ψ^Γ_y in the case Ψ is of the form $(y : A)$.

This translation is sufficiently simple so as to satisfy the standard substitution-commutation and β-preservation lemmas fairly straightforwardly.

Fact 7. *(ξ^Γ_y commutes with substitution) For expressions M and N, variables x and y, and context Γ,*

$$\xi^\Gamma_y(M[N/x]) = \xi^\Gamma_y(M)[\xi^\Gamma_y(N)/x].$$

Fact 8. *(ξ_y^Γ preserves β-reductions) For expressions M and N, if $M \to_\beta N$, then $\xi_y^\Gamma(M) \to_\beta \xi_y^\Gamma(N)$. Furthermore, if $M =_\beta N$, then $\xi_y^\Gamma(M) =_\beta \xi_y^\Gamma(N)$.*

And now for the lemma itself. First, the simple case in which the context on the right is a single variable. The more general case is a corollary. The proof follows the usual pattern. I also note that these results can be further generalized to the case of any pure type system, but it is simpler to present it here in the setting of tiered systems. Note that the ξ-translation is extended to contexts in the usual way.

Lemma 35. *Let Γ, Ξ, and Φ be contexts, let ${}^{s_j}x$ be variable a such that Ξ is generalizable over s_j, and let A, B and M be expressions. Then*

$$\Gamma, \Xi, {}^{s_j}x : A, \Phi \vdash M : B$$

implies

$$\Gamma, {}^{s_j}x : \Pi\Xi.\ A, \Xi, \xi_x^\Xi(\Phi) \vdash \xi_x^\Xi(M) : \xi_x^\Xi(C).$$

Proof. By induction on the structure of derivations. The cases in which the last inference is an axiom, product type formation, or an abstraction are straightforward.

Variable Introduction. First suppose $\Phi = \varnothing$. Then the last inference is of the form

$$\frac{\Gamma, \Xi \vdash A : s_j}{\Gamma, \Xi, {}^{s_j}x : A \vdash {}^{s_j}x : A}$$

Since Ξ is generalizable over s_j, it is easy to get to the derivation

$$\frac{\Gamma \vdash \Pi\Xi.\ A : s_j}{\Gamma, {}^{s_j}x : \Pi\Xi.\ A \vdash {}^{s_j}x : \Pi\Xi.\ A}$$

by a sequence of product type formations and a variable introduction. Then by thinning can derive

$$\Gamma, {}^{s_j}x : \Pi\Xi.\ A, \Xi \vdash {}^{s_j}x : \Pi\Xi.\ A$$

It is then similarly easy to derive

$$\Gamma, {}^{s_j}x : \Pi\Xi.\ A, \Xi \vdash {}^{s_j}x\Xi : A$$

by a sequence of applications. Note that x does not appear in A.

Next suppose Φ is nonempty. Then the last inference is of the form

$$\frac{\Gamma, \Xi, {}^{s_j}x : A, \Phi' \vdash B : s_k}{\Gamma, \Xi, {}^{s_j}x : A, \Phi', {}^{s_k}y : B \vdash {}^{s_k}y : B}$$

Then by induction on the length of Φ, we can derive

$$\frac{\Gamma, {}^{s_j}x : \Pi\Xi.\ A, \Xi, \xi_x^{\Xi}(\Phi') \vdash \xi_x^{\Xi}(B) : s_k}{\Gamma, {}^{s_j}x : \Pi\Xi.\ A, \Xi, \xi_x^{\Xi}(\Phi'), {}^{s_k}y : \xi_x^{\Xi}(B) \vdash {}^{s_k}y : \xi_x^{\Xi}(B)}$$

Weakening. Again, first suppose $\Phi = \varnothing$. Then the last inference is of the form

$$\frac{\Gamma, \Xi \vdash M : B \qquad \Gamma, \Xi \vdash A : s_j}{\Gamma, \Xi, {}^{s_j}x : B \vdash M : B}$$

As in the previous case, we can derive

$$\Gamma \vdash \Pi\Xi.\ A : s_j$$

which, by thinning, implies

$$\Gamma, {}^{s_j}x : \Pi\Xi.\ A, \Xi \vdash M : B$$

The case in which Φ is nonempty is similar to the analogous case for variable introduction.

Application. Suppose the last inference is of the form

$$\frac{\Gamma, \Xi, {}^{s_j}x : A, \Phi \vdash P : \Pi y^C.\ D \qquad \Gamma, \Xi, {}^{s_j}x : A, \Phi \vdash Q : C}{\Gamma, \Xi, {}^{s_j}x : A, \Phi \vdash PQ : D[Q/y]}$$

This case follows by applying the inductive hypothesis to each antecedant judgment and then applying substitution preservation for ξ_x^{Ξ} (Fact 7) to ensure that the type of the conclusion is correct.

Conversion. As with the previous case, we can apply in the inductive hypothesis to each antecedent judgment and then apply β-preservation for ξ_x^{Ξ} (Fact 8) to ensure that the type of the conclusion is correct. □

Corollary 1. *Let Γ, Ξ, Ψ, and Φ be contexts, such that Ξ is generalizable over Ψ (and, equivalently, over $\xi_{\Psi}^{\Xi}(\Psi)$), and let M and A be expressions. Then*

$$\Gamma, \Xi, \Psi, \Phi \vdash M : A$$

implies

$$\Gamma, \Pi\Xi.\ \xi_{\Psi}^{\Xi}(\Psi), \Xi, \xi_{\Psi}^{\Xi}(\Phi) \vdash \xi_{\Psi}^{\Xi}(M) : \xi_{\Psi}^{\Xi}(A).$$

Proof. By repeated application of Lemma 35. □

Next, we define the padding functions used in the main translation below. The types that require padding functions are those which appear as the arguments of abstractions, and these are inductively build up based on the term being translated; this accounts for the dependence on the expression M in the definition below. The dependence on the context Γ below keeps track of how the above lemma has been applied in previous steps of the translation. Also note that there are no polymorphic padding variables. Though it would be possible to include them, it is not terribly useful in combination with the Skolemization-style argument.

Definition 33. *Define the contexts $\Upsilon_{i,M}^{\Gamma}$ as follows by simultaneous induction on the struc-*

ture of M and length of Γ, lexicographically ordered.

$$
\begin{aligned}
\Upsilon^{\Gamma}_{i,s_{i-2}} &\triangleq \varnothing \\
\Upsilon^{\Gamma}_{i,s_i x} &\triangleq \varnothing \\
\Upsilon^{\Gamma}_{i,\Pi^{s_j} x^A.\ B} &\triangleq \begin{cases} \Upsilon^{\Gamma,x:A}_{i,B} & (s_j, s_i) \in \mathcal{R}_{\lambda S} \\ \Upsilon^{\Gamma}_{i,A}, \Upsilon^{\Gamma}_{i,B} & \textit{otherwise} \end{cases} \\
\Upsilon^{\Gamma}_{i,\lambda x^A.\ M} &\triangleq \mathsf{pad}^{\Gamma}_{x} : \Pi \rho'(\Gamma).\ \rho'_i(A) \to \bot_i \to \bot_i, \Upsilon^{\Gamma,x:A}_{i,M} \\
\Upsilon^{\Gamma}_{i,MN} &\triangleq \Upsilon^{\Gamma}_{i,M}, \Upsilon^{\Gamma}_{i,N}
\end{aligned}
$$

Similar to what is done in the previous section, I use the shorthand $\langle M, N\rangle^{\Gamma}_{x}$ for $\mathsf{pad}^{\Gamma}_{x}\rho'(\Gamma)MN$.

It is possible to characterize the dependence on Γ using the generalized Π-bindings from Definition 31.

Lemma 36. *For any expression M and contexts Γ and Φ,*

$$\Upsilon^{\Gamma,\Phi}_{M} = \Pi\Gamma.\ \Upsilon^{\Phi}_{M}.$$

Proof. By induction on the structure of M. The cases in which M is a sort, variable or application are straightforward

Π-expression. Suppose M is of the form $\Pi x^A.\ B$. If $(s_j, s_i) \in \mathcal{R}_{\lambda S}$ then

$$
\begin{aligned}
\Upsilon^{\Gamma,\Phi}_{\Pi x^A.\ B} &= \Upsilon^{\Gamma,\Phi,x:A}_{B} \\
&= \Pi\Gamma.\ \Upsilon^{\Phi,x:A}_{B} \\
&= \Pi\Gamma.\ \Upsilon^{\Phi}_{\Pi x^A.\ B}
\end{aligned}
$$

The other cases are straightforward.

λ-expression. Suppose M is of the form $\lambda x^A.\ B$. Then

$$\begin{aligned}
\Upsilon^{\Gamma,\Phi}_{\lambda x^A.\ B} &= \mathsf{pad}_x : \Pi(\Gamma,\Phi).\ \rho'(A) \to \bot \to \bot, \Upsilon^{\Gamma,\Phi,x:A}_{B} \\
&= \mathsf{pad}_x : \Pi(\Gamma,\Phi).\ \rho'(A) \to \bot \to \bot, \Pi\Gamma.\ \Upsilon^{\Phi,x:A}_{B} \\
&= \Pi\Gamma.\ (\mathsf{pad}_x : \Pi\Phi.\ \rho'(A) \to \bot \to \bot, \Upsilon^{\Phi,x:A}_{B}) \\
&= \Pi\Gamma.\ \Upsilon^{\Phi}_{\lambda x^A.\ B}
\end{aligned}$$

□

Finally, the main translation of the section. It is similar to the translation in the previous section but with a dependence on contexts. The contexts play a similar role here as they do in the definitions above; they explicitly keep track of what needs to be ξ-translated.

Definition 34. *For each context* Γ*, define the functions*

$$\tau^{\Gamma}_i : \mathsf{T}_{i-1} \to \mathsf{T} \qquad \textit{and} \qquad \tau'^{\Gamma}_i : \mathsf{T}_{i-1} \to \mathsf{T}$$

as follows by induction on the structure of its argument and the length of Γ*, lexicographically*

ordered.

$$
\begin{aligned}
\tau_i^\Gamma(s_{i-2}) &\triangleq \bullet_i \\
\tau_i^\Gamma({}^{s_i}x) &\triangleq {}^{s_i}x\bullet_i \\
\tau_i^\Gamma(\Pi x^A.\ B) &\triangleq \begin{cases} \tau_i^\Gamma(A) \to \tau_i^\Gamma(B) & \deg(A) = i-1 \\ \Pi x^{\rho_i'(A)}.\ \tau_i^{\Gamma,x:A}(B) & (s_j, s_i) \in \mathcal{R}_{\lambda S} \\ \Pi x^{\rho_i'(A)}.\ \tau_i^\Gamma(B) & \textit{otherwise} \end{cases} \\
\tau_i^\Gamma(\lambda x^A.\ M) &\triangleq \lambda x^{\rho_i'(A)}.\ \tau_i'^{\Gamma,x:A}(M)\langle x, \bullet_i\rangle_x^\Gamma \\
\tau_i^\Gamma(MN) &\triangleq \begin{cases} \tau_i^\Gamma(M)\tau_i'^\Gamma(N) & \deg(N) = i-1 \\ \tau_i^\Gamma(M)\rho_i(N) & \textit{otherwise} \end{cases} \\
\tau_i'^\Gamma(M) &\triangleq \lambda\ \bullet_i^{\perp_i}\ .\ \tau_i^\Gamma(M)
\end{aligned}
$$

where $j \geq i-1$ and $\bullet_i$ is a distinguished variable.

As above, it is useful to characterize the dependence on Γ, this time with relation to the ξ-translation.

Lemma 37. *For contexts Γ and Φ and expression M,*

$$\tau^{\Gamma,\Phi}(M) = \xi^\Gamma_{\Upsilon^\Phi_M}(\tau^\Phi(M))$$

Proof. By induction on the structure M. The only case of interest is the one in which M is of the form $\lambda x^A.\ B$. In this case,

$$\tau^{\Gamma,\Phi}(\lambda x^A.\ B) = \lambda x^{\rho'(A)}.\ \tau'^{\Gamma,\Phi,x:A}(M)(\mathsf{pad}_x^{\Gamma,\Phi}\rho'(\Gamma,\Phi)x\bullet)$$

By the inductive hypothesis,

$$\tau^{\Gamma,\Phi,x:A}(M) = \xi^{\Gamma}_{\Upsilon^{\Phi,x:A}_{M}}(\tau^{\Phi}(M))$$

and since pad_x does not appear in $\tau^{\Phi}(M)$, we have

$$\xi^{\Gamma}_{\Upsilon^{\Phi,x:A}_{M}}(\tau^{\Phi}(M)) = \xi^{\Gamma}_{\Upsilon^{\Phi}_{\lambda x^A.\ M}}(\tau^{\Phi}(M))$$

It is also straightforward to check that

$$\mathsf{pad}_x^{\Gamma,\Phi}\rho'(\Gamma,\Phi)x\bullet = \xi^{\Gamma}_{\Upsilon^{\Phi}_{\lambda x^A.\ M}}(\mathsf{pad}_x^{\Phi}\rho'(\Phi)x\bullet)$$

and

$$\rho'(A) = \xi^{\Gamma}_{\Upsilon^{\Phi}_{\lambda x^A.\ M}}(\rho'(A))$$

So by definition of the ξ-translation, we have

$$\begin{aligned}
&\tau^{\Gamma,\Phi}(\lambda x^A.\ B)\\
&= \lambda x^{\rho'(A)}.\ \tau'^{\Gamma,\Phi,x:A}(M)\langle x,\bullet\rangle_x^{\Gamma,\Phi}\\
&= \xi^{\Gamma}_{\Upsilon^{\Phi}_{\lambda x^A.\ M}}(\lambda x^{\rho'(A)}.\ \tau'^{\Phi,x:A}(M)\langle x,\bullet\rangle_x^{\Phi})\\
&= \xi^{\Gamma}_{\Upsilon^{\Phi}_{\lambda x^A.\ M}}(\tau^{\Phi}(\lambda x^A.\ B))
\end{aligned}$$

□

As in the non-negatable case, the translation must preserve typability, but this time in the same system. Note that this lemma allows for commutation with the padding variables.

Lemma 38. *Let λS be a non-dependent tiered pure type system and suppose $\Gamma \vdash_{\lambda S} M : A$ and $\Gamma \vdash_{\lambda S} A : s_i$. Further suppose that Φ is a context which is generalizable over s_i. Then*

$\Gamma, \Phi \vdash_{\lambda S} M : A$ *implies*

$$\Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon^{\Phi}_{M}, \rho'(\Phi) \vdash_{\lambda S} \tau^{\Phi}(M) : \rho(M)$$

Proof. By induction on the structure of derivations and the length of Φ, lexicographically ordered. I focus on the cases of product type formation and abstraction, which are notably different than the analogous cases in Lemma 20.

Product Type Formation. Suppose the last inference is of the form

$$\frac{\Gamma, \Phi \vdash A : s_j \qquad \Gamma, \Phi, {}^{s_j}x : A \vdash B : s_{i-1}}{\Gamma, \Phi \vdash \Pi x^{A}.\ B : s_{i-1}}$$

The case in which $\deg(A) = i - 1$ is similar to the analogous case in Lemma 20, so suppose $\deg(A) > i - 1$. There are two remaining cases to consider. First, if $(s_j, s_i) \in \mathcal{R}_{\lambda S}$, then $\Phi, x : A$ is generalizable over s_i, which means

$$\Delta, \bullet : \bot, \rho'(\Gamma), \Upsilon^{\Phi, x:A}_{B}, \rho'(\Phi), x : \rho'(A) \vdash \tau_i^{\Phi, x:A}(B) : s_{i-1}$$

Otherwise, $j \notin \mathcal{D}_{\lambda S}(i)$ which implies $j \notin \mathcal{D}_{\lambda S}(k)$ for any rules $(s_k, s_i$ with $i \leq k \leq j$ and by an argument similar to the one for Lemma 19, the variable x does not appear free in $\Upsilon^{\varnothing}_{B}$. So by permutation and the inductive hypothesis,

$$\Delta, \bullet : \bot, \rho'(\Gamma), \rho'(\Phi), \Upsilon^{\varnothing}_{B}, x : \rho'(A) \vdash \tau(B) : s_{i-1}$$

and by product type formation, it is possible to derive

$$\Delta, \bullet : \bot, \rho'(\Gamma), \rho'(\Phi), \Upsilon^{\varnothing}_{B} \vdash \Pi x^{\rho'(A)}.\ \tau(B) : s_{i-1}$$

Finally by Corollary 1, we can derive

$$\Delta, \bullet : \bot, \rho'(\Gamma), \Pi\Phi.\ \Upsilon^{\varnothing}_{\Pi x^A.\ B}, \rho'(\Phi) \vdash \xi^{\Phi}_{\Upsilon^{\varnothing}_{\Pi x^A.\ B}}(\Pi x^{\rho'(A)}.\ \tau(B)) : s_{i-1}$$

where

$$\Pi\Phi.\ \Upsilon^{\varnothing}_{\Pi x^A.\ B} = \Upsilon^{\Phi}_{\Pi x^A.\ B}$$

by Lemma 36 and

$$\xi^{\Phi}_{\Upsilon^{\varnothing}_{\Pi x^A.\ B}}(\Pi x^{\rho'(A)}.\ \tau(B)) = \tau^{\Phi}(\Pi x^A.\ B)$$

by Lemma 37.

Abstraction. Suppose the last inference is of the form

$$\frac{\Gamma, \Phi, {}^{s_j}x : A \vdash M : B \qquad \Gamma, \Phi \vdash \Pi x^A.\ B : s_i}{\Gamma, \Phi \vdash \lambda x^A.\ M : \Pi x^A.\ B}$$

By the inductive hypothesis, we have

$$\Delta, \rho'(\Gamma), \rho'(\Phi) \vdash \rho'(A) : s_j$$

and

$$\Delta, \rho'(\Gamma), \Upsilon^{\Phi, x:A}_{M}, \rho'(\Phi), x : \rho'(A) \vdash \tau'^{\Phi, x:A}(M) : \bot \to \rho(A)$$

From the first we can derive by thinning and product type formation

$$\Delta, \bullet : \bot, \rho'(\Gamma) \vdash \Pi\rho'(\Phi).\ \rho'(A) \to \bot \to \bot : s_i$$

which by thinning can be added to the context of the second:

$$\begin{aligned}\Delta, \bullet : \bot, \rho'(\Gamma), \mathsf{pad}_x : \Pi\rho'(\Phi).\ \rho'(A) \to \bot \to \bot, \Upsilon^{\Phi, x:A}_{M}, \rho'(\Phi), x : \rho'(A)\\ \vdash \tau'^{\Phi, x:A}(M) : \bot \to \rho(A)\end{aligned}$$

By thinning and repeated application, we can derive

$$\Delta, \bullet : \bot, \rho'(\Gamma), \mathsf{pad}_x : \Pi\rho'(\Phi).\ \rho'(A) \to \bot \to \bot, \Upsilon_M^{\Phi, x:A}, \rho'(\Phi), x : \rho'(A)$$
$$\vdash \mathsf{pad}_x \rho'(\Phi) x \bullet : \bot$$

and by application and abstraction we have

$$\Delta, \bullet : \bot, \rho'(\Gamma), \mathsf{pad}_x : \Pi\rho'(\Phi).\ \rho'(A) \to \bot \to \bot, \Upsilon_M^{\Phi, x:A}, \rho'(\Phi)$$
$$\vdash \lambda x^{\rho'(A)}.\ \tau'(M)\langle x, \bullet \rangle_x^{\Phi} : \rho(A)$$

□

The last two lemmas follow the same pattern of the previous section, but with updates in the case of product types. See also my related research report Mull [2022] for further details.

Lemma 39. *For variable ${}^{s_k}x$ and expressions M and N where* $\deg(M) = i-1$ *and* $\deg(N) = k-1$*, the following hold.*

1. *If $k = i$, then $\tau^{\Phi}(M[N/{}^{s_k}x]) \twoheadleftarrow_{\beta} \tau^{\Phi}(M)[\tau'^{\Phi}(N)/{}^{s_k}x]$.*
2. *If $k > i$, then $\tau^{\Phi}(M[N/{}^{s_k}x]) = \tau^{\Phi}(M)[\rho(N)/{}^{s_k}x]$.*

Lemma 40. *Let λS be a weakly normalizing non-dependent tiered pure type system. For any derivable expression M with* $\deg(M) = i - 1$*, we have*

$$\mu(M) \leq \mu(\tau(M)) = \mu(\tau'(M))$$

And, finally, the corresponding lemma for the negatable case.

Lemma 41. *Let λS be an i-secure weakly normalizing n-tiered pure type system which is i-negatable, and i-downward clean. Then every derivable expression M such that* $\deg(M) = i - 1$ *is strongly normalizing. That is, λS is $(i-1)$-secure.*

2.3.3 The Main Result

I conclude this section with a statement of the main result of this paper, a special case of the Barendregt-Geuvers-Klop conjecture, which is an extention of the result of the BHS result.

Theorem 2. *Let λS be a non-dependent n-tiered pure type system which is i-weakly clean or i-negatable and i-downward clean for each $i \in [n-1]$. If λS is weakly normalizing then λS is strongly normalizing.*

Proof. By reverse induction on i (from n to 0) it follows that λS is i-secure, *i.e.*, all expressions in $\mathsf{T}_{\geq i}$ are strongly normalizing. By the classification lemma (Lemma 9), the only expressions of degree n are types and by the top-sort lemma (Lemma 7) and non-dependence, they are generated at most by s_{n-1} and Π-types using the rule (s_n, s_n) if it appears in $\mathcal{R}_{\lambda S}$. It is straightforward to verify that this set of types is strongly normalizing. So suppose that λS is k-secure. It then follows directly from Lemma 34 and Lemma 41 tha λS is $(k-1)$-secure. □

2.4 Conclusions

I have presented an generalization of Xi's thunkification to a class of non-dependent persisent pure type systems which extends the BHS result. This class unfortunately does not contain *all* non-dependent persistent pure type sytems, but it demonstrates that progress can be made, that the state of the Barendregt-Geuvers-Klop conjecture does not need to be stagnate. I've also presented the class of tiered pure type systems, which I believe to be an important target for further work regarding the conjecture. I should note that the solving the conjecture is not so important, but rather the general question: is it possible for a typed expression to be weakly normalizing but not strongly normalizing? What would such an expression have to look like? And, furthermore, can this normalizing behavior be reflected in another system via translation? What is are the limitations of type preserving translations of this form?

A final note on a feature which is not immediately obvious about the thunkification translation versus the CPS translation. One somewhat unfortunate feature of CPS translations is that writing down the translation itself *depends on λS being weakly normalizing*, as the types of continuations need the type of the expression being translated. This is not the case for thunkification translations. One consequence of this fact is that the translation can be applied to non-normalizing expressions to give non-normalizing I-expressions. It is natural to wonder if this is always possible. If λS is not weakly normalizing, is it possible to show that its I-fragment is not strongly normalizing? This may be of independent interested, as it relates to *relevance type systems*.

CHAPTER 3

A DEPENDENCY-ELIMINATING TRANSLATION FOR TIERED PURE TYPE SYSTEMS

This is work I presented in part at the 28th International Conference on Types for Proofs and Programs.

3.1 Introduction

The last few decades have seen quite a few techniques for proving normalization of systems in the λ-cube and their extensions, but many of these techniques have not been generalized to pure type systems. The purpose of the work presented in this chapter is to generalize one such technique which might be call *dependency eliminating translations*. The idea is the following: to show that a system is strongly normalizing, it suffices to define a typability-preserving infinite-reduction-sequence-preserving translation from that system into another system which is already known to be strongly normalizing. Harper et al. [1993] define such a translation from λP to $\lambda_{\rightarrow}$ and Geuvers and Nederhof [1991] extend that translation to one from λC to $\lambda\omega$. Both of these translations can be viewed as removing the *dependent* rule in the corresponding system, *i.e.*, the rule $(*, \Box)$ which allows types to depend on terms. For sufficiently well-structured pure type systems, this notion of dependence can be generalized, as is done by Barthe et al. [2001] for their definition of generalized non-dependent pure type systems (which is similar the definition of logical non-dependent pure type systems given by Coquand and Herbelin [1994]). I extend the above mentioned translations to pure type systems in a way that maintains this dependent rule elimination property. The translation can be applied to some weak sub-systems of ECC^n (defined by Luo [1990]) and can be combined with the CPS translation of Barthe et al. [2001] to provide evidence for the strong form of the Barendregt-Geuvers-Klop conjecture, which asks if the proof of strong

normalization from weak normalization can be carried out in Peano arithmetic or even Heyting arithmetic. It has the benefit of being applicable to *non-normalizing* systems like λU, which implies non-normalization of λU can be reduced to non-normalization of λU extended with dependent rules, a setting in which constructions like Σ-types can be encoded.

In what follows, I present a few preliminary notions, and then I define the translation, which is given in two parts for reasons described in the corresponding section. After this is a short section on the applications of this translation.

3.2 The Translation

Recall that a tiered pure type system λS is **non-dependent** if its rules are non-dependent, *i.e.*, if $(s, s') \in \mathcal{R}_{\lambda S}$ implies $s \geq_{\mathcal{A}_{\lambda S}} s'$. The translation defined in the subsequent section eliminates the dependent part of a given tiered pure type system. This motivates the following definition.

Definition 35. *The **non-dependent restriction** of a tiered pure type system λS, denoted here as λS^*, is the system specified by*

$$\mathcal{S}_{\lambda S^*} \triangleq \mathcal{S}_{\lambda S}$$
$$\mathcal{A}_{\lambda S^*} \triangleq \mathcal{A}_{\lambda S}$$
$$\mathcal{R}_{\lambda S^*} \triangleq \{(s, s', s') \in \mathcal{R}_{\lambda S} \mid (s \geq s')\}$$

For all other technical notions, I refer because to the section on definitions in the introductory chapter (Section 1.1).

I present a translation from a pure type system λS with sufficiently rich non-dependent structure to its non-dependent restriction λS^* which preserves typability and infinite reduction sequences. The requirements on the non-dependent structure will be quite strong, but I believe the translation to be a fairly faithful generalization of Geuvers-Nederhof translation.

At a high level, the translation eliminates dependent uses of the product type formation rule, but because it also has to preserve infinite reduction paths, it can't delete too much sub-expression information; dropping sub-expressions might eliminate redexes that account for reduction paths in pre-translated term. The rough idea is to translate all sorts down to the lowest sort s_1, so that when there is a term M of type $\Pi x^A.\ B$, the translated form of B, denoted for now as $\rho(B)$, is of sort s_1, which ensures that any product type formations with $\rho(B)$ as the target type are necessarily non-dependent (there are no sorts lower than s_1). The first issue is that translated terms must be typeable by these translated types, which indicates that there need to be two separate translations, one for types and one for terms. The second issue is that types may contain terms (*e.g.*, $(\lambda x^{s_i}.\ x)A$ for some A where $\Gamma \vdash A : s_i$) which indicates the need for separate translation for these terms as well. This process also needs to be iterated up to the top-sort, for which there are no subterms of this form by the top-sort lemma (Lemma 7).

Thus, the translation is defined in two steps. First, we define a sequence of translations that culminate in a translation ρ_0 for a types. Each translation maps sorts to a lower sort then the last, all the way down to a *type* 0 (a distinguished variable) of sort s_1. Then we define a translation γ for terms so that there is a context Γ^* where $\Gamma \vdash_{\lambda S} M : A$ implies $\Gamma^* \vdash_{\lambda S^*} \gamma(()M) : \rho_0(A)$.

The strong requirement on the non-dependent part of λS comes from the fact that each subsequent translation includes a new variable in the context, and as more variables are included in the context, more variables need to be abstracted over so as not to lose sub-expression information. This motivates the following definition.

Definition 36. *A tiered pure type system is (i,j)-**full** if its rules contain $\{(s_l, s_k) \mid l \leq i$ and $l \leq k \leq j\}$ and is **full** if it satisfies the following closure property: if $(s_i, s_j) \in \mathcal{R}_{\lambda S}$ then λS is (j,i)-full.*

We can already see how restrictive this definition is. A system which is (i,j)-full for $i \geq 2$

and $j \geq 3$ contains λU, and so is inconsistent. The strongest consistent full n-tiered system is of the has the rules

$$\{(s_k, s_1) \mid k \in [n]\} \cup \{(s_1, s_k) \mid k \in [n]\} \cup \{(s_2, s_k) \mid k \in [n]\}$$

which is a subsystem of ECC^n.

For the remainder of this section, fix a **full n-tiered pure type system λS.**

3.2.1 The Type-Level Translation

The type-level translation is the last of $n+1$ inductively defined translations. Each previously defined translation is used to prove that the subsequently defined translations preserve typability. In standard abuse of notation, sequences of Πs, λs, or applications may be empty.

Definition 37. *For each i satisfying $0 \leq i \leq n$, define the function $\rho_i : \mathsf{T}_{\geq i+1} \to \mathsf{T}_{i+1}$ inductively as follows.*

$$\rho_i(s_j) \triangleq \begin{cases} 0 & i = 0 \\ s_i & \textit{otherwise} \end{cases}$$

$$\rho_i({}^{s_j}x) \triangleq {}^{s_{i+2}}x \quad \textit{(where } j \geq i+2\textit{)}$$

$$\rho_i(\Pi x^A.\ B) \triangleq \Pi x^{\rho_{\deg A - 1}(A)}. \ldots \Pi x^{\rho_i(A)}.\ \rho_i(B)$$

$$\rho_i(\lambda x^A.\ M) \triangleq \lambda x^{\rho_{\deg A - 1}(A)}. \ldots \lambda x^{\rho_{i+1}(A)}.\ \rho_i(M)$$

$$\rho_i(MN) \triangleq \rho_i(M)\rho_{\deg N - 1}(N) \ldots \rho_i(N)$$

where 0 is a distinguished variable. The definition of each translation is restricted to those cases for which there are terms in the intended domain. For example, the most basic translation $\rho_n : \mathsf{T}_{\geq n+1} \to \mathsf{T}_{n+1}$ is defined only on $\{s_n\}$ with $\rho_n(s_n) = s_n$. All other cases are

ignored. These translations are extended to contexts as follows.

$$\rho_i(\varnothing) \triangleq (0 : s_1)$$
$$\rho_i(\Gamma, {}^{s_j}x : A) \triangleq \rho_i(\Gamma), {}^{s_j}x : \rho_{j-1}(A), \ldots, {}^{s_{i+1}}x : \rho_i(A) \qquad j \geq i+1$$

Note that $\rho_j(\Gamma) \subset \rho_i(\Gamma)$ if $i < j$, a fact which is used frequently in tandem with the thinning lemma (Lemma 4).

The following three lemmas describe the key features of the translation. First, substitution in some sense commutes the translation.

Lemma 42. *(ρ_i commutes with substitution) For all i satisfying $0 \leq i \leq n$, all j satisfying $1 \leq j \leq n$, and all terms A and B such that $A \in \mathsf{T}_{\geq i+1}$ and $\deg B = j-1$,*

$$\rho_i(A[B/^{s_j}x]) = \rho_i(A)[\rho_{j-2}(B)/^{s_j}x] \ldots [\rho_i(B)/^{s_{i+2}}x]$$

Proof. By reverse induction on i. It holds trivially for ρ_n. For fixed i, we proceed by induction on the structure of A.

Sort. Suppose A is of the form s^k where $k \geq i$. Then for any value of j,

$$\begin{aligned}\rho_i(s_k[B/^{s_j}x]) &= \rho_i(s_k) \\ &= \rho_i(s_k)[\rho_{j+2}(B)/^{s_j}x] \ldots [\rho_i(B)/^{s_{i+2}}x]\end{aligned}$$

since s_k has no free variables and 0 is distinct from all other variables.

Variable. Suppose A is of the form ${}^{s_k}y$ where $k \geq i+2$. If $j < i+2$, then

$$\rho_i({}^{s_k}y[B/^{s_j}x]) = \rho_i({}^{s_k}y)$$

If $j \geq i+2$ and ${}^{s_k}y = {}^{s_j}x$ then

$$\begin{aligned}\rho_i({}^{s_j}x[B/{}^{s_j}x]) &= \rho_i(B)\\ &= {}^{s_{i+2}}x[\rho_i(B)/{}^{s_{i+2}}x]\\ &= \rho_i({}^{s_j}x)[\rho_{j-2}(B)/{}^{s_j}x]\ldots[\rho_i(B)/{}^{s_{i+2}}(x)]\end{aligned}$$

If $j \geq i+2$ and ${}^{s_k}y \neq {}^{s_j}x$ then

$$\begin{aligned}\rho_i({}^{s_k}y[B/{}^{s_j}x]) &= \rho_i({}^{s_k}y)\\ &= \rho_i({}^{s_k}y)[\rho_{j-2}(B)/{}^{s_j}x]\ldots[\rho_i(B)/{}^{s_{i+2}}(x)]\end{aligned}$$

Π-Expression. Suppose A is of the form $\Pi y^C.\ D$. If $\deg C < i+1$, then

$$\begin{aligned}\rho_i((\Pi y^C.\ D)[B/{}^{s_j}x]) &= \rho_i(\Pi y^{C[B/{}^{s_j}x]}.\ D[B/{}^{s_j}x])\\ &= \rho_i(D[B/{}^{s_j}x])\end{aligned}$$

If $j < i+2$ then by the inductive hypothesis

$$\rho_i(D[B/{}^{s_j}x]) = \rho_i(D) = \rho_i(\Pi x^C.\ D)$$

and likewise if $j \geq i+2$ then

$$\begin{aligned}\rho_i(D[B/{}^{s_j}x]) &= \rho_i(D)[\rho_{j-2}(B)/{}^{s_j}x]\ldots[\rho_i(B)/{}^{s_{i+2}}x]\\ &= \rho_i(\Pi x^C.\ D)[\rho_{j-2}(B)/{}^{s_j}x]\ldots[\rho_i(B)/{}^{s_{i+2}}x]\end{aligned}$$

If $\deg C \geq i+1$ then

$$\begin{aligned}
&\rho_i((\Pi y^C.\ D)[B/^{s_j}x]) \\
&= \rho_i(\Pi y^{C[B/^{s_j}x]}.\ D[B/^{s_j}x]) \\
&= \Pi y^{\rho_{\deg C-1}(C[B/^{s_j}x])}. \ldots \Pi y^{\rho_i(C[B/^{s_j}x])}.\ \rho_i(D[B/^{s_j}x])
\end{aligned}$$

If $j < i+2$ then by the inductive hypothesis

$$\begin{aligned}
&\rho_i(\Pi y^{C[B/^{s_j}x]}.\ D[B/^{s_j}x]) \\
&= \Pi y^{\rho_{\deg C-1}(C[B/^{s_j}x])}. \ldots \Pi y^{\rho_i(C[B/^{s_j}x])}.\ \rho_i(D[B/^{s_j}x]) \\
&= \Pi y^{\rho_{\deg C-1}(C)}. \ldots \Pi y^{\rho_i(C)}.\ \rho_i(D) \\
&= \rho_i(\Pi y^C.\ D)
\end{aligned}$$

and likewise if $j \geq i+2$

$$\begin{aligned}
&\rho_i(\Pi y^{C[B/^{s_j}x]}.\ D[B/^{s_j}x]) \\
&= \Pi y^{\rho_{\deg C-1}(C[B/^{s_j}x])}. \ldots \Pi y^{\rho_i(C[B/^{s_j}x])}.\ \rho_i(D[B/^{s_j}x]) \\
&= \Pi y^{\rho_{\deg C-1}(C)[\rho_{j-2}(B)/^{s_j}x]\ldots[\rho_i(B)/^{s_{i+2}}x]} \\
&\qquad \ldots \Pi y^{\rho_i(C)[\rho_{j-2}(B)/^{s_j}x]\ldots[\rho_i(B)/^{s_{i+2}}x]} \\
&\qquad (\rho_i(D)[\rho_{j-2}(B)/^{s_j}x] \ldots [\rho_i(B)/^{s_{i+2}}x]) \\
&= (\Pi y^{\rho_{\deg C-1}(C[B/^{s_j}x])}. \ldots \Pi y^{\rho_i(C[B/^{s_j}x])}.\ \rho_i(D[B/^{s_j}x])) \\
&\qquad [\rho_{j-2}(B)/^{s_j}x] \ldots [\rho_i(B)/^{s_{i+2}}x] \\
&= \rho_i(\Pi y^C.\ D)[\rho_{j-2}(B)/^{s_j}x] \ldots [\rho_i(B)/^{s_{i+2}}x]
\end{aligned}$$

Note that here we are implicitly using the fact that if $\rho_k(C[B/^{s_j}x]) = \rho_k(C)$ then

$$\rho_k(C) = \rho_k(C)[\rho_{j-2}(B)/^{s_j}x] \dots [\rho_i(B)/^{s_{i+2}}x]$$

λ-Expression.. Suppose that A is of the form $\lambda x^C.\ D$. If $\deg C < i+2$ then

$$\begin{aligned}\rho_i((\lambda x^C.\ D)[B/^{s_j}x]) &= \rho_i(\lambda x^{C[B/^{s_j}x]}.\ D[B/^{s_j}x]) \\ &= \rho_i(D[B/^{s_j}x])\end{aligned}$$

and if $\deg C \geq i+2$ then

$$\begin{aligned}&\rho_i(\lambda x^{C[B/^{s_j}x]}.\ D[B/^{s_j}x]) \\ &= \lambda x^{\rho_{\deg C-1}(C[B/^{s_j}x])}.\ \dots \lambda x^{\rho_{i+1}(C[B/^{s_j}x])}.\ \rho_i(D[B/^{s_j}x])\end{aligned}$$

The remainder of the proof of this case is similar to the one for Π-terms.

Application. Suppose A is of the form MN. If $\deg N < i+1$, then

$$\begin{aligned}\rho_i((MN)[B/^{s_j}x]) &= \rho_i((M[B/^{s_j}x])(N[B/^{s_j}x])) \\ &= \rho_i(M[B/^{s_j}x])\end{aligned}$$

and if $\deg N \geq i+1$, then

$$\begin{aligned}&\rho_i((M[B/^{s_j}x])(N[B/^{s_j}x])) \\ &= \rho_i(M[B/^{s_j}x])\rho_{\deg N-1}(N[B/^{s_j}x]) \dots \rho_i(N[B/^{s_j}x])\end{aligned}$$

The remainder of the proof of this case is similar to the one for Π-terms. □

The next lemma says that the translation preserves β-reductions. This is necessary for

translating conversion rules; we need to be able to replace β-equivalent types after translation. Note that the lemma does not imply the translation preserves infinite reduction paths. This is because sub-expression information is lost in the translation, *e.g.*, in the case of application it may be that $\rho_i(MN) = \rho_i(M)$ and there would be no hope in preserving an infinite reduction sequence on the sub-expression N.

Lemma 43. *For all i satisfying $0 \leq i \leq n$, and all terms A and B such that $A \in \mathsf{T}_{\geq i+1}$ and $A \twoheadrightarrow_\beta B$, it follows that $\rho_i(A) \twoheadrightarrow_\beta \rho_i(B)$.*

Proof. By reverse induction on i. This holds trivially for ρ_n. For fixed i, we proceed by induction on the definition of the multi-step β-reduction relation. In the case of the reducts:

$$\rho_i\left((\lambda x^A.\ M)N\right) \to_\beta \rho_i\left(M[N/^{s_{\deg A}}x]\right)$$

if $\deg N < i+1$ (and $\deg A < i+2$) then

$$\begin{aligned}\rho_i\left((\lambda x^A.\ M)N\right) &= \rho_i\left(\lambda x^A.\ M\right)\\ &= \rho_i(M)\\ &= \rho_i(M[^{s_{\deg A}}x/N])\end{aligned}$$

where the last equality follows from Lemma 42, and if $\deg N \geq i+1$ (and $\deg A \geq i+2$) then

$$\begin{aligned}\rho_i\left((\lambda x^A.\ M)N\right) &= \rho_i(\lambda x^A.\ M)\rho_{\deg N-1}(N)\ldots\rho_i(N)\\ &= \left(\lambda x^{\rho_{\deg A-1}(A)}.\ \ldots\lambda x^{\rho_{i+1}(A)}.\ \rho_i(M)\right)\rho_{\deg N-1}(N)\ldots\rho_i(N)\\ &\twoheadrightarrow_\beta \rho_i(M)[\rho_{\deg N-1}(N)/^{s_{\deg A}}x]\ldots[\rho_i(N)/^{s_{i+2}}x]\end{aligned}$$

Note that here we implicitly use the fact that ${}^{s_{\deg A}}x$ does not appear in A, and so

$$\rho_k(A)[\rho_l(N)/{}^{s_{l+2}}x] = \rho_k(A)$$

where $i+1 \leq k \leq \deg A - 1$ and $i \leq l \leq \deg N - 1$ so each of the subsequent β-reductions above do not change the type annotations in the associated λ-terms.

It is straightforward to verify that the translation preserves β-reductions with respect to compatibility. For example, suppose A is of the form $\Pi x^C.\ D$ and B is of the form $\Pi x^{C'}.\ D$ where $C \twoheadrightarrow_\beta C'$. If $\deg C < i+1$ then

$$\rho_i(\Pi x^C.\ D) = \rho_i(D) = \rho_i(\Pi x^{C'}.\ D)$$

If $\deg C \geq i+1$ then

$$\begin{aligned}\rho_i(\Pi x^C.\ D) &= \Pi x^{\rho_{\deg C-1}(C)}. \ldots \Pi x^{\rho_i(C)}.\ \rho_i(D)\\ &\twoheadrightarrow_\beta \Pi x^{\rho_{\deg C-1}(C')}. \ldots \Pi x^{\rho_i(C')}.\ \rho_i(D)\\ &= \rho_i(\Pi x^{C'}.\ D)\end{aligned}$$

where $\rho_i(C) \twoheadrightarrow_\beta \rho_i(C')$ by the induction on the multi-step β-reduction relation and $\rho_k(C) \twoheadrightarrow_\beta \rho_k(C')$ for $k > i$ by the induction on i. Note also that we implicitly use the fact that $\deg C = \deg C'$ (Lemma 10). □

Finally, it must be that the translation preserves typability. This ensures the translation gives well-defined types when used in combination with the term translation in the next section. Recall that λS^* is the non-dependent restriction of λS.

Lemma 44. *For all i satisfying $0 \leq i \leq n$ the following holds. Let Γ be a context and let A*

and B be terms such that $A \in \mathsf{T}_{\geq i+1}$ and $\Gamma \vdash_{\lambda S} A : B$. Then

$$\rho_{i+1}(\Gamma) \vdash_{\lambda S^*} \rho_i(A) : \rho_{i+1}(B)$$

Proof. By reverse induction on i. This holds trivially for ρ_n. For fixed i, it follows by induction on derivations.

Axiom. Suppose the derivation is of the form

$$\vdash_{\lambda S} s_j : s_{j+1}$$

where $j \geq i$. If $i \geq 1$ then the translated derivation is

$$0 : s_1 \vdash_{\lambda S^*} s_i : s_{i+1}$$

and otherwise

$$0 : s_1 \vdash_{\lambda S^*} 0 : s_1$$

This judgment is derivable because any every full system contains the rule (s_1, s_2).

Variable Introduction. Suppose the last inference is of the form

$$\frac{\Gamma' \vdash_{\lambda S} B : s_j}{\Gamma', {}^{s_j}x : B \vdash_{\lambda S} {}^{s_j}x : B}$$

where $j \geq i+2$ (so that $\deg({}^{s_j}x) \geq i+1$). By the inductive hypothesis

$$\rho_{k+1}(\Gamma') \vdash_{\lambda S^*} \rho_k(B) : s_{k+1}$$

for each k satisfying $i \leq k \leq j-1$, and since the context $\rho_{i+1}(\Gamma')$ is valid, as it appears in the case with k instantiated as i, we have

$$\rho_{i+1}(\Gamma') \vdash_{\lambda S^*} \rho_k(B) : s_{k+1}$$

for each k by weakening. By further repeated weakening

$$\rho_{i+1}(\Gamma'), {}^{s_j}x : \rho_{j-1}(B), \ldots, {}^{s_{i+3}}x : \rho_{i+2}(B) \vdash_{\lambda S^*} \rho_{i+1}(B) : s_{i+2}$$

and can apply the variable introduction rule:

$$\rho_{i+1}(\Gamma'), {}^{s_j}x : \rho_{j-1}(B), \ldots, {}^{s_{i+2}}x : \rho_{i+1}(B), \vdash_{\lambda S^*} {}^{s_{i+2}}x : \rho_{i+1}(B)$$

Weakening. Suppose the last inference is of the form

$$\frac{\Gamma' \vdash_{\lambda S} A : B \qquad \Gamma' \vdash_{\lambda S} C : s_j}{\Gamma', {}^{s_j}x : C \vdash_{\lambda S} A : B}$$

If $j < i + 2$, then the corresponding inference is

$$\rho_{i+1}(\Gamma') \vdash_{\lambda S^*} \rho_i(A) : \rho_{i+1}(B)$$

which is obtained by applying the inductive hypothesis to the left antecedent judgment. Note that the variable ${}^{s_j}x$ is dropped by the translation ρ_{i+1}. If $j \geq i + 2$, we use a similar procedure as the one for the *Variable introduction* step above, *i.e.*

$$\rho_{i+1}(\Gamma') \vdash_{\lambda S^*} \rho_k(C) : s_{k+1}$$

for each k satisfying $i \leq k \leq j - 1$, and then by repeated weakening

$$\rho_{i+1}(\Gamma'), {}^{s_j}x : \rho_{j-1}(C), \ldots, {}^{s_{i+2}}x : \rho_{i+1}(C) \vdash_{\lambda S^*} \rho_i(A) : \rho_{i+1}(B)$$

Product Type Formation. Suppose the last inference is of the form

$$\frac{\Gamma \vdash_{\lambda S} C : s_j \qquad \Gamma, {}^{s_j}x : C \vdash_{\lambda S} D : s_k}{\Gamma \vdash_{\lambda S} \Pi x^C.\ D : s_k}$$

where $k \geq i+1$. If $j < i+1$ then

$$\rho_{i+1}(\Gamma) \vdash_{\lambda S^*} \rho_i(D) : s_{i+1}$$

which is obtained by applying the inductive hypothesis to the right antecedent judgment. If $j \geq i+1$, then we can derive

$$\rho_{k+1}(\Gamma) \vdash_{\lambda S^*} \rho_k(C) : s_{k+1}$$

for each k satisfying $i \leq k \leq j-1$, and each of these judgments can be weakened as

$$\rho_{i+1}(\Gamma), {}^{s_j}x : \rho_{j-1}(C), \ldots, {}^{s_{k+2}}x : \rho_{k+1}(C) \vdash_{\lambda S^*} \rho_k(C) : s_{k+1}$$

Then with repeated uses of the product formation rule together with

$$\rho_{i+1}(\Gamma), {}^{s_j}x : \rho_{j-1}(C), \ldots, {}^{s_{i+2}}x : \rho_{i+1}(C) \vdash_{\lambda S^*} \rho_i(D) : s_{i+1}$$

we can derive

$$\rho_{i+1}(\Gamma) \vdash_{\lambda S^*} \Pi x^{\rho_{j-1}(C)}. \ldots \Pi x^{\rho_{i+1}(C)}. (\rho_i(C) \to \rho_i(D)) : s_{i+1}$$

Here we use the assumption that λS^* has the rules (s_k, s_{i+1}, s_{i+1}) for each k satisfying $k \geq i+1$. Also note the use of function notation since $\rho_i(C)$ does not appear in the context above.

Abstraction.. Suppose the last inference is of the form

$$\frac{\Gamma, {}^{s_j}x : C \vdash_{\lambda S} M : D \qquad \Gamma \vdash_{\lambda S} \Pi x^C.\ D : s_k}{\Gamma \vdash_{\lambda S} \lambda x^C.\ M : \Pi x^C.\ D}$$

where $k \geq i+2$. If $\deg C < i+2$ (*i.e.*, if $j < i+2$), then the corresponding inference is

$$\rho_{i+1}(\Gamma) \vdash_{\lambda S^*} \rho_i(M) : \rho_{i+1}(B)$$

since the variable ${}^{s_j}x$ is dropped by the translation ρ_{i+1}. If $\deg C \geq i+2$ then we can derive

$$\rho_{i+1}(\Gamma), {}^{s_j}x : \rho_{j-1}(C), \ldots, {}^{s_{i+2}}x : \rho_{i+1}(C) \vdash_{\lambda S^*} \rho_i(M) : \rho_{i+1}(D)$$

and

$$\rho_{i+1}(\Gamma) \vdash_{\lambda S^*} \Pi x^{\rho_{j-1}(C)}. \ldots \Pi x^{\rho_{i+1}(C)}. \rho_{i+1}(D) : s_{i+2}$$

And so by repeated use of the generation lemma (Lemma 1), if $\deg C \geq i+3$, we can also derive

$$\Pi x^{\rho_k(C)}. \ldots \Pi x^{\rho_{i+1}(C)}. \rho_{i+1}(D) : s_{i+2}$$

in λS^* under the context

$$\rho_{i+1}(\Gamma), {}^{s_j}x : \rho_{j-1}(C), \ldots, {}^{s_{k+2}}x : \rho_{k+1}(C)$$

for each k satisfying $i+1 \leq k \leq j-2$. And so by repeated use of the abstraction rule, we can derive

$$\lambda x^{\rho_{j-1}(C)}. \ldots \lambda x^{\rho_{i+1}(C)}. \rho_i(M) : \Pi x^{\rho_{j-1}(C)}. \ldots \Pi x^{\rho_{i+1}(C)}. \rho_{i+1}(D)$$

in λS^* under the context $\rho_{i+1}(\Gamma)$.

Application. Suppose the last inference is of the form Since $\deg(MN) = \deg M$, we can apply the inductive hypothesis to the left antecedent judgment:

$$\rho_{i+1}(\Gamma) \vdash_{\lambda S^*} \rho_i(M) : \rho_{i+1}(\Pi x^C. D)$$

If $\deg N < i+1$ (and $\deg C < i+2$) then

$$\rho_{i+1}(\Gamma) \vdash_{\lambda S^*} \rho_i(M) : \rho_{i+1}(B)$$

If $\deg N \geq i+1$ (and $\deg C \geq i+2$) then we have

$$\rho_{i+1}(\Gamma) \vdash_{\lambda S^*} \rho_i(M) : \Pi x^{\rho_{\deg C-1}(C)}. \ldots \Pi x^{\rho_{i+1}(C)}. \rho_{i+1}(D)$$

and

$$\rho_{i+1}(\Gamma) \vdash_{\lambda S^*} \rho_k(N) : \rho_{k+1}(C)$$

for each k satisfying $i \leq k \leq \deg N - 1$. The first application $\rho_i(M)\rho_{\deg N-1}(N)$ has the type

$$\begin{aligned}&\Pi x^{\rho_{\deg C-2}(C)[\rho_{\deg C-2}(N)/^{s_{\deg C}}x]} \\ &\quad \ldots \Pi x^{\rho_{i+1}(C)[\rho_{\deg C-2}(N)/^{s_{\deg C}}x]}(\rho_{i+1}(D)[\rho_{\deg N-1}(N)/^{s_{\deg C}}x])\end{aligned}$$

The term $\rho_{\deg C-2}(C)$ does not have the variable ${}^{s_{\deg C}}x$, so this type simplifies to

$$\Pi x^{\rho_{\deg C-2}(C)}. \ldots \Pi x^{\rho_{i+1}(C)}. (\rho_{i+1}(D)[\rho_{\deg N-1}(N)/^{s_{\deg C}}x])$$

repeating this process, we have

$$\begin{aligned}&\rho_{i+1}(\Gamma) \vdash_{\lambda S^*} \rho_i(M)\rho_{j-1}(N) \ldots \rho_i(N) \\ &\quad : \rho_{i+1}(D)[\rho_{\deg N-1}(N)/^{s_{\deg C}}x] \ldots [\rho_i(N)/^{s_{i+2}}x]\end{aligned}$$

Since ${}^{s_{i+2}}x$ does not appear in $\rho_{i+1}(D)$, the type is

$$\rho_{i+1}(D)[\rho_{\deg N-1}(N)/^{s_{\deg C}}x] \ldots [\rho_{i+1}(N)/^{s_{i+3}}x]$$

which is the same as $\rho_{i+1}(B[N/x])$ by Lemma 42.

Conversion. Suppose the last inference is of the form

$$\frac{\Gamma \vdash_{\lambda S} A : C \qquad \Gamma \vdash_{\lambda S} B : s_k}{\Gamma \vdash_{\lambda S} A : B}$$

where $k \geq i+2$. Then the corresponding derivation is

$$\frac{\rho_{i+1}(\Gamma) \vdash_{\lambda S^*} \rho_i(A) : \rho_{i+1}(C) \qquad \dfrac{\rho_{i+2}(\Gamma) \vdash_{\lambda S^*} \rho_{i+1}(B) : s_{i+2}}{\rho_{i+1}(\Gamma) \vdash_{\lambda S^*} \rho_{i+1}(B) : s_{i+2}}}{\rho_{i+1}\Gamma \vdash_{\lambda S^*} \rho_i(A) : \rho_{i+1}(B)}$$

where $\rho_{i+1}(B) =_\beta \rho_{i+1}(C)$ by Lemma 43. □

3.2.2 The Term-Level Translation

I next define the translation $\gamma : \mathsf{T} \to \mathsf{T}_0$ on all expressions. This requires one of the greater departures from the Geuvers-Nederhof translation. In their translation, variables are occasionally substituted with dummy expression via $\bot$-terms that are added to the context. This is possible since $(s_2, s_1) \in \mathcal{R}_{\lambda\omega}$. Using this technique directly would require the rules (s_{i+1}, s_i) for $i \in [n-1]$. However, by fullness, $(s_j, s_k) \in \mathcal{R}_{\lambda S}$ implies $(s_l, s_1) \in \mathcal{R}_{\lambda S}$ for $l \in [k]$, which will allow us a product encoding variable $\mathsf{prod} : \Pi A^{s_1}.\ 0 \to A \to 0$, which only requires the rule (s_2, s_1), which we assume appear in λS.

Definition 38. *Define the function* $\gamma : \mathsf{T} \to \mathsf{T}$ *inductively as follows.*

$$\begin{aligned}
\gamma(s_i) &\triangleq \bullet \\
\gamma({}^{s_i}x) &\triangleq {}^{s_1}x \\
\gamma(\Pi x^A.\ B) &\triangleq \mathsf{prod}(\Pi x^{\rho_{\deg A-1}(A)}.\ \ldots \Pi x^{\rho_0(A)}.\ 0)\gamma(A) \\
&\qquad (\lambda x^{\rho_{\deg A-1}(A)}.\ \ldots \lambda x^{\rho_0(A)}.\ \gamma(B)) \\
\gamma(\lambda x^A.\ M) &\triangleq (\lambda z^0.\ \lambda x^{\rho_{\deg A-1}(A)}.\ \ldots \lambda x^{\rho_0(A)}.\ \gamma(M))\gamma(A) \\
\gamma(MN) &\triangleq \gamma(M)\rho_{\deg N-1}(N)\ldots\rho_0(N)\gamma(N)
\end{aligned}$$

where prod, $\bullet$ *and* z *are distinguished variables.*

We prove the same three lemmas as the previous section: typability preservation, substitution commutation, and β-preservation.

Lemma 45. *For terms* A *and* B, *if* $\Gamma \vdash_{\lambda S} A : B$, *then*

$$0 : s_1, \bullet : 0, \mathsf{prod} : \Pi A^{s_1}.\ 0 \to A \to 0, \rho_0(\Gamma) \vdash_{\lambda S^*} \gamma(A) : \rho_0(B)$$

Proof. By induction on derivations. The cases in which the last derivation is a variable introduction or weakening are similar to the analogous cases for Lemma 44. In what follows, let Δ denote the context $(0 : s_1, \bullet : 0, \mathsf{prod} : \Pi A^{s_1}.\ 0 \to A \to 0)$.

Axioms. If the derivation is of the form

$$\vdash s_i : s_{i+1}$$

then translated derivation is

$$\Delta \vdash \bullet : 0$$

In particular, the context is well-formed.

Product type formation. Suppose the last inference is of the form

$$\frac{\Gamma \vdash C : s_i \qquad \Gamma, {}^{s_i}x : C \vdash D : s_j}{\Gamma \vdash \Pi x^C.\ D : s_j}$$

By the inductive hypothesis

$$\Delta, \rho_0(\Gamma) \vdash \gamma(C) : 0$$

and

$$\Delta, \rho_0(\Gamma), {}^{s_i}x : \rho_{i-1}(C), \ldots, {}^{s_1}x : \rho_0(C) \vdash \gamma(D) : 0$$

By weakening, we can derive

$$\Delta, \rho_0(\Gamma), {}^{s_i}x : \rho_{i-1}(C), \ldots, {}^{s_1}x : \rho_0(C) \vdash 0 : s_1$$

Note that, by fullness, since $(s_i, s_j) \in \mathcal{R}_{\lambda S}$, it must be that $(s_k, s_1) \in \mathcal{R}_{\lambda S}$ for $k \in [i]$, and so by repeated product type formation all with the previously defined translations

$$\rho_{k+1}(\Gamma) \vdash \rho_k(C) : s_{k+1}$$

we can derive

$$\Delta, \rho_0(\Gamma), {}^{s_i}x : \rho_{i-1}(C), \ldots, {}^{s_{k+1}}x : \rho_k(C) \vdash \Pi^{s_k}x^{\rho_{k-1}(C)}. \ldots \Pi^{s_1}x^{\rho_0(C)}.\ 0 : s_1$$

where $1 \leq k \leq i$. These may be used for each abstraction step to define

$$\Delta, \rho_0(\Gamma) \vdash \lambda^{s_i}x^{\rho_{i-1}(C)}.\ \ldots \lambda^{s_1}x^{\rho_0(C)}.\ \gamma(D) : 0$$

Finally by two applications using the previously defined derivations, we have

$$\Delta, \rho_0(\Gamma)$$
$$\vdash \mathsf{prod}(\Pi^{s_i} x^{\rho_{i-1}(C)}. \ldots \Pi^{s_1} x^{\rho_0(C)}.\ 0)\gamma(A)(\lambda^{s_i} x^{\rho_{i-1}(C)}.\ \ldots \lambda^{s_1} x^{\rho_0(C)}.\ \gamma(D))0$$

Abstraction. Suppose the last inference is of the form

$$\frac{\Gamma, {}^{s_i}x : C \vdash M : D \qquad \Gamma \vdash \Pi x^C.\ D : s_j}{\Gamma \vdash \lambda x^C.\ M : \Pi x^C.\ D}$$

Similar to the same case in the proof of Lemma 44, we can derive

$$\rho_0(\Gamma) \vdash \lambda x^{\rho_{i-1}(C)}.\ \ldots \lambda x^{\rho_0(C)}.\ \gamma(D) : \Pi x^{\rho_{i-1}(C)}. \ldots \Pi x^{\rho_0(C)}.\ \rho_0(D)$$

and by weakening,

$$\rho_0(\Gamma), z : 0 \vdash \lambda x^{\rho_{i-1}(C)}.\ \ldots \lambda x^{\rho_0(C)}.\ \gamma(D) : \Pi x^{\rho_{i-1}(C)}. \ldots \Pi x^{\rho_0(C)}.\ \rho_0(D)$$

for a fresh variable z. Furthermore, we have

$$\frac{\rho_0(\Gamma) \vdash \Pi x^{\rho_{i-1}(C)}. \ldots \Pi x^{\rho_0(C)}.\ \rho_0(D) : s_1 \qquad \rho_0(\Gamma) \vdash 0 : s_1}{\rho_0(\Gamma) \vdash_{\lambda S^*} 0 \to \Pi x^{\rho_{i-1}(C)}. \ldots \Pi x^{\rho_0(C)}.\ \rho_0(D) : s_1}$$

so by abstraction and application,

$$\rho_0(\Gamma) \vdash (\lambda z^0.\ \lambda x^{\rho_{i-1}(C)}.\ \ldots \lambda x^{\rho_0(C)}.\ \gamma(D))\gamma(C) : \Pi x^{\rho_{i-1}(C)}. \ldots \Pi x^{\rho_0(C)}.\ \rho_0(D)$$

Application. Suppose the last inference is of the form

$$\frac{\Gamma \vdash M : \Pi x^C.\ D \qquad \Gamma \vdash N : C}{\Gamma \vdash MN : D[N/^{s_{\deg C}} x]}$$

If $\deg N = 0$, then we have

$$\frac{\rho_0(\Gamma) \vdash \gamma(M) : \rho_0(C) \to \rho_0(D) \qquad \rho_0(\Gamma) \vdash \gamma(N) : \tau(C)}{\rho_0(\Gamma) \vdash \gamma(M)\gamma(N) : \rho_0(D)}$$

Note that $\rho_0(D[N/^{s_1}x]) = \rho_0(D)$ by Lemma 42. The case in which $\deg N \geq 1$ is similar to the same case in the proof of Lemma 44.

Conversion. Suppose the last inference is of the form

$$\frac{\Gamma \vdash A : C \qquad \Gamma \vdash B : s_i}{\Gamma \vdash A : B}$$

It follows from Lemma 43 that $\rho_0(C) =_\beta \rho_0(B)$ so we have

$$\frac{\rho_0(\Gamma) \vdash \gamma(A) : \rho_0(C) \qquad \dfrac{\rho_1(\Gamma) \vdash \rho_0(B) : s_1}{\rho_0(\Gamma) \vdash \rho_0(B) : s_1}}{\rho_0(\Gamma) \vdash \gamma(A) : \rho_0(B)}$$

□

Next, we again show that substitution commutes with the translation. In this case, the role of this result is more auxiliary; it is necessary for the next lemma on β-reduction preservation.

Lemma 46. *For all j satisfying $1 \leq j \leq n$, and all terms A and B such that $\deg B = j - 1$,*

$$\gamma(A[B/^{s_j}x]) = \gamma(A)[\rho_{j-2}(B)/^{s_j}x] \ldots [\rho_0(B)/^{s_2}x][\gamma(B)/^{s_1}x]$$

Proof. By induction on the structure of A. The cases in which A is a sort or a variable are straightforward.

Π-Expressions. Suppose A is of the form $\Pi y^C.\ D$. Borrowing notation from Barendregt, in what follows, let M^+ denote $M[B/^{s_j}x]$ for any term M. Then

$$\begin{aligned}
&\gamma((\Pi y^C.\ D)^+) \\
&= \gamma(\Pi y^{C^+}.\ D^+) \\
&= \mathsf{prod}(\Pi^{s_{\deg(C^+)}} x^{\rho_{\deg(C^+)-1}(C^+)}. \ldots \Pi^{s_1} x^{\rho_0(C^+)}.\ 0)\gamma(A) \\
&\quad (\lambda^{s_i} x^{\rho_{\deg(C^+)-1}(C^+)}. \ldots \lambda^{s_1} x^{\rho_0(C^+)}.\ \gamma(D^+))
\end{aligned}$$

Since all substitutions commute, either by the inductive hypothesis, or by Lemma 42, the desired equality follows.

The case of λ-expressions and applications are similar. □

Lemma 47. *For all terms A and B, if $A \to_\beta B$, then $\gamma(A) \twoheadrightarrow_\beta^+ \gamma(B)$.*

Proof. By induction on the definition of the multi-step β-reduction relation. For the case of reducts of the form

$$(\lambda x^A.\ M)N \to_\beta M[N/^{s_{\deg A}} x]$$

If $\deg N = 0$ (and $\deg A = 1$), then

$$\begin{aligned}
\gamma((\lambda x^A.\ M)N) &= \gamma(\lambda x^A.\ M)\gamma(N) \\
&= (\lambda z^0.\ \lambda x^{\rho_0(A)}.\ \gamma(M))\gamma(A)\gamma(N) \\
&\to_\beta (\lambda x^{\rho_0(A)}.\ \gamma(M))\gamma(N) \\
&\to_\beta \gamma(M)[\gamma(N)/^{s_1} x] \\
&= \gamma(M[N/^{s_1} x])
\end{aligned}$$

If $\deg N \geq 1$, then

$$\begin{aligned}
\gamma((\lambda x^A.\ M)N) &= \gamma(\lambda x^A.\ M)\, \rho_{\deg N - 1}(N) \dots \rho_0(N)\gamma(N) \\
&= (\lambda z^0.\ \lambda x^{\rho_{\deg A - 1}(A)}.\ \dots \lambda x^{\rho_0(A)}.\ \gamma(M))\gamma(A)\, \rho_{\deg N - 1}(N) \dots \rho_0(N)\gamma(N) \\
&\to_\beta (\lambda x^{\rho_{\deg A - 1}(A)}.\ \dots \lambda x^{\rho_0(A)}.\ \gamma(M))\, \rho_{\deg N - 1}(N) \dots \rho_0(N)\gamma(N) \\
&\twoheadrightarrow_\beta \gamma(M)[\rho_{\deg N - 1}(N)/^{s_{\deg A}} x] \dots [\rho_0(N)/^{s_2} x][\gamma(N)/^{s_1} x] \\
&= \gamma(M[N/^{s_{\deg A}} x])
\end{aligned}$$

As in the proof of Lemma 43, verifying that the translation preserves β-reductions with respect to compatibility is straightforward. The key observation is that γ appears applied

to every sub-expression of the pre-translated expression, so compatibility preserves *non-zero* β-reductions. □

3.2.3 The Main Result

Theorem 3. *Let λS be an n-tiered pure type system that is full up to some i (where $i < n$). Then λS is strongly normalizing if and only if λS^* is strongly normalizing.*

Proof. Since all derivable terms of λS^* are also derivable in λS, if λS is strongly normalizing then so is λS^*. So suppose that λS is not strongly normalizing. Then there is an infinite reduction sequence

$$M_1 \to_\beta M_2 \to_\beta \dots$$

containing terms derivable in λS. By Lemma 45 and Lemma 47,

$$\gamma(M_1) \twoheadrightarrow_\beta^+ \gamma(M_2) \twoheadrightarrow_\beta^+ \dots$$

is an infinite sequence of terms derivable in λS^*, so λS^* is not strongly normalizing either. □

The only really interesting system to which this theorem can be applied are the n-tiered system with the rules

$$\{(s_k, s_1) \mid k \in [n]\} \cup \{(s_1, s_k) \mid k \in [n]\} \cup \{(s_2, s_k) \mid k \in [n]\}$$

as well as any subsystem of this one with the same non-dependent rules. Theorem 6 says that the strong normalization of this system is equivalent to the one with the rules.

$$\{(s_k, s_1) \mid k \in [n]\} \cup \{(s_2, s_2)\}$$

And though it would be possible to verify formally, a cursory inspection of the technique

indicates that it is meta-theoretically weakly, and can be carried out in Peano arithmetic. So when bootstrapped with the result of Barthe et al. [2001], we get a special case of the strong Barendregt-Geuvers-Klop conjecture.

Corollary 2. *Weak normalization implies strong normalization for the n-tiered pure type system with rules*

$$\{(s_k, s_1) \mid k \in [n]\} \cup \{(s_1, s_k) \mid k \in [n]\} \cup \{(s_2, s_k) \mid k \in [n]\}$$

and, furthermore, this proof can be carried out in Peano Arithmetic.

3.3 Conclusions

In some sense, this result is a failed experiment in generalization. It is unfortunate, though somewhat interesting, that what I believe to be the most natural generalization of the Geuver-Nederhof translation breaks down very early when generalized to higher impredicative systems. All of my attempts to weaken the requirements on the non-dependent part of the underlying system were met by some issue, usually in the form of non-closure under substitution or β-equivalence. It does seem as though it should be possible to make a minor adjustment to the proof to get the same result for the n-tiered systems with rules

$$\{(s_1, s_k) \mid k \in [n]\} \cup \{(s_i, s_j) \mid 1 \leq i \leq j \leq n\}$$

as this would seem to better capture the role of the Geuver-Nederhof result.

CHAPTER 4

AN IRRELEVANCY-ELIMINATING TRANSLATION FOR TIERED PURE TYPE SYSTEMS

This work appears in the Post-Proceedings of the 28th International Conference on Types for Proofs and Programs.

4.1 Introduction

Very little progress has been made on the Barendregt-Geuvers Klop conjecture in part because pure type systems in general are too unstructured to be amenable to standard techniques. Though natural, the generalization to pure type systems from the lambda cube is in some sense the most obvious one, a basic syntactic ambiguation of the inference rules to allow for maximal freedom where the lambda cube imposes its sort-structural restrictions. The resulting systems may fail to have the meta-theoretic properties one might expect (*e.g.*, type unicity). It is, therefore, common to consider classes of pure type systems that maintain these meta-theoretic properties. The state of the art of the conjecture is the result of Barthe *et al.* Barthe et al. [2001], which states that weak normalization implies strong normalization for *generalized non-dependent, clean, negatable*[1] pure type systems. The proof of this result depends on a very nice—though somewhat complicated—generalization of Sørensen's CPS translation Sørensen [1997] for $\lambda\omega$ (among other systems). I have recently given a slightly simpler presentation of the same result Mull [2022] but based on Xi's Thunkification translation Xi [1997].

I propose revisiting the Barendregt-Geuvers-Klop conjecture in a slightly simpler framework. I start by presenting a class of basic, concrete pure type systems I call *tiered* pure type systems. Despite their simplicity they are sufficient to consider with regards to questions

1. These are technical restrictions, discussed further in the later exposition.

about normalization. Stratified persistent systems (and generalized non-dependent systems) are disjoint unions of tiered systems, so tiered systems are in some sense the *atoms* of these systems. For concreteness, the conjecture restricted to this setting is that *weak normalization implies strong normalization in all* ***tiered*** *pure type systems.* The result of Barthe *et al.* implies the conjecture for non-dependent, clean, negatable tiered systems, and so it remains to remove any and all of these restrictions.

This simple reframing of the problem (a moving of the goalposts, so to speak) is a minor though I believe important step towards making further progress on the full version of the conjecture. But even in this setting, there is a huge number of systems to consider, many of which contain what amounts to "junk" structure. The primary contribution of this paper is a translation of pure type systems which preserves typability and infinite reduction paths (I will simply write "path-preserving" from this point forward) and removes this irrelevant structure. By removing structure here, I mean that the target system of the translation is the same as the source system but with some sorts and rules removed.

Consider, for example, the system $\lambda\mathsf{HOL}$, which may be thought of as the system $\lambda\omega$ with an additional *superkind* sort $\triangle$ that allows for the introduction of kind variables that can appear in expressions but cannot be abstracted over. The introduction of $\triangle$ is meaningful with respect to what expressions can be derived, but unsurprisingly both $\lambda\mathsf{HOL}$ and $\lambda\omega$ are strongly normalizing. One basic observation here is that there is a single expression inhabiting $\triangle$, namely the *kind* sort $\Box$. This sparsity of inhabitation can be leveraged to define a path-preserving translation from $\lambda\mathsf{HOL}$ to $\lambda\omega$ and, in fact, from any pure type system with an isolated top-sort to the same system but without the top-sort.

I generalize this basic idea in two ways. First, I define a path-preserving translation that eliminates not just top-sorts but also any sort which is *top-sort-like*. Second, I extend this translation to eliminate not just isolated sorts, but also sorts which may appear as sources of Π-types. In particular, this allows for the elimination of some *dependent* rules in

the system. This translation can be iteratively applied to a system λS until a fixed point $\lambda S^{\downarrow}$ is reached, which is equivalent to λS with respect to strong normalization. Thus, this translation can be used to prove the strong normalization of systems λS for which $\lambda S^{\downarrow}$ is known to be strongly normalizing. But perhaps more importantly, it can be bootstrapped with existing results for the Barendregt-Geuvers-Klop conjecture. The argument is simple: if λS is weakly normalizing, then so is $\lambda S^{\downarrow}$ since it can be embedded in λS. Then $\lambda S^{\downarrow}$ is strongly normalizing by assumption, and so λS is strongly normalizing by the path-preserving translation. Bootstrapping with the result of Barthe *et al.* yields a proof of the Barendregt-Geuvers-Klop conjecture for a larger class of systems.

This technique bears a resemblance to the one used by Roux and van Doorn Roux and Doorn [2014] in their structural theory of pure type systems, which in turn resembles the techniques of Geuvers and Nederhof Geuvers and Nederhof [1991] and Harper *et al.* Harper et al. [1993]. In all these works, a translation is defined from one pure type system into another which has fewer rules. And though it is not explicitly stated, the translation of Roux and van Doorn can be bootstrapped in the same way as described above to push the state of the conjecture. In fact, their translation can be used to eliminate rules *between* disjoint systems in order to, say, make it stratified and persistent (*i.e.*, a disjoint unions of tiered systems) whereas the translation presented here eliminates rules *within* the individual summands in a disjoint union of tiered systems.

It is important to emphasize that this results depends on the fact that the additional structure that can be handled is in some sense irrelevant. But if we do want to prove the full conjecture, we also have to prove it for a class of "junk" systems, ones which may not be interesting in their own right and may have rules which don't add much to the system. In a way, this result is perhaps more meaningfully interpreted in the reverse direction: the systems $\lambda S^{\downarrow}$ for which the conjecture is *not* known to hold are targets for the developments of better techniques. Ideally, some technique could handle all these systems uniformly, but

for now it may be useful to further develop the theory regarding what barriers exist, what systems beyond the lambda cube, natural or not, may be most important to study.

In what follows I present some preliminary material, which includes some exposition on tiered systems. I then define the irrelevancy-eliminating translation in two parts: one part for eliminating rules and one for eliminating sorts. The final translation will be taken as the composition of these two translations. Finally, I present its application to the Barendregt-Geuvers-Klop conjecture and conclude with a short section on what it implies about the systems which remain to be studied.

4.2 The Translation

All relevant preliminary material can be found in the section on definitions in the Introductory Chapter (Section 1.1). Fix an n-tiered pure type system λS. We first describe the sorts which are sufficiently *top-sort-like*, as well as the properties of these sorts that induce irrelevant structure. In what follows, it will be convenient to consider *sets* of top-sort-like sorts. We call a subset $\mathcal{I}$ of $[n]$ an **index set** for λS, and denote by $\mathcal{S}_\mathcal{I}$ the set $\{s_i \mid i \in \mathcal{I}\}$.

Definition 39.

- *A sort s_i is **rule-isolated** if it does not appear in any rules, i.e., for all j, neither (s_j, s_i) nor (s_i, s_j) appear in $\mathcal{R}_{\lambda S}$.*
- *A sort s_i is **top-sort-like** if $i < n$ implies s_{i+1} is rule-isolated (i.e., s_i is a top sort or its succeeding sort is rule-isolated).*
- *For any index set $\mathcal{J}$, a sort s_i is **$\mathcal{J}$-irrelevant** if there is no sort s_j such that $j \in \mathcal{J}$ and $(s_j, s_i) \in \mathcal{R}_{\lambda S}$. We say s_i is **irrelevant** if it is $[n]$-irrelevant.*
- *An index set $\mathcal{I}$ is **completely irrelevant** in λS, if for each i in $\mathcal{I}$,*
  - *s_i is top-sort-like and irrelevant;*

– s_{i-1} *is* $([n] \setminus \mathcal{I})$*-irrelevant.*

- *A sort* s_i *is* ***completely isolated*** *if* s_i *is top-sort-like and rule-isolated.*

In the case of complete irrelevance, if $\mathcal{I}$ is a singleton set $\{i\}$, then the only rule with s_{i-1} appearing second is (s_i, s_{i-1}). By considering sets of indices simultaneously, we can make weaker assumptions on these preceding sorts. The condition of $([n] \setminus \mathcal{I})$-irrelevance ensures that s_{i-1} becomes isolated after removing the rules associated with sorts in $\mathcal{S}_{\mathcal{I}}$. Note also that if $(s_i, s_i) \in \mathcal{R}_{\lambda S}$, then any completely irrelevant index set cannot contain $i-1$, i or $i+1$. In particular, for based systems, 1 and 2 cannot appear in any completely irrelevant index set. Finally, it is important that there is a *unique maximum completely irrelevant index set.* In particular, the union of any two completely irrelevant index sets is completely irrelevant.

4.2.1 Eliminating Completely Irrelevant Rules

This section contains the translation which removes the rules associated with sorts whose indices appear in a completely irrelevant index set. For the remainder of the section, fix such a set $\mathcal{I}$. We begin by showing that sorts in $\mathcal{S}_{\mathcal{I}}$ are sparsely inhabited.

Lemma 48. *Let* s_i *be an irrelevant sort such that* s_i *is a top-sort or* s_{i+1} *is irrelevant. For context* Γ *and expression* A*, if* $\Gamma \vdash A : s_i$*, then* $A = s_{i-1}$ *or* $A \in \mathsf{V}_{s_{i+1}}$*.*

Proof. If $i = n$, then this follows directly from the top-sort lemma (Lemma 7) and the fact that s_n is irrelevant. In fact, in this case s_n is inhabited solely by s_{n-1}. If $i \neq n$, this follows in a similar way, *i.e.*, by induction on the structure of A. By the generation lemma (lemma 1), A cannot be a Π-expression as s_i is irrelevant. Likewise, A cannot be a λ-expression since $s_i \neq_\beta \Pi x^B.\, C$ for any expressions B and C. Finally, A cannot be an application MN since the generation lemma would imply that there is a context Γ and expressions B and C such that $\Gamma \vdash M : \Pi x^B.\, C$ and $\Gamma \vdash \Pi x^B.\, C : s_{i+1}$. But this is not possible since $i+1$ is irrelevant. □

This does not hold if s_{i+1} is not irrelevant. If $(s_{i+1}, s_{i+1}) \in \mathcal{R}_{\lambda S}$, for example, then $\varnothing \vdash s_i \to s_i : s_{i+1}$ and $\varnothing \vdash \lambda x^{s_i}.\ x : s_i \to s_i$ so $\varnothing \vdash (\lambda x^{s_i}.\ x) s_{i-1} : s_i$. This is why we require *both* s_i and s_{i+1} to be irrelevant. Similarly, this does not hold for all expressions of degree i. If $(s_{i+2}, s_{i+2}) \in \mathcal{R}_{\lambda S}$ then $\varnothing \vdash s_{i+1} \to s_{i+1} : s_{i+2}$ and $\varnothing \vdash \lambda x^{s_{i+1}}.\ x : s_{i+1} \to s_{i+1}$.

The primary challenge moving forward is dealing with the fact that variables may appear as types of sort s_i. These variables are what will necessitate s_{i+1} being not just irrelevant, but also isolated. Regardless, the sparsity of types of sort s_i induces sparsity of expressions of degree $i - 1$.

Lemma 49. *For index i in $\mathcal{I}$, context Γ and expression M, if $\Gamma \vdash M : s_{i-1}$, then M is of the form $\Pi x_1{}^{A_1}. \ldots \Pi x_k{}^{A_k}.\ B$ where $\deg(A_j) \in \mathcal{I}$ for all j and either $B = s_{i-2}$ or $B \in \mathsf{V}_{s_i}$.*

Proof. By induction on the structure of derivations. The cases in which the last inference is an axiom, variable introduction, or weakening are straightforward. The last inference clearly cannot be an abstraction, and it cannot be an application since s_i is irrelevant. What follows are the remaining two cases.

Product Type Formation. Suppose the last inference is of the form

$$\frac{\Gamma \vdash A : s_j \qquad \Gamma, x : A \vdash B : s_{i-1}}{\Gamma \vdash \Pi x^A.\ B : s_{i-1}}$$

Since s_{i-1} is $(\mathcal{S}_{\lambda S} \setminus \mathcal{I})$-irrelevant, it must be that $j \in \mathcal{I}$. The desired result holds after applying the inductive hypothesis to the right antecedent judgment.

Conversion. Suppose the last inference is of the form

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash s_{i-1} : s_i}{\Gamma \vdash M : s_{i-1}}$$

where $A =_\beta s_{i-1}$. Note that $\deg(A) = i$ so $\Gamma \vdash A : s_i$ by type correctness. Thus, $A = s_{i-1}$ by Lemma 48, which means the inductive hypothesis can be applied directly to the left antecedent judgment. □

Lemma 50. *For index i in $\mathcal{I}$, context Γ, expression A and variable ${}^{s_{i+1}}x$, if $\Gamma \vdash A : {}^{s_{i+1}}x$, then $A \in \mathsf{V}_{s_i}$.*

Proof. By induction on the structure of derivations. The cases in which the last inference is an axiom, variable introduction, or weakening are straightforward. The last inference clearly cannot be a product type formation or an abstraction. The last inference cannot be an application because s_i is irrelevant. Finally, all conversions are trivial by the same argument as in the previous lemma. □

Corollary 3. *For index i in $\mathcal{I}$, every derivable expression M of degree $i-1$ is of the form $\Pi {x_1}^{A_1}. \ldots \Pi {x_k}^{A_k}. B$ where $\deg(A_j) \in \mathcal{I}$ for all j and $B = s_{i-2}$ or $B \in \mathsf{V}_{s_i}$ (and k may be 0).*

The Translation

The following translation is defined such that it essentially pre-reduces all redexes whose source types have degree in $\mathcal{I}$. Naturally, this means it does not strictly preserve β-reductions, but because these sources types are so sparsely inhabited, we can define a complexity measure on expressions which is monotonically decreasing in the β-reductions that are pre-performed by the translation. This is similar to the technique used by Sørensen for simulating π-reductions Sørensen [1997].

The other wrinkle in defining this translation is that it is difficult to pre-reduce expressions of variable type because even though such types are sparse-inhabitance, it is unclear *a priori* what the value of the expression will be after a series of reductions. By Lemma 50, we know it reduces to a variable, but we don't know *which* variable, and it may be one that is generalized or abstracted over. We ensure this doesn't happen by requiring s_{i+1} is isolated, not just irrelevant. We also introduce a distinguished variable $\bullet_z$ of type z for each variable z of sort s_{i+1} in the context. This gives us a canonical term that the translation can assign to expressions of this type.

Definition 40. *The context-indexed family of functions* $(\tau_\Gamma : \mathsf{T} \to \mathsf{T})_{\Gamma \in \mathcal{C}}$ *as given as follows.*

$$\tau_\Gamma(s_i) \triangleq s_i$$

$$\tau_\Gamma({}^{s_i}x) \triangleq \begin{cases} s_{i-2} & i \in \mathcal{I} \text{ and } ({}^{s_i}x : s_{i-1}) \in \Gamma \\ \bullet_z & i \in \mathcal{I} \text{ and } ({}^{s_i}x : {}^{s_{i+1}}z) \in \Gamma \\ {}^{s_i}x & \textit{otherwise} \end{cases}$$

$$\tau_\Gamma(\Pi x^A.\ B) \triangleq \begin{cases} \tau_{\Gamma,x:A}(B) & \deg(A) \in \mathcal{I} \\ \Pi x^{\tau_\Gamma(A)}.\ \tau_{\Gamma,x:A}(B) & \textit{otherwise} \end{cases}$$

$$\tau_\Gamma(\lambda x^A.\ M) \triangleq \begin{cases} \tau_{\Gamma,x:A}(M) & \deg(A) \in \mathcal{I} \\ \lambda x^{\tau_\Gamma(A)}.\ \tau_{\Gamma,x:A}(M) & \textit{otherwise} \end{cases}$$

$$\tau_\Gamma(MN) \triangleq \begin{cases} \tau_\Gamma(M) & \deg(N) + 1 \in \mathcal{I} \\ \tau_\Gamma(M)\tau_\Gamma(N) & \textit{otherwise} \end{cases}$$

where $\bullet_z$ *is a distinguished variable. This family of function is extended to a single function on contexts as*

$$\tau(\varnothing) \triangleq \varnothing$$

$$\tau(\Gamma, {}^{s_j}x : A) \triangleq \begin{cases} \tau(\Gamma) & j \in \mathcal{I} \\ \tau(\Gamma), {}^{s_j}x : s_{j-1}, \bullet_x : {}^{s_j}x & j - 1 \in \mathcal{I} \text{ and } A = s_{j-1} \\ \tau(\Gamma), {}^{s_j}x : \tau_\Gamma(A) & \textit{otherwise.} \end{cases}$$

Note that this class of functions is well-defined with respect to the dependence on contexts; each function can be defined simultaneously, inductively on the structure of the input. As for proving the desired features of this translation, first note if $i \in \mathcal{I}$, then the translation

maps terms of degree $i-1$ (where $i \in \mathcal{I}$) to canonical atomic terms.

Proposition 1. *For any index i in $\mathcal{I}$, context Γ, and term A, if $\Gamma \vdash A : s_{i-1}$, then $\tau_\Gamma(A) = s_{i-2}$, and if $\Gamma \vdash A : {}^{s_{i+1}}x$ for some variable ${}^{s_{i+1}}x$, then $\tau_\Gamma(A) = \bullet_x$.*

It suffices to consider the expressions of the form specified by Corollary 3, for which the above fact clearly holds. This turns out to be a key feature of the translation. Because the translation is able to drop so much information about these expressions, we can pre-reduce redexes in which they appear on the right.

We also use the fact that the context associated with a translation can be weakened when the variable does not appear in its argument.

Proposition 2. *For any index i, context Γ, expressions M, A, and B, and variable ${}^{s_i}x$, if $\Gamma \vdash M : A$ and $\Gamma \vdash B : s_i$ then $\tau_{\Gamma,{}^{s_i}x:B}(M) = \tau_\Gamma(M)$.*

We now prove the standard substitution-commutation and β-preservation lemmas for this translation.

Lemma 51. *For any index i, context Γ, expressions M, N, A and B, and variable ${}^{s_i}x$, if $\Gamma, {}^{s_i}x : A \vdash M : B$ and $\Gamma \vdash N : A$ then*

$$\tau_\Gamma(M[N/{}^{s_i}x]) = \begin{cases} \tau_{\Gamma,{}^{s_i}x:A}(M) & i \in \mathcal{I} \\ \tau_{\Gamma,{}^{s_i}x:A}(M)[\tau_\Gamma(N)/{}^{s_i}x] & \textit{otherwise.} \end{cases}$$

Proof. By induction on the structure of M. First suppose that $i \in \mathcal{I}$.

Sort. If M is of the form s_j, then $\tau_\Gamma(s_j[N/{}^{s_i}x]) = \tau_\Gamma(s_j)$.

Variable. First suppose M is of the form ${}^{s_i}x$. In particular, $A =_\beta B$, and since $\deg(A) = \deg(B) = i$, we have $A = B$ by Lemma 48. If $A = s_{i-1}$, then by Proposition 1 we have $\tau_\Gamma(N) = s_{i-2}$ and

$$\tau_\Gamma({}^{s_i}x[N/{}^{s_i}x]) = \tau_\Gamma(N) = s_{i-2} = \tau_{\Gamma,x:s_{i-1}}({}^{s_i}x).$$

Similarly, if A is of the form ${}^{s_{i+1}}y$, then $\tau_\Gamma(N) = \bullet_y$ and

$$\tau_\Gamma({}^{s_i}x[N/{}^{s_i}x]) = \tau_\Gamma(N) = \bullet_y = \tau_{\Gamma,x:y}({}^{s_i}x).$$

If M is of the form ${}^{s_j}y$ where ${}^{s_j}y \neq {}^{s_i}x$, then $\tau({}^{s_j}y[N/{}^{s_i}x]) = \tau({}^{s_j}y)$.

Π-Expression. If M is of the form $\Pi y^A.\ B$, then

$$\begin{aligned}\tau_\Gamma((\Pi y^A.\ B)[N/x]) &= \tau_\Gamma(\Pi y^{A[N/x]}.\ B[N/x]) \\ &= \begin{cases} \tau_{\Gamma,y:A}(B) & \deg(A) \in \mathcal{I} \\ \Pi y^{\tau_\Gamma(A)}.\ \tau_{\Gamma,y:A}(B) & \text{otherwise} \end{cases}\end{aligned}$$

where the last equality follows from the definition of τ and the inductive hypothesis. This also depends on Proposition 2 to show that $\tau_{\Gamma,y:A}(A) = \tau_\Gamma(A)$. The cases in which M is a λ-expression or application are similar. Furthermore, when $i \notin \mathcal{I}$, all cases are analogous. □

Before proving the β-preservation lemma, it is convenient to partition the β-reduction relation into two parts, one part which is directly preserved by the translation (β_1) and one part which is pre-reduced by the translation (β_2).

Definition 41. *Let β_2 denote the notion of reduction given by*

$$(\lambda x^A.\ M)N \to_{\beta_2} M[N/x]$$

where $\deg(A) \in \mathcal{I}$, extended to a congruence relation in the usual way. Let β_1 denote the same notion of reduction but with $\deg(A) \notin I$, so that $\beta_1 \cap \beta_2 = \emptyset$ and $\beta_1 \cup \beta_2 = \beta$.

Lemma 52. *For expressions M and N derivable in the context Γ, the following hold.*

- *If $M \to_{\beta_1} N$, then $\tau_\Gamma(M) \to_\beta \tau_\Gamma(N)$;*
- *if $M \to_{\beta_2} N$, then $\tau_\Gamma(M) = \tau_\Gamma(N)$;*

- *in particular, if* $M =_\beta N$*, then* $\tau_\Gamma(M) =_\beta \tau_\Gamma(N)$.

Proof. The last item follows directly from the first two. We prove the first two items by induction on the structure of the one-step β-reduction relation. In the case a redex $(\lambda x^A.\ M)N$, if $\deg(A) \notin \mathcal{I}$, then we have

$$\begin{aligned}\tau_\Gamma((\lambda x^A.\ M)N) &= \tau_\Gamma(\lambda x^A.\ M)\tau_\Gamma(N) \\ &= (\lambda x^{\tau_\Gamma(A)}.\ \tau_{\Gamma,x:A}(M))\tau_\Gamma(N) \\ &\to_\beta \tau_{\Gamma,x:A}(M)[\tau_\Gamma(N)/x] \\ &= \tau_\Gamma(M[N/x])\end{aligned}$$

and otherwise,

$$\begin{aligned}\tau_\Gamma((\lambda x^A.\ M)N) &= \tau_\Gamma(\lambda x^A.\ M) \\ &= \tau_{\Gamma,x:A}(M) \\ &= \tau_\Gamma(M[N/x])\end{aligned}$$

where the last equality in each sequence of equalities follows from the substitution-commutation lemma (Lemma 51). To show the desired result holds up to congruences, it must follow that expressions dropped by the translation are already in normal form.

Π-Expression. Suppose M is of the form $\Pi x^A.\ B$ and N is of the form $\Pi x^{A'}.\ B'$ where

$$\Pi x^A.\ B \to_\beta \Pi x^{A'}.\ B'$$

If $\deg(A) \notin \mathcal{I}$, then either $A \to_\beta A'$ and $B = B'$ or $B \to_\beta B'$ and $A = A'$ and the inductive hypothesis can be safely applied. If $\deg(A) \in \mathcal{I}$, then Lemma 48 implies that A is in normal form, so $A = A'$ and $B \to_\beta B'$, and the inductive hypothesis can be safely applied. The case in which M is a λ-expression is similar.

Application. Suppose M is of the form PQ and N is of the form $P'Q'$ where

$$PQ \to_{\beta_1} P'Q'$$

If $\deg(Q) + 1 \notin \mathcal{I}$, then either $P \to_\beta P'$ and $Q = Q'$ or $Q \to_\beta Q'$ and $P = P'$ and the inductive hypothesis can be safely applied. If $\deg(Q) + 1 \in \mathcal{I}$, Corollary 3 implies that Q is in normal form, so $Q = Q'$ and $P \to_\beta P'$ and the inductive hypothesis can be safely applied. □

With these two lemmas, we can now prove that the translation preserves typability. The system we translate to is defined simply as the one in which the rules associated with sorts in $\mathcal{S}_\mathcal{I}$ are dropped.

Definition 42. *The **irrelevance reduction** of an n-tiered pure type system λS, denoted here by λS^-, is the n-tiered system specified by the rules*

$$\mathcal{R}_{\lambda S} \setminus \{(s_i, s_j) \mid i \in \mathcal{I} \text{ and } j \in [n]\}.$$

Lemma 53. *For context Γ and expressions M and A, if*

$$\Gamma \vdash_{\lambda S} M : A \qquad \text{then} \qquad \tau(\Gamma) \vdash_{\lambda S^-} \tau_\Gamma(M) : \tau_\Gamma(A).$$

Proof. By induction on the structure of derivations.

Axiom. If the derivation is a single axiom $\vdash s_i : s_{i+1}$ then the translated derivation is the same axiom.

Variable Introduction. Suppose the last inference is of the form

$$\frac{\Gamma \vdash A : s_i}{\Gamma, {}^{s_i}x : A \vdash {}^{s_i}x : A}$$

First suppose $i \in \mathcal{I}$. If $A = s_{i-1}$, then $\tau_{\Gamma,x:s_{i-1}}(x) = s_{i-2}$ and $\tau(\Gamma) \vdash s_{i-2} : s_{i-1}$ where $\tau(\Gamma)$ is well-formed by the inductive hypothesis; that is, $\tau(\Gamma) \vdash \tau_\Gamma(A) : s_i$ implies $\tau(\Gamma)$ is well-formed. If A is of the form ${}^{s_{i+1}}y$, then $({}^{s_{i+1}}y : s_{i-1}) \in \Gamma$, which implies $(\bullet_y : {}^{s_{i+1}}y) \in \tau(\Gamma)$ and $\tau(\Gamma) \vdash \bullet_y : {}^{s_{i+1}}y$ where $\tau(\Gamma)$ is again well-formed by the inductive hypothesis.

Next suppose $i - 1 \in \mathcal{I}$ and $A = s_{i-1}$. By the inductive hypothesis, we can derive

$$\frac{\tau(\Gamma) \vdash s_{i-1} : s_i}{\tau(\Gamma), {}^{s_i}x : s_{i-1} \vdash {}^{s_i}x : s_{i-1}}$$

and so by weakening,

$$\frac{\tau(\Gamma), {}^{s_i}x : s_{i-1} \vdash {}^{s_i}x : s_{i-1} \qquad \tau(\Gamma), {}^{s_i}x : s_{i-1} \vdash {}^{s_i}x : s_{i-1}}{\tau(\Gamma), {}^{s_i}x : s_{i-1}, \bullet_x : {}^{s_i}x \vdash {}^{s_i}x : s_{i-1}}$$

The remaining cases are straightforward.

Weakening. Suppose the last inference is of the form

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s_i}{\Gamma, {}^{s_i}x : B \vdash M : A}$$

By Proposition 1, we have $\tau_{\Gamma,x:B}(M) = \tau_\Gamma(M)$. By type correctness, $\Gamma \vdash A : s_j$ for some index j, so $\tau_{\Gamma,x:B}(A) = \tau_\Gamma(A)$. So the inductive hypothesis implies

$$\tau(\Gamma) \vdash \tau_{\Gamma,x:B}(M) : \tau_{\Gamma,x:B}(A)$$

We can then use an argument similar to the one in the previous case to extend the context to $\tau(\Gamma, x : B)$.

Product Type Formation. Suppose the last inference is of the form

$$\frac{\Gamma \vdash A : s_i \qquad \Gamma, x : A \vdash B : s_j}{\Gamma \vdash \Pi x^A.\ B : s_j}$$

if $i \in \mathcal{I}$, then $\tau(\Gamma) = \tau(\Gamma, x : A)$ and $\tau_\Gamma(\Pi x^A.\ B) = \tau_{\Gamma,x:A}(B)$ and so $\tau(\Gamma) \vdash \tau_{\Gamma,x:A}(B) : s_j$ by the inductive hypothesis applied to the right antecedent judgment. It cannot be the case that $i - 1 \in \mathcal{I}$ and $A = s_{i-1}$ since s_i is rule-isolated in this case. The remaining case is straightforward.

Abstraction. Suppose the last inference is of the form

$$\frac{\Gamma, {}^{s_i}x : A \vdash M : B \qquad \Gamma \vdash \Pi x^A.\ B : s_j}{\Gamma \vdash \lambda x^A.\ M : \Pi x^A.\ B}$$

If $i \in \mathcal{I}$, then

$$\tau(\Gamma) = \tau(\Gamma, {}^{s_i}x : A)$$
$$\tau_\Gamma(\lambda x^A.\ M) = \tau_{\Gamma, x:A}(M)$$
$$\tau_\Gamma(\Pi x^A.\ B) = \tau_{\Gamma, x:A}(B)$$

so the desired judgment follows directly from the inductive hypothesis applied to the left antecedent judgment. Again, it cannot be the case that $i - 1 \in \mathcal{I}$ and $A = s_{i-1}$ since s_i is rule-isolated in this case. The remaining case is straightforward.

Application. Suppose the last inference is of the form

$$\frac{\Gamma \vdash M : \Pi x^A.\ B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B[N/{}^{s_i}x]}$$

By type correctness, $\Gamma \vdash \Pi x^A.\ B : s_j$ for some sort s_j, and by generation, we have

$$\Gamma, {}^{s_i}x : A \vdash B : s_j$$

so by Lemma 51, if $i \in \mathcal{I}$ (*i.e.*, $\deg(N) + 1 \in \mathcal{I}$), then $\tau_\Gamma(MN) = \tau_\Gamma(M)$ and

$$\tau_\Gamma(B[N/{}^{s_i}x]) = \tau_{\Gamma, x:A}(B) = \tau_\Gamma(\Pi x^A.\ B).$$

The desired result then follows directly from the inductive hypothesis applied to the left antecedent judgment. And if $i \notin \mathcal{I}$, then $\tau_\Gamma(B[N/x]) = \tau_{\Gamma, x:A}(B)[\tau_\Gamma(N)/x]$ and we have

$$\frac{\tau(\Gamma) \vdash \tau_\Gamma(M) : \Pi x^{\tau_\Gamma(A)}.\ \tau_{\Gamma, x:A}(B) \qquad \tau(\Gamma) \vdash \tau_\Gamma(N) : \tau_\Gamma(A)}{\tau(\Gamma) \vdash \tau_\Gamma(M)\tau_\Gamma(N) : \tau_{\Gamma, x:A}(B)[\tau_\Gamma(N)/x]}$$

Conversion. Suppose the last inference is of the form

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s_i}{\Gamma \vdash M : B}$$

where $A =_\beta B$. Then we have

$$\frac{\tau(\Gamma) \vdash \tau_\Gamma(M) : \tau_\Gamma(A) \qquad \tau(\Gamma) \vdash \tau_\Gamma(B) : s_i}{\tau(\Gamma) \vdash \tau_\Gamma(M) : \tau_\Gamma(B)}$$

where $\tau_\Gamma(A) =_\beta \tau_\Gamma(B)$ by Lemma 52. □

It remains to show that this translation is path-preserving. The guiding observation is that β_2-reductions cannot make more redexes, and what redexes may persist must be simpler in some sense. We define a complexity measure which captures this observation by its being monotonically decreasing in β_2-reductions.

Definition 43. *The **shallow λ-depth** of a term M is the number of top-level λ's appearing in it, i.e., the function $\delta : \mathsf{T} \to \mathbb{N}$ is given by $\delta(\lambda x^A.\ N) \triangleq 1 + \delta(N)$ and $\delta(M) \triangleq 0$ otherwise. The **shallow λ-depth of a redex** $(\lambda x^A.\ M)N$ is the shallow λ-depth of its left term $\lambda x^A.\ M$.*

I will simply write "depth" from this point forward.

Definition 44. *Define $\mu : \mathsf{T} \to \mathbb{N}$ to be the function which maps an expression to the sum of the depths of its β_2-redexes, i.e.,*

$$\mu(s_i) = \mu(x) \triangleq 0$$

$$\mu(\Pi x^A.\ B) = \mu(\lambda x^A.\ B) \triangleq \mu(A) + \mu(B)$$

$$\mu(MN) \triangleq \begin{cases} \mu(M) + \mu(N) + \delta(MN) & MN \text{ is a } \beta_2\text{-redex} \\ \mu(M) + \mu(N) & \textit{otherwise.} \end{cases}$$

Finally, we prove the monotonicity lemma. It depends on the typed-version of a result by Lévy for the untyped lambda calculus about the creation of new redexes Lévy [1978]. I give

the statement of the result here without proof. I also include a definition of the standard notion of a *residual* redex, adapted from Xi [2006] as well as Huet [1994].

Definition 45. *Let M and N be expressions such that $M \to_\beta N$ by reducing the redex $(\lambda x^A.\ P)Q$, and let R be a redex in M. The **residuals** of R in N are the copies of R which appear in N after reducing $(\lambda x^A.\ P)Q$. That is,*

- *if R is $(\lambda x^A.\ P)Q$, then R has no residuals.*
- *if $R \subset Q$, then the residuals are R in N are the copies of R in $P[Q/x]$;*
- *if $R \subset P$, then the copy of R in P after substitution of the form $R[Q/x]$ is the residual of R in N;*
- *if the redex $(\lambda x^A.\ P)Q$ is contained in R, then R after substitution (i.e., R with $(\lambda x^A.\ P)Q$ replaced by $P[Q/x]$) is the residual of R in N*
- *if R is disjoint from $(\lambda x^A.\ P)Q$ then R itself is the residual of R in N.*

Lemma 54. *(Lévy, 1978) For expressions M and N such that $M \to_\beta N$, if $(\lambda x^A.\ P)Q$ is a redex of N which is not a residual of a redex in M, then it is creates in one of the following ways.*

1. $(\lambda y^B.\ y)(\lambda x^A.\ P)Q \to_\beta (\lambda x^A.\ P)Q$;
2. $(\lambda y^C.\ \lambda x^D.\ R)SQ \to_\beta (\lambda x^{D[S/y]}.\ R[S/y])Q$ *where* $A = D[S/y]$ *and* $P = R[S/y]$;
3. $(\lambda y^B.\ R)(\lambda x^A.\ P) \to_\beta R[\lambda x^A.\ P/y]$ *where yQ is a sub-expression of R.*

Lemma 55. *For derivable expressions M and N, if $M \to_{\beta_2} N$, then $\mu(M) > \mu(N)$.*

Proof. Suppose M reduces to N by reducing the β_2-redex $(\lambda x^C.\ P)Q$. By Corollary 3, the expression Q is of the form $\Pi {x_1}^{A_1}. \ldots \Pi {x_k}^{A_k}.\ B$ where $\deg(A_j) \in \mathcal{I}$ for all j and either $B = s_{i-2}$ or $B \in \mathsf{V}_{s_i}$. This means reducing a β_2-redex cannot duplicate existing redexes in

M, so every redex has at most one residual in N. Furthermore, if N has a new β_2-redex, it is by item 2 of Lemma 54, *i.e.*, there are expressions C, D, R, and S, and variable z such that $P = \lambda z^D.\ R$ and

$$(\lambda x^C.\ \lambda z^D.\ R)QS \to_\beta (\lambda z^{D[Q/x]}.\ R[Q/x])S.$$

It is easy to verify that, because of the form of Q, only one new β-redex is created and, furthermore, $\delta(R[Q/x]) \leq \delta(R)$. This implies the new redex has smaller depth than the redex that was reduced, so even if it is a β_2-redex, the complexity of M decreases.

□

The proof of the main theorem of this sub-section is standard.

Theorem 4. *If λS^- is strongly normalizing, then λS is strongly normalizing.*

Proof. Suppose there is an infinite reduction sequence in λS

$$M_1 \to_\beta M_2 \to_\beta \dots$$

where M_1 is derivable from the context Γ. Since μ is monotonically decreasing in β_2-reductions (Lemma 55), there cannot be an infinite sequence of solely β_2-reductions contained in this sequence. This means there are infinitely many β_1 reductions in this sequence, which by Lemma 52 implies there infinitely many β-reductions in the reduction path

$$\tau_\Gamma(M_1) \to_\beta \tau_\Gamma(M_2) \to_\beta \dots$$

which is in λS^- by Lemma 53. □

4.2.2 Eliminating Completely Isolated Sorts

We now handle completely isolated sorts. Recall that a sort s_i is completely isolated if s_i is top-sort-like and rule-isolated. This translation is slightly simpler than the first. It is a generalization of the observation made in the introduction that one can define a path-preserving translation from $\lambda\mathsf{HOL}$ to $\lambda\omega$, *i.e.*, one that eliminates the rule-isolated top-sort. The two translations can, in fact, be collapsed into a single translation, but it is convenient to separate it into two parts, especially since the result can be made slightly stronger; the conditions on s_{i-1} are weaker for completely isolation than for complete irrelevancy.

Fix an n-tiered pure type system λS with $n > 2$, and a completely isolated sort s_i.[2] In essence, the following translation removes the completely isolated sort and shifts down all the sorts that might be above it. Because isolated sorts can only really be used to introduce variables into the context, the translation pre-substitutes those variables with dummy values that won't affect the normalization behavior of the expression after translation.

One notable feature of this translation is that it does not preserve the number of sorts in the system and, furthermore, does not preserve degree. Thus, it will be useful to be more careful about variable annotations in the following definitions and lemmas.

2. The restriction on n is a technicality that ensures the target system is nontrivial. See, for example, the variable case of Definition 46.

Definition 46. *The context-indexed family of function* $\{\theta_\Gamma : \mathsf{T} \to \mathsf{T}\}_{\Gamma \in \mathcal{C}}$ *is given as follows.*

$$\theta_\Gamma(s_j) \triangleq \begin{cases} s_j & j < i \\ s_{j-1} & \textit{otherwise} \end{cases}$$

$$\theta_\Gamma({}^{s_j}x) \triangleq \begin{cases} s_{i-2} & j = i \text{ and } ({}^{s_i}x : s_{i-1}) \in \Gamma \\ {}^{s_j}x & j < i \\ {}^{s_{j-1}}x & \textit{otherwise} \end{cases}$$

$$\theta_\Gamma(\Pi^{s_j}x^A.\ B) \triangleq \Pi^{\theta_\Gamma(s_j)}x^{\theta_\Gamma(A)}.\ \theta_{\Gamma,x:A}(B)$$

$$\theta_\Gamma(\lambda^{s_j}x^A.\ M) \triangleq \lambda^{\theta_\Gamma(s_j)}x^{\theta_\Gamma(A)}.\ \theta_{\Gamma,x:A}(M)$$

$$\theta_\Gamma(MN) \triangleq \theta_\Gamma(M)\theta_\Gamma(N)$$

This family of functions is extended to a single function on contexts as

$$\theta(\varnothing) \triangleq \varnothing$$

$$\theta(\Gamma, {}^{s_j}x : A) \triangleq \begin{cases} \theta(\Gamma) & j = i \text{ and } A = s_{i-1} \\ \theta(\Gamma), {}^{\theta_\Gamma(s_j)}x : \theta_\Gamma(A) & \textit{otherwise.} \end{cases}$$

As with the previous translation, this one is well-defined with respect to its dependence on context, and the contexts can be weakened without changing the value of the function (in analogy with Proposition 2 for τ_Γ). We go on to prove substitution-commutation, β-reduction preservation, and typability preservation. The proofs are similar to those in the previous sub-section and, consequently, are slightly abbreviated.

Lemma 56. *For context* Γ*, expressions* M*,* N*,* A *and* B*, and variable* ${}^{s_j}x$*, if* $j \neq i$ *and* $\Gamma, {}^{s_j}x : A \vdash M : B$ *and* $\Gamma \vdash N : A$ *then* $\theta_\Gamma(M[N/{}^{s_j}x]) = \theta_{\Gamma,{}^{s_j}x:A}(M)[\theta_\Gamma(N)/{}^{\theta_\Gamma(s_j)}x]$.

Proof. By induction on the structure of M. All cases are straightforward except the case

in which M is a variable, but then the assumption that $j \neq i$ ensures the desired equality holds. □

Lemma 57. *For expressions M and N derivable from Γ, if $M \to_\beta N$, then $\theta_\Gamma(M) \to_\beta \theta_\Gamma(N)$. Furthermore, if $M =_\beta N$, then $\theta_\Gamma(M) =_\beta \theta_\Gamma(N)$.*

Proof. The second part follows directly from the first, which follows by induction on the structure of the one-step β-reduction relation. In the case of a redex $(\lambda x^A.\ M)N$, we have

$$\begin{aligned}\theta_\Gamma((\lambda x^A.\ M)N) &= (\lambda x^{\theta_\Gamma(A)}.\ \theta_{\Gamma,x:A}(M))\theta_\Gamma(N)\\ &\to_\beta \theta_{\Gamma,x:A}(M)[\theta_\Gamma(N)/^{\theta_\Gamma(s_j)}x]\\ &= \theta_\Gamma(M[N/^{\theta_\Gamma(s_j)}x])\end{aligned}$$

where the last equality follows from Lemma 56, keeping in mind that $j \neq i$ since i is isolated, so the lemma can be safely applied. □

Finally, typability preservation. The target system is as expected, the isolated sort is removed and potential sorts above it are shifted down.

Definition 47. *The i-**collapse** of an n-tiered pure type system λS, denote here by λS^*, is the $(n-1)$-tiered systems specified by the rules*

$$\mathcal{R}_{\lambda S^*} \triangleq \{(\theta_\varnothing(s_j), \theta_\varnothing(s_k)) \mid (s_j, s_k) \in \mathcal{R}_{\lambda S} \text{ and } j \neq i \text{ and } k \neq i\}.$$

Lemma 58. *For context Γ and expressions M and A where $M \neq s_{i-1}$, if*

$$\Gamma \vdash M : A \qquad \textit{then} \qquad \theta(\Gamma) \vdash \theta_\Gamma(M) : \theta_\Gamma(A).$$

Proof. By induction on the structure of derivations. The proof differs slightly depending on whether or not s_i is a top-sort. I make clear below which cases differ.

Axiom. Since $M \neq s_{i-1}$, the judgment $\varnothing \vdash \theta_\varnothing(s_j) : \theta_\varnothing(s_{j+1})$ is still an axiom.

Variable Introduction. Suppose the last inference is of the form

$$\frac{\Gamma \vdash A : s_j}{\Gamma, {}^{s_j}x : A \vdash {}^{s_j}x : A}$$

If $j = i$ and $A = s_{i-1}$, then $\theta(\Gamma) \vdash s_{i-2} : s_{i-1}$ is still derivable. Note that $\theta(\Gamma)$ can be proved to be well-formed by the inductive hypothesis. If $j < i$, then we have

$$\frac{\theta(\Gamma) \vdash \theta_\Gamma(A) : s_j}{\theta(\Gamma), {}^{s_j}x : \theta_\Gamma(A) \vdash {}^{s_j}x : \theta_\Gamma(A)}$$

If $j > i$, then in particular s_i is not a top-sort. This case is then similar to the previous one, keeping in mind that this might use the rule (s_{i-1}, s_i) for the translated derivation *in the system* λS^*, but not in the case that s_i is a top-sort.

Weakening. This case follows directly from the fact that $\theta_{\Gamma,x:B}(M) = \theta_\Gamma(M)$ whenever M and B are derivable from Γ. It is also similar to the analogous case in the previous sub-section.

Product Type Formation. Suppose the last inference is of the form

$$\frac{\Gamma \vdash A : s_j \qquad \Gamma, {}^{s_j}x : A \vdash B : s_k}{\Gamma \vdash \Pi x^A.\ B : s_k}$$

Note that $j \neq i$ and $k \neq i$ since s_i is rule-isolated. In particular, neither A nor B are s_{i-1}. Therefore, we can apply the inductive hypothesis directly to each antecedent judgment and derive the desired consequent judgment.

Abstraction. Suppose the last inference is of the form

$$\frac{\Gamma, {}^{s_j}x : A \vdash M : B \qquad \Gamma \vdash \Pi x^A.\ B : s_k}{\Gamma \vdash \lambda x^A.\ M : \Pi x^A.\ B}$$

Note that $j \neq i$ since s_i is rule-isolated, and so $\Pi x^A.\ B$ would not be derivable. Furthermore, $B \neq s_i$ (so $M \neq s_{i-1}$) since s_i is irrelevant. Therefore, we can apply the inductive hypothesis directly to each antecedent judgment and derive the desired consequent judgment.

Application. Suppose the last inference is of the form

$$\frac{\Gamma \vdash M : \Pi x^A.\ B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B[N/x]}$$

Note that $\deg(A) \neq i+1$ (and in particular $N \neq s_{i-1}$), since s_{i+1} is rule-isolated. Furthermore, $\deg(A) \neq i$ (and $\deg(N) \neq i-1$) since s_i is rule-isolated. Therefore, we can apply the inductive hypothesis directly to each antecedent judgment.

$$\theta(\Gamma) \vdash \theta_\Gamma(M)\theta_\Gamma(N) : \theta_{\Gamma,x:A}(B)[\theta_\Gamma(N)/x]$$

and $\theta_{\Gamma,x:A}(B)[\theta_\Gamma(N)/x] = \theta_\Gamma(B[N/x])$ by Lemma 56.

Conversion. Suppose the last inference is of the form

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s_j}{\Gamma \vdash M : B}$$

If $M = s_{i-1}$, then $A =_\beta s_i =_\beta B$. Then by Lemma 48, in fact $A = B$. If $B = s_{i-1}$, then by Corollary 3 we again have $A = B$. Otherwise, by Lemma 57, $\theta_\Gamma(A) =_\beta \theta_\Gamma(B)$ and we can derive $\theta(\Gamma) \vdash \theta_\Gamma(M) : \theta_\Gamma(B)$ by the inductive hypothesis and conversion. □

Since β-reductions are simulated directly, the argument for the final theorem is simple.

Theorem 5. *If λS^* is strongly normalizing then λS is strongly normalizing.*

Proof. Suppose there is an infinite reduction path in λS starting at M, which is derivable from the context Γ. Note that this sequence cannot contain the term s_{i-1} since this is a normal form. Therefore, applying θ_Γ to each term yields an infinite reduction path in λS^*. □

It is also important to note that weak normalization is preserved in the opposite direction.

Proposition 3. *If λS is weakly normalizing then so is λS^* is weakly normalizing.*

This is trivial in the case that the sorts are maintained—as for the irrelevance reduction—and also in the case that the eliminated sort is a top-sort, but requires, in the case

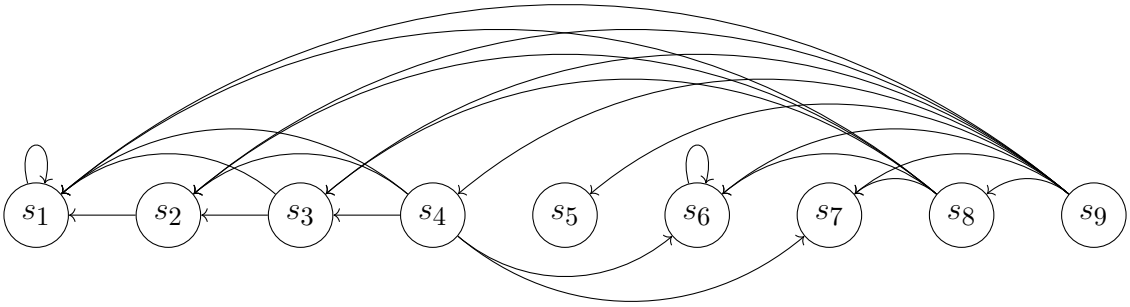


Figure 4.1: The iteration of irrelevance reductions can be fairly complex; it may be that a sort cannot appear in a completely irrelevant index set until after several iterations. In the system above, the maximum completely irrelevant index set of this system is $\{9\}$, but after eliminating the rules associated with s_9, both s_9 and s_5 become rule-isolated, and so the next maximum completely irrelevant index set is $\{4, 8\}$. It is not difficult to imagine how this effect can be scaled up to larger systems.

of a top-sort-like sort, noting that any term in λS^* can be embedded into λS up to sort renaming.

4.2.3 The Main Result

With these two translations, we can now define one final translation, which is the composition of the fixed-points of these to translations.

Definition 48. *For any tiered pure type system λS, let $\mathcal{I}_{\lambda S}$ denote its unique maximum completely irrelevant index set and let $i_{\lambda S}$ denote the maximum index of a completely isolated sort in λS, if one exists. Let $\tau(\lambda S)$ denote the fixed-point of taking the irrelevance reduction of λS with respect to its maximum completely irrelevant set $\mathcal{I}_{\lambda S}$ (i.e., until $\mathcal{I}_{\lambda S} = \emptyset$) and let $\theta(\lambda S)$ denote the fixed-point of taking the $i_{\lambda S}$-collapse of λS (i.e., until λS has no completely isolated sort or is 2-tiered). The **irrelevance elimination of** λS, denoted as $\lambda S^{\downarrow}$ is the system $\theta(\tau(\lambda S))$.*

See Figure 4.1 for an example of this transformation. The main theorem of this paper is as follows. It is simply the observation that the translations from the previous sub-sections

can be composed.

Theorem 6. *For any tiered pure type system λS, if $\lambda S^{\downarrow}$ is strongly normalizing, then λS is strongly normalizing.*

And, as prefaced above, this result can be bootstrapped with existing results for the Barendregt-Geuvers-Klop conjecture in the expected way.

Corollary 4. *For any tiered pure type system λS, if weak normalization implies strong normalization for $\lambda S^{\downarrow}$, then weak normalization implies strong normalization for λS.*

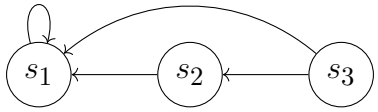
Proof. If λS is weakly normalizing, then $\lambda S^{\downarrow}$ is weakly normalizing, since any term in $\lambda S^{\downarrow}$ can be embedded in λS. But then $\lambda S^{\downarrow}$ is strongly normalizing by assumption, which implies λS is strongly normalizing by Theorem 6. □

By the result of Barthe *et al.*, if $\lambda S^{\downarrow}$ is non-dependent, clean, and negatable, then weak normalization implies strong normalization in λS. Since $\lambda S^{\downarrow}$ can be non-dependent even if λS has dependent rules, this extension allows us to prove the conjecture of some systems with dependencies. Cleanliness is a technical restriction that I won't go into here, but a pure type system is **negatable** if it has the rule (s_i, s_i) whenever s_i is relevant. So this extension also allows us to consider systems with non-negatable sorts.

4.3 Conclusions

I have presented a path-preserving translation from a tiered pure type system λS to a weaker system $\lambda S^{\downarrow}$ which is, in essence, λS without its irrelevant structure. When combined with results for the Barendregt-Geuvers-Klop conjecture, it widens the class of systems for which the conjecture applies, most notably to systems with dependent rules, but also to systems with non-negatable sorts. This is a step towards proving the conjecture for all tiered systems, in particular because it highlights those systems which require further analysis. For example,

besides dealing with systems that have more dependences, it appears that dealing with circular rules is one of the clear barriers in strengthening these results. For 3-tiered systems, we extend existing results to include the system specified by



but not to the same system with the additional rule (s_3, s_3). Circular rules break irrelevancy and, consequently, induce much more complicated structure in the system. There is also likely other forms of irrelevant structure that are not covered by the translation in this paper.

Additionally, it is worth noting that the conditions on completely irrelevant index sets cannot be trivially weakened. If, for example the irrelevance condition on preceding sorts was removed, this technique would apply to λU (*i.e.*, the same system presented above but with the additional rule (s_2, s_2)), leading to a contradiction since λU is non-normalizing. Circular rules again seem to be at the core of this issue. More carefully considering λU and related non-normalizing systems through the lens of these results—particularly why the techniques don't apply to these systems—may yield a more structural understanding of the non-normalization of λU, independent of Girard's paradox. Regardless, I hope to have demonstrated with this translation that, despite the full Barendregt-Geuvers-Klop conjecture seeming quite far from being solved, there are still a number of approachable questions and avenues for further development.

REFERENCES

Henk Barendregt. Introduction to generalized type systems. *Journal of Functional Programming*, 1(2):125–154, 1991.

Henk Barendregt. Lambda Calculi with Types. In *Handbook of Logic in Computer Science, Volume II*, pages 117–309. Oxford University Press, 1993.

Henk Barendregt and Giulio Manzonettoa. Turing's Contributions to Lambda Calculus. In *Alan Turing: His Work and Impact*, pages 139–144. Elsevier, 2013.

Henk Barendregt, Wil Dekkers, and Richard Statman. *Lambda Calculus with Types*. Perspectives in Logic. Cambridge University Press, 2013.

Henk P. Barendregt. *The Lambda Calculus, It's Syntax and Semantics*. North-Holland Amsterdam, 1984.

Gilles Barthe and Morten Heine Sørensen. Domain-free pure type systems. *Journal of Functional Programming*, 10(5):417–452, 2000.

Gilles Barthe, John Hatcliff, and Morten Heine Sørensen. Weak normalization implies strong normalization in a class of non-dependent pure type systems. *Theoretical Computer Science*, 269(1-2):317–361, 2001.

Stefano Berardi. Towards a mathematical analysis of the Coquand-Huet calculus of constructions and the other systems in Barendregt's cube. Technical report, Carnegie Mellon University, Universita di Torino, 1988.

Stefano Berardi. *Type Dependence and Constructive Mathematics*. PhD thesis, Dipartimento di Informatica, Torino, Italy, 1990.

Marc Bezem, Jan Willem Klop, and Roel de Vrijer. *Term Rewriting Systems*. Cambridge University Press, 2003.

Alonzo Church. A Set of Postulates for the Foundation of Logic. *Annals of Mathematics*, pages 346–366, 1932.

Alonzo Church. A Formulation of the Simple Theory of Types. *The journal of symbolic logic*, 5(2):56–68, 1940.

Alonzo Church. *The Calculi of Lambda-Conversion*. Number 6. Princeton University Press, 1941.

Alonzo Church and J. Barkley Rosser. Some Properties of Conversion. *Transactions of the American Mathematical Society*, 39(3):472–482, 1936.

Thierry Coquand. A New Paradox in Type Theory. In *Studies in Logic and the Foundations of Mathematics*, volume 134, pages 555–570. Elsevier, 1995.

Thierry Coquand and Hugo Herbelin. A-Translation and Looping Combinators in Pure Type Systems. *Journal of Functional Programming*, 4(1):77–88, 1994.

Robin O. Gandy. An early proof of normalization by A.M. Turing. In *To H.B. Curry: essays on combinatory logic, lambda calculus and formalism*, volume 267, pages 453–455. Academic Press, 1980.

Herman Geuvers and Mark-Jan Nederhof. Modular proof of strong normalization for the calculus of constructions. *Journal of Functional Programming*, 1(2):155–189, 1991.

Herman Geuvers and Joep Verkoelen. On fixed point and looping combinators in type theory. 2009.

Herman Geuvers and Joep Verkoelen. The non-typability of some fixed-point combinators in Pure Type Systems. 2015.

Jean-Yves Girard. *Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur*. PhD thesis, Éditeur inconnu, 1972.

Inge Li Gørtz, Signe Reuss, and Morten Heine Sørensen. Strong Normalization from Weak Normalization by Translation into the Lambda-I-Calculus. *Higher-Order and Symbolic Computation*, 16(3):253–285, 2003.

Robert Harper, Furio Honsell, and Gordon Plotkin. A Framework for Defining Logics. *Journal of the ACM*, 40(1):143–184, 1993.

Gérard Huet. Residual Theory in λ-Calculus: A Formal Development. *Journal of Functional Programming*, 4(3):371–394, 1994.

Fairouz Kamareddine, Twan Laan, and Rob Nederpelt. *A Modern Perspective on Type Theory: From its Origins until Today*, volume 29 of *Applied Logic Series*. Springer, 2004.

Jan Willem Klop, Vincent Van Oostrom, and Femke Van Raamsdonk. Combinatory reduction systems: introduction and survey. *Theoretical computer science*, 121(1-2):279–308, 1993.

Jean-Jacques Lévy. *Réductions Correctes et Optimales dans le Lambda-Calcul*. PhD thesis, L'Université Paris VII, 1978.

Ralph Loader. Normalization by Calculation. 1995.

Zhaohui Luo. *An extended calculus of constructions*. PhD thesis, University of Edinburgh, 1990.

Nathan Mull. Strong Normalization from Weak Normalization in Non-Dependent Pure Type Systems via Thunkification. Research Report, `https://nmmull.github.io/thunk.pdf`, 2022.

Rob Nederpelt and Herman Geuvers. *Type Theory and Formal Proof: An Introduction*. Cambridge University Press, 2014.

nLab. concept with an attitude, 2023. URL https://ncatlab.org/nlab/show/concept+with+an+attitude.

John C Reynolds. Polymorphism is not Set-Theoretic. In *International Symposium on Semantics of Data Types*, pages 145–156. Springer, 1984.

Cody Roux and Floris van Doorn. The Structural Theory of Pure Type Systems. In *Rewriting and Typed Lambda Calculi*, pages 364–378. Springer, 2014.

Bertrand Russell and Alfred North Whitehead. *Principia Mathematica*, volume 1-3. Cambridge University Press, 1910, 1912, 1913.

Luis Elpidio Sanchis. Functionals Defined by Recursion. *Notre Dame Journal of Formal Logic*, 8(3):161–174, 1967.

Morten Heine Sørensen. Strong Normalization from Weak Normalization in Typedλ-Calculi. *Information and Computation*, 133(1):35–71, 1997.

William W. Tait. Intensional Interpretations of Functionals of Finite Type I. *The journal of symbolic logic*, 32(2):198–212, 1967.

Jan Terlouw. Een nadere bewijstheoretische analyse van GSTT's. Technical report, Department of Computer Science, University of Nijmege, 1989.

Femke van Raamsdonk, Paula Severi, Morten Heine B Sørensen, and Hongwei Xi. Perpetual Reductions in λ-Calculus. *Information and Computation*, 149(2):173–225, 1999.

Hongwei Xi. On Weak and Strong Normalisations. Technical report, Carnegie Mellon University, Department of Mathematics, 1996.

Hongwei Xi. Weak and Strong Beta Normalisations in Typed Lambda-Calculi. In *Proceedings of Typed Lambda Calculi and Applications*, volume 1210 of *Lecture Notes in Computer Science*, pages 390–404. Springer, 1997.

Hongwei Xi. Development Separation in Lambda-Calculus. *Electronic Notes in Theoretical Computer Science*, 143:207–221, 2006.