Strong Normalization from Weak Normalization in Non-Dependent Pure Type Systems via Thunkification

Nathan Mull

October 13, 2022

Abstract

In this report, I present a generalization of Xi's thunkification translation [14] to a class of non-dependent pure type systems, in analogy with the generalization of Sørensen's CPS translation [12] by Barthe *et al.* [3]. The primary benefit of this result, as with Xi's original translation, is that its proof is quite a bit simpler than its CPS-based counterpart. As a further simplification, I also present the class of *tiered* pure type systems, which are concretely specified, and so simpler to work with than the systems considered by Barthe *et al.* and others, but are equivalent with respect to questions regarding normalization.

1 Introduction

Pure type systems were introduced by Terlouw [13], Berardi [4], and later by Barendregt [1, 2], as a natural generalization of the lambda cube; they include the lambda cube as well as systems with more complex sort structure and product type formation. The study of pure type systems can in some sense be viewed as the study of how sort structure affects the meta-theoretic properties of a type system, especially because of the minimal set of type formers (*e.g.*, there are no Σ-types). One such meta-theoretic property, arguably one of the most important, is normalization. A type system is *weakly normalizing* if every typable term has a normal form and is *strongly normalizing* if no typable term appears in an infinite reduction sequence. Girard [7] demonstrated that sort structure can have a non-trivial effect on the normalization behavior of a type system by showing that the pure type system λU is not strongly normalizing, the import being that circularity in the sort structure of a pure type system is not a necessary condition for non-normalization. This leaves open the problem of understanding the interplay of sort structure and normalization. One particular open question that has persisted is the Baredregt-Geuvers-Klop conjecture, which states that weak normalization implies strong normalization for all pure type systems.

The primary technique for proving strong normalization from weak normalization passes through some form of the λI-calculus, where it readily follows from a conservation theorem along the lines of the one proved by Church and Rosser for the untyped lambda calculus [6]. The argument is roughly as follows: Suppose a system λS is weakly normalizing. Define a translation from expressions to λI-expressions in λS which preserve infinite reduction paths. The translated terms are weakly normalizing and, hence, are strongly normalizing by the conservation theorem. And since the translation preserves infinite reduction paths, the untranslated terms are themselves strongly normalizing.

This technique was originally used by Sørensen [12] via continuation-passing (CPS), and Xi [14] subsequently presented an alternative proof via thunkification that is arguably simpler; rather than passing around continuations, expressions are thunked and the unit term which is passed along to evaluate thunks is padded with sub-expression information necessary to translate to λI-expressions. Barthe *et al.* [3] generalized Sørensen's result to a class of non-dependent pure type systems which includes non-normalizing systems like λU. This report contains the analogous generalization for Xi's thunkification translation, with the hope of providing a simpler, more approachable proof of the main result of the Barthe *et al.* that may present opportunities for further improvements.

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 behavior if we want to derive the same result as the one by Barthe *et al.* Tiered pure type systems can be viewed as the *atoms* of persistent stratified systems, *i.e.*, the persistent stratified systems are disjoint unions of tiered systems.

In what follows I present some preliminary material, which includes some exposition on tiered systems. I then present the generalized thunkification translation in two parts: one part for the type-level translation and one part for term-level translation. Finally, I show how to derive strong normalization from weak normalization with this translation.

2 Preliminaries

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. We use s and t as a meta-variable for sorts.[1]

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 $\bigcup_{s \in \mathcal{S}} \mathsf{V}_s$. We use x, y, and z as meta-variables for expression variables. The choice to annotate variables with sorts is one of convenience. The annotations

[1]For any subsequent meta-variables, we also allow 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. We will try to be as clear as possible when distinguishing between these two cases of notation.

can be dropped for the systems we 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

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

We use M, N, P, A, B, and C as meta-variables for expressions. Free variables, bound variables, α-congruence, β-reduction, substitution, *etc.* are defined as usual (see, for example, Barendregt's presentation [2]). Substitution of x with N in M is denoted $M[N/x]$.

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. We call these statements appearing in proto-contexts **declarations**. We use Γ, Δ, 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 $(x : A)$ we write $(x : A) \in \Gamma$ if that statement appears in Γ. We define the subset relation $\Gamma \subset \Delta$ for contexts Γ and Δ analogously.

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.[2]

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. We say that 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 fresh variables ${}^{s}x$*

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

- ***Weakening.*** *For a fresh variable ${}^{s}x$*

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

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

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

[2]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').

- ***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 affect the statement of the generation lemma (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. We also say that M is a term and A is a type. 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.

2.1 Meta-Theory

We 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 ([2, 3, 8], 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
$$\Gamma \vdash \Pi^s x^B.\ C : s' \qquad \text{and} \qquad \Gamma, {}^s x : B \vdash M : C$$
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*
$$\Gamma, x : A, \Delta \vdash M : B \qquad \text{and} \qquad \Gamma \vdash N : A$$
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. $\Gamma \nvdash s : A$
2. $\Gamma \nvdash x : s$
3. $\Gamma \nvdash AB : s$
4. $\Gamma \nvdash \lambda x^A.\ B : s$.

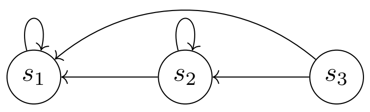

Figure 1: A visual representation of the system λU, where an arrow (s_i, s_j) indicates the presence of the rule (s_i, s_j, s_j) (axioms are not represented in the graph except in the ordered the nodes are presented).

2.2 Tiered Pure Type Systems

General pure type systems are notoriously difficult to work with so it is typical to consider a class of pure type systems satisfying a collection of properties, *e.g.*, functionality, persistence, and stratification, as defined below. Here I choose to work with an 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*

$$\begin{aligned}
\mathcal{S} &= \{s_i \mid i \in [n]\} \\
\mathcal{A} &= \{(s_i, s_{i+1}) \mid i \in [n-1]\} \\
\mathcal{R} &\subset \{(s_1, s_1, s_1)\} \cup \{(s, s', s') \mid (s, s') \in \mathcal{S} \times \mathcal{S}\}
\end{aligned}$$

From this point forward, I will 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;
- the 2-tiered systems are exactly the lambda cube;
- the n-tiered systems and are considered in passing by Barthes *et al.* (Remark 2.39, [3]). They include natural subsystems of ECC^n (as defined in [10]) with only the two-sorted rules.

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. This is likely a folklore result, as I could not find a reference for it, so I have included the proof here. First, some standard definitions, along with a couple definitions taken from Barthe *et al.* [3] for their definition of generalized non-dependent systems. Note that we have already seen the definition of functional pure type systems (Definition 2) in the previous section.

Definition 5. *A pure type system is **persistent** if it is functional and*

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

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

Definition 6. *A pure type system is **weakly stratified** if there is no infinite sequence of sorts $s, s', s'', \ldots$ such that*

$$s < s' < s'' < \ldots \qquad \textit{or} \qquad s > s' > s'' > \ldots$$[3]

In order to state the following equivalence, we work in the structural theory of pure type systems of Roux and van Doorn [11] (albeit, not the particularly interesting part of it).

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

$$\begin{aligned}
\mathcal{S}_{\lambda S \sqcup \lambda S'} &\triangleq \mathcal{S}_{\lambda S} \sqcup \mathcal{S}_{\lambda S'} \\
\mathcal{A}_{\lambda S \sqcup \lambda S'} &\triangleq \mathcal{A}_{\lambda S} \sqcup \mathcal{A}_{\lambda S'} \\
\mathcal{R}_{\lambda S \sqcup \lambda S'} &\triangleq \mathcal{R}_{\lambda S} \sqcup \mathcal{R}_{\lambda S'}
\end{aligned}$$

Lemma 8. *A pure type system is persistent and weakly stratified 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 weakly stratified, 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 weakly stratified and let T denote the set of top-sorts in $\mathcal{S}$. Considered the T-indexed partition $\{\mathcal{S}_t\}_{t \in T}$ of $\mathcal{S}$ where $\mathcal{S}_t = \{s \mid s \leq t\}$. We say a **chain** from s to s' is a sequence of sorts $(\upsilon_1, \ldots, \upsilon_k)$ such that $\upsilon_1 = s$ and $\upsilon_k = s'$ and $(\upsilon_i, \upsilon_{i+1}) \in \mathcal{A}$ for each i in $[k-1]$. Persistence ensures that each set in this partition is totally ordered by $\leq_{\mathcal{A}}$. In particular, it is possible to show that there is at most one chain ending at t of any length n contained in a set $\mathcal{S}_t$ for a top-sort t. This also implies that each set in the partition is disjoint. Stratification ensures that each set is finite. Finally note that the partition covers all of $\mathcal{S}$. If s is not in $\mathcal{S}_t$ for some top-sort t, then since s is not a top-sort, there is some other sort s' such that $(s, s') \in \mathcal{A}$ and s' is not in any set of the partition. This process can iterated to build an infinite ascending sequence of sorts. So $\{\mathcal{S}_t\}_{t \in T}$ is in fact a partition.

Let λS_t denote the pure system specified by

$$\begin{aligned}
\mathcal{S}_{\lambda S_t} &\triangleq \mathcal{S}_t \\
\mathcal{A}_{\lambda S_t} &\triangleq \mathcal{A}_{\lambda S} \cap (\mathcal{S}_t \times \mathcal{S}_t) \\
\mathcal{R}_{\lambda S_t} &\triangleq \mathcal{R}_{\lambda S} \cap (\mathcal{S}_t \times \mathcal{S}_t \times \mathcal{S}_t)
\end{aligned}$$

The axioms and rules of each system are clearly pairwise disjoint. They also cover all axioms and rules of λS. For suppose that (s, s') is an axiom such that $s \in \mathcal{S}_t$ and $s' \in \mathcal{S}_{t'}$ for distinct top-sorts t and t'. Since s is not a top-sort, there must be some other sort s'' in $\mathcal{S}_t$ such that $(s, s'') \in \mathcal{A}$. Then persistence implies that $s' = s''$, contradicting disjointness. The same kind of argument applies for the rules. Finally, the fact that each

[3]This is a weaker notion of stratification than the one given by Barthe et al. [3], in part because it is decoupled from the notion of non-dependence, and in part because their definition does not account for infinite descending sequences of sorts.

$\mathcal{S}_t$ is totally ordered with respect to $\leq_{\mathcal{A}}$ and is finite implies that each system λS_t is tiered. Therefore, we can view λS as the system $\bigsqcup_{t \in T} \lambda S_t$. Formally, they are isomorphic pure type systems.[4] $\square$

This fact can be easily lifted to generalized non-dependent systems.

Definition 8. *A pure type system λS is* ***generalized non-dependent*** *if it is persistent and weakly stratified and its rules are non-dependent, i.e., if $(s, s') \in \mathcal{R}_{\lambda S}$ then $s \geq s'$. A tiered pure type system λS is* ***non-dependent*** *its rules are non-dependent.*

Corollary 1. *A pure type system is generalized non-dependent if and only if it is the disjoint union of non-dependent tiered pure type systems.*

Roux and van Doorn [11] 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 persistent, weakly stratified (*e.g.*, generalized non-dependent) pure type systems, it suffices to consider tiered 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 expression and classifying expressions according to their degree. This result is due to Berardi [5], and the presentation here roughly follows the same course.

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

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

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, a couple meta-theoretic lemmas specific to the systems we will be considering. The first contains some useful facts about degree. See the presentation by Barendregt [2] for proofs in the 2-tiered case.

[4]The definition of a pure type system homomorphism is as one might expect, see [11] for more details.

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* $\deg A = \deg B$.

And last, the main lemma about non-dependent systems. One of the tricky aspects of working with dependencies is that they can introduce sub-expressions of a lower degree than the expression itself. This makes it very difficult to reverse induct on degree, which is an important proof technique for these systems. We won't need the full version of this lemma, just the simplified version 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 $j >$* $\deg(A)$.

3 The Generalized Thunkification Translation

Xi's thunkification translation [14] was introduced as a simpler approach to deriving strong normalization from weak normalization in typed lambda calculi, as compared to the CPS translation of Sørensen [12]. Roughly speaking, rather than passing continuations, Xi's translation pervasively thunkifies expressions, using uninterpreted padding functions to store additional sub-expression information in the evaluation of thunkified expressions. This allows for expressions to be mapped into the λI-calculus, where strong normalization is more readily proved from weak normalization. Sørensen's translation was subsequently generalized by Barthe *et al.* to generalized non-dependent, clean, negatable pure type systems [3]. In this section, we extend Xi's translation by analogy. This yields a simple alternative proof of the main result of Barthe *et al.*

Without loss of generality, we work with a fixed n-tiered pure type system λS. We present two families of translations, $\{\rho_i\}_{i\in[n]}$ for types and $\{\tau_i\}_{i\in[n]}$ for terms.

3.1 The Type-Level Translation

For the type-level translation, we need a distinguished unit type $\bot_i$ for each sort s_i. This is the type of the value passed to a thunkified term to evaluate it. In the case of s_1, this requires an additional type variable in the context.

Definition 10. *For $i \in [2 \ldots n]$, let $\bot_i$ denote s_{i-1} and let $\bot_1$ be a distinguished variable. Also let Δ_1 and Δ_2 denote the context $(\bot_1 : s_1)$ and let Δ_i denote the empty context in all other cases.*[5]

[5]We define Δ_2 in this way because we want that in all contexts Δ_j where $j \geq 2$, the expression $\bot_{j-1}$ is derivable. This will play a similar role to the variable I included in contexts by Barthe *et al.* (Definition 3.2, [3]).

The family of translations $\{\rho_i\}_{i\in[n]}$ gives the types of thunkified terms. This means pervasively replacing each type A where $\deg(A) = i$ with its corresponding thunkified form, *i.e.*, $\bot_i \to A$. For notational convenience, let T_j denote the set $\{M \in \mathsf{T} \mid \deg(M) = j\}$ and let $\mathsf{T}_{\geq j}$ denote the set $\{M \in \mathsf{T} \mid \deg(M) \geq j\}$.

Definition 11. *For each index i, let the functions $\rho_i : \mathsf{T}_{\geq i} \to \mathsf{T}$ and $\rho_i' : \mathsf{T}_{\geq i} \to \mathsf{T}$ be given simultaneously as follows.*

$$\begin{aligned}
\rho_i(s_j) &\triangleq s_j \quad \text{(where } j \geq i) \\
\rho_i(x) &\triangleq x \\
\rho_i(\Pi x^A.\ B) &\triangleq \Pi x^{\rho_i'(A)}.\ \rho_i(B) \\
\rho_i(\lambda x^A.\ M) &\triangleq \lambda x^{\rho_i(A)}.\ \rho_i(M) \\
\rho_i(MN) &\triangleq \rho_i(M)\rho_i(N) \\
\rho_i'(A) &\triangleq \begin{cases} \bot_i \to \rho_i(A) & \deg(A) = i \\ \rho_i(A) & \text{otherwise} \end{cases}
\end{aligned}$$

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

Lemma 12. *(ρ_i and ρ_i' commute with substitution) For index i, variable ${}^{s_j}x$ and expressions M and N where $\deg(N) = j - 1$ the following hold.*

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

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

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

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

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

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

Π-Expression. If M is of the form $\Pi y^A.\ B$ then

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

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

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

The case that M is an application is similar. □

Lemma 13. *(ρ_i and ρ'_i preserve beta-reductions) For index i and derivable expressions M and N, if $M \to_\beta N$ then $\rho_i(M) \to_\beta \rho_i(N)$ and $\rho'_i(M) \to_\beta \rho'_i(N)$. In particular, if $M =_\beta N$ then $\rho_i(M) =_\beta \rho_i(N)$ and $\rho'_i(M) =_\beta \rho'_i(N)$.*

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_i and ρ'_i 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'_i given that it holds inductively for ρ_i. So we proceed by induction in the case of ρ_i. In the case of a redex,

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

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

Next, we prove that the type-level translation preserves typability. This will ensure 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. We will additionally need to translate contexts in derivations, so we extend ρ'_i to contexts as follows.

$$\begin{aligned}\rho'_i(\varnothing) &\triangleq \Delta_i \\ \rho'_i(\Gamma, x : A) &\triangleq \begin{cases}\rho'_i(\Gamma), x : \rho'_i(A) & \deg(A) \geq i \\ \rho'_i(\Gamma) & \text{otherwise}\end{cases}\end{aligned}$$

Note that, for convenience, we translate the empty context to be Δ_i. We also need to restrict our focus to systems for which it is possible define types of the form $\bot_i \to A$. This justifies the following definition.

Definition 12. *Let λS be a tiered pure type system.*

- *A sort s_j is **relevant** if $(s_i, s_j) \in \mathcal{R}_{\lambda S}$ for some sort s_i.*
- *A sort s_j is **negatable** if $(s_j, s_j) \in \mathcal{R}_{\lambda S}$.*
- *λS is **negatable** if all relevant sorts are negatable.*

Lemma 14. *(ρ_i and ρ'_i preserve typability) Let λS be a non-dependent tiered pure type systems. For index i, context Γ, and expressions M and A, if s_i is negatable then the following hold.*

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

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

$$\frac{\rho'_i(\Gamma) \vdash \bot_i : s_i \qquad \rho'_i(\Gamma) \vdash \rho_i(A) : s_i}{\rho'_i(\Gamma) \vdash \bot_i \to \rho_i(A) : s_i}$$

Note that this judgment is derivable since s_i is negatable.

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 $\rho'_i(A) = \rho_i(A)$ since $\deg(A) > i$. So by the inductive hypothesis, we have

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

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, $\rho'_i(\Gamma) \vdash \rho_i(M) : \rho_i(A)$, so if $\deg(B) \leq i$, then we're done. Otherwise, we have

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

where the right antecedent judgment also comes 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

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

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

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

<u>Abstraction.</u> 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 since λS is non-dependent, we have $\deg(A) \geq \deg(B) > i$. In particular, $\rho'_i(A) = \rho_i(A)$. Therefore, we have

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

<u>Application.</u> Suppose the last inference is of the form

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

Since λS is non-dependent, $\deg A \geq \deg B > \deg M \geq i$ and $\rho'_i(A) = \rho_i(A)$. So by the inductive hypothesis we have

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

where $\rho_i(B)[\rho_i(N)/x] = \rho_i(B[N/x])$ by Lemma 12.
<u>Conversion.</u> 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

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

where $\rho_i(A) =_\beta \rho_i(B)$ by Lemma 13. □

3.2 The Term-Level Translation

Next we define the translation of terms. In addition to the extra type information in Δ_i, we also need what amount to uninterpreted padding variables in the context for collecting expressions which are arguments to λ-terms. These will be used to ensure that translated terms are λI-terms in the appropriate sense (see Definition 17). The padding variables are made to be polymorphic when possible, but the type may need to be more strictly specified when this is not possible. To more carefully make this distinction, we need the following definition.

Definition 13. *Let λS be a tiered pure type system.*

- *A rule (s_i, s_j) is **generalizable** if $(s_{i+1}, s_j) \in \mathcal{R}_{\lambda S}$.*
- *A rules (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 $k > j$.*
- *λS is **clean** if all its rules are generalizable or harmless.*

Generalizability corresponds the inclusion of rules for typing polymorphic padding variables, and harmlessness ensures that, when those rules are not included, the padding variables never become untypable because a variable it depends on leaves the context. See Barthe et al. (Remark 4.1, [3]) for more exposition on the motivations of this definition. We collect the padding variables in a single context that is dependent on the subject of the judgment we are translating.

Definition 14. *For each index i, define $\Upsilon_{i,M}$ with respect to the structure of M as follows.*

$$\begin{aligned}
\Upsilon_{i,s_j} &\triangleq \varnothing \\
\Upsilon_{i,x} &\triangleq \varnothing \\
\Upsilon_{i,\Pi x^A.\ B} &\triangleq \Upsilon_{i,A}, \Upsilon_{i,B} \\
\Upsilon_{i,\lambda x^A.\ M} &\triangleq p_x : \alpha_{i,A}, \Upsilon_{i,M} \\
\Upsilon_{i,MN} &\triangleq \Upsilon_{i,M}, \Upsilon_{i,N}
\end{aligned}$$

where

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

We will write $\langle M, N\rangle_{\rho'_i(A)}$ for both $p_x \rho'_i(A) MN$ and $p_x MN$ (*i.e.*, for each case of $\alpha_{i,A}$) when there is no fear of ambiguity.

The context of padding variables appears *after* the translation of the given context, as it may depend on some of the declarations in it, so to translate abstraction or product type formations, we need some declarations to commute with the context of padding variables. The following ensures that this is possible.

Lemma 15. *If (s_j, s_i) or (s_j, s_{i-1}) are in $\mathcal{R}_{\lambda S}$, then no undistinguished variables (i.e., ignoring $\bot_i$) of degree $j-1$ (i.e., of the form ${}^{s_j}x$) can appear free in $\Upsilon_{i,M}$ for any term 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 $\lambda x^A.\ N$ and $(\deg(A), i)$ is harmless. If ${}^{s_k}x$ appears free in $\rho'_i(A)$, then it must also appear free in A (this is straightforward to verify), and so by Lemma 11, it must be that $k > \deg(A)$. But by harmlessness, $j \leq \deg(A)$, so $k \neq j$. $\square$

We now define the term-level translation. The translation is fairly simple, with the exception of two cases which are discussed in remark following the definition.

Definition 15. *For each index i, define the functions $\tau_i : \mathsf{T}_{i-1} \to \mathsf{T}$ and $\tau'_i : \mathsf{T}_{i-1} \to \mathsf{T}$ simultaneously as follows.*

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

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

We are taking advantage of a couple representation tricks in this translation. For the case of sorts, we use the fact that $\bot_i = s_{i-1}$ and for the case of Π-types, we use the fact that $\Delta_i \vdash \bot_{i-1} : \bot_i$ if $i \geq 2$. These tricks are used to maintain one of the most important features of this translation, *i.e.*, that $\tau'(M)$ is linear in $\bullet_i$ on terms of degree $i-1$.

Lemma 16. *For any term M where $\deg(M) = i-1$, the variable $\bullet_i$ appears free exactly once in $\tau(M)$.*

Finally, we show that the term-level translations preserve typability.

Lemma 17. *(τ_i and τ'_i preserve typability) Let λS be a clean tiered pure type system and suppose that s_i is negatable. For any context Γ and expressions M and A, if $\Gamma \vdash M : A$ and $\Gamma \vdash A : s_i$, then*

1. $\rho'_i(\Gamma), \Upsilon_{i,M}, \bullet_i : \bot_i \vdash \tau_i(M) : \rho_i(A)$
2. $\rho'_i(\Gamma), \Upsilon_{i,M} \vdash \tau'_i(M) : \rho'_i(A)$

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

$$\rho'_i(\Gamma), \Upsilon_{i,M}, \bullet_i : \bot_i \vdash \tau_i(M) : \rho_i(A)$$

By Lemma 14 and thinning (Lemma 4) we can derive

$$\rho'_i(\Gamma), \Upsilon_{i,M} \vdash \bot_i \to \rho_i(A) : s_i$$

which means by abstraction we can derive

$$\rho'_i(\Gamma), \Upsilon_{i,M} \vdash \lambda \bullet_i^{\bot_i} .\ \tau_i(M) : \bot_i \to \rho_i(A)$$

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 $\Delta_i, \bullet_i : \bot_i \vdash \bullet_i : \bot_i$, noting that

$$\bot_i = s_{i-1} = \rho_i(s_{i-1})$$

Variable Introduction. Suppose the last derivation is of the form

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

By Lemma 14 and thinning (Lemma 4), we have the inference

$$\frac{\rho_i'(\Gamma), \Upsilon_{i,x} \vdash \rho_i'(A) : s_i}{\rho_i'(\Gamma), \Upsilon_{i,x}, x : \rho_i'(A) \vdash x : \rho_i'(A)}$$

Since $\Upsilon_{i,x} = \varnothing$, we can move it in the context freely and apply weakening to derive

$$\rho_i'(\Gamma), x : \rho_i'(A), \Upsilon_{i,x}, \bullet_i : \bot_i \vdash x : \rho_i'(A)$$

It is straightforward to then derive by application

$$\rho_i'(\Gamma), x : \rho_i'(A), \Upsilon_{i,x}, \bullet_i : \bot_i \vdash x\bullet_i : \rho_i(A)$$

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

$$\rho_i'(\Gamma), \Upsilon_{i,M}, \bullet_i : \bot_i \vdash \tau_i(M) : \rho_i(A)$$

directly by the inductive hypothesis. Otherwise, B is in the domain of ρ_i' and by the inductive hypothesis, Lemma 14, and thinning (Lemma 4), we have

$$\frac{\begin{array}{c}\rho_i'(\Gamma), \Upsilon_{i,M}, \bullet_i : \bot_i \vdash \tau_i(M) : \rho_i(A) \\ \rho_i'(\Gamma), \Upsilon_{i,M}, \bullet_i : \bot_i \vdash \rho_i'(B) : s_j\end{array}}{\rho_i'(\Gamma), \Upsilon_{i,M}, \bullet_i : \bot_i, x : \rho_i'(B) \vdash \tau_i(M) : \rho_i(A)}$$

And since $\bullet_i$ does not appear free in $\rho_i'(B)$ and nor do any of the variables in $\Upsilon_{i,M}$, by permutation (Lemma 5) we can also derive

$$\rho_i'(\Gamma), x : \rho_i'(B), \Upsilon_{i,M}, \bullet_i : \bot_i, \vdash \tau_i(M) : \rho_i(A)$$

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, we can derive

$$\rho_i'(\Gamma), \Upsilon_{i,A}, \bullet_i : \bot_i \vdash \tau_i'(A) : \bot_i \to s_{i-1}$$

and since $\vdash \bot_{i-1} : \bot_i$ and $\bot_{i-1}$ is derivable from Δ_i, we can derive by application

$$\rho_i'(\Gamma), \Upsilon_{i,A}, \bullet_i : \bot_i \vdash \tau_i'(A)\bot_{i-1} : s_{i-1}$$

which, with thinning, gives us the inference

$$\frac{\begin{array}{c}\rho_i'(\Gamma), \Upsilon_{i,A}, \Upsilon_{i,B}, \bullet_i : \bot_i \vdash \tau_i'(A)\bot_{i-1} : s_{i-1} \\ \rho_i'(\Gamma), \Upsilon_{i,A}, \Upsilon_{i,B}, \bullet_i : \bot_i \vdash \tau_i(B) : s_{i-1}\end{array}}{\rho_i'(\Gamma), \Upsilon_{i,A}, \Upsilon_{i,B}, \bullet_i : \bot_i \vdash \tau_i(A)'\bot_{i-1} \to \tau_i(B) : s_{i-1}}$$

Otherwise, A is in the domain of ρ_i, and by Lemma 15, the variable ${}^{s_j}x$ does not appear free in $\Upsilon_{i,B}$, so by permutation (Lemma 5), we can also derive

$$\rho_i'(\Gamma), \Upsilon_{i,B}, \bullet_i : \bot_i, {}^{s_j}x : \rho_i'(A) \vdash \tau_i(B) : s_{i-1}$$

from which we have the inference

$$\frac{\begin{array}{c}\rho_i'(\Gamma), \Upsilon_{i,A}, \Upsilon_{i,B}, \bullet_i : \bot_i \vdash \rho'(A) : s_j \\ \rho_i'(\Gamma), \Upsilon_{i,A}, \Upsilon_{i,B}, \bullet_i : \bot_i, {}^{s_j}x : \rho_i'(A), \vdash \tau_i(B) : s_k\end{array}}{\rho_i'(\Gamma), \Upsilon_{i,A}, \Upsilon_{i,B}, \bullet_i : \bot_i \vdash \Pi x^{\rho_i'(A)}.\ \tau_i(B) : s_k}$$

where the first antecedent judgment is attained by Lemma 14 and thinning.

Abstraction. Suppose the last inference is of the form

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

By the inductive hypothesis,

$$\rho_i'(\Gamma), x : \rho_i'(A), \Upsilon_{i,M}, \bullet_i : \bot_i \vdash \tau_i'(M) : \bot_i \to \rho_i(B)$$

Since $(\deg(A), i) \in \mathcal{R}_{\lambda S}$, it is either generalizable or harmless. In the first case, it is possible to derive

$$\Delta_i \vdash \Pi t^{s_{\deg(A)}}.\ t \to \bot_i \to \bot_i : s_i$$

Otherwise, by Lemma 14 a and couple additional steps (including a use of generation (Lemma 1)[6]), we can derive

$$\rho_i'(\Gamma) \vdash \rho_i'(A) \to \bot_i \to \bot_i : s_i$$

so in both cases we can thin the above judgment to

$$\rho_i'(\Gamma), x : \rho_i'(A), \Upsilon_{i,\lambda x^A.\ M}, \bullet_i : \bot_i \vdash \tau_i'(M) : \bot_i \to \rho_i(B)$$

Therefore, in a few steps we can derive

$$\rho_i'(\Gamma), x : \rho_i'(A), \Upsilon_{i,\lambda x^A.\ M}, \bullet_i : \bot_i \vdash \langle x, \bullet_i \rangle_{\rho_i'(A)} : \bot_i$$

which can be used in the application

[6]Alternatively, we can consider the system which includes $\Gamma \vdash A : s_{\deg(A)}$ as an antecedent judgment to abstraction, which is an equivalent system.

$$\frac{\rho'_i(\Gamma), x : \rho'_i(A), \Upsilon_{i,\lambda x^A.\ M}, \bullet_i : \bot_i \vdash \tau'_i(M) : \bot_i \to \rho_i(B) \qquad \rho'_i(\Gamma), x : \rho'_i(A), \Upsilon_{i,\lambda x^A.\ M}, \bullet_i : \bot_i \vdash \langle x, \bullet_i\rangle_{\rho'_i(A)} : \bot_i}{\rho'_i(\Gamma), x : \rho'_i(A), \Upsilon_{i,\lambda x^A.\ M}, \bullet_i : \bot_i \vdash \tau'_i(M)\langle x, \bullet_i\rangle_{\rho'_i(A)} : \rho_i(B)}$$

Finally, since x does not appear free in $\Upsilon_{i,\lambda x^A.\ M}$ (Lemma 15), we can also derive

$$\Delta_i, \rho'_i(\Gamma), \Upsilon_{i,\lambda x^A.\ M}, \bullet_i : \bot_i, x : \rho'_i(A) \vdash \tau'_i(M)\langle x, \bullet_i\rangle_{\rho'_i(A)} : \rho_i(B)$$

which can be used in the last abstraction

$$\frac{\rho'_i(\Gamma), \Upsilon_{i,\lambda x^A.\ M}, \bullet_i : \bot_i, x : \rho'_i(A) \vdash \tau'_i(M)\langle x, \bullet_i\rangle_{\rho'_i(A)} : \rho_i(B) \qquad \rho'_i(\Gamma), \Upsilon_{i,\lambda x^A.\ M}, \bullet_i : \bot_i \vdash \Pi x^{\rho'_i(A)}.\ \rho_i(B) : s_j}{\rho'_i(\Gamma), \Upsilon_{i,\lambda x^A.\ M}, \bullet_i : \bot_i \vdash \tau_i(\lambda x^A.\ M) : \Pi x^{\rho'_i(A)}.\ \rho_i(B)}$$

Application. Suppose the last inference is of the form

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

If $\deg N = i-1$, then by the inductive hypothesis, Lemma 14, and thinning we have

$$\frac{\rho'_i(\Gamma), \Upsilon_{i,M}, \Upsilon_{i,N}, \bullet_i : \bot_i \vdash \tau_i(M) : \Pi x^{\rho'_i(A)}.\ \rho_i(B) \qquad \rho'_i(\Gamma), \Upsilon_{i,M}, \Upsilon_{i,N}, \bullet_i : \bot_i \vdash \tau'_i(N) : \rho'_i(A)}{\rho'_i(\Gamma), \Upsilon_{i,MN}, \bullet_i : \bot_i \vdash \tau_i(M)\tau'_i(N) : \rho_i(B)[\tau'_i(N)/x]}$$

By Lemma 11, the variable x does not appear free in $\rho_i(B)$ in this case, so $\rho_i(B)[\tau'_i(N)/x] = \rho_i(B) = \rho_i(B[N/x])$. In the other case, we have $\deg N \geq i$ which implies N is in the domain of ρ_i, so we have

$$\frac{\rho'_i(\Gamma), \Upsilon_{i,M}, \Upsilon_{i,N}, \bullet_i : \bot_i \vdash \tau_i(M) : \Pi x^{\rho_i(A)}.\ \rho_i(B) \qquad \rho'_i(\Gamma), \Upsilon_{i,M}, \Upsilon_{i,N}, \bullet_i : \bot_i \vdash \rho_i(N) : \rho_i(A)}{\rho'_i(\Gamma), \Upsilon_{i,MN}, \bullet_i : \bot_i \vdash \tau_i(M)\rho_i(N) : \rho_i(B)[\rho_i(N)/x]}$$

and $\rho_i(B)[\rho_i(N)/x] = \rho_i(B[N/x])$ by Lemma 12.
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, Lemma 14, thinning, and the fact that ρ_i preserves β-equivalence (Lemma 13), we have

$$\frac{\rho'_i(\Gamma), \Upsilon_{i,M}, \bullet_i : \bot_i \vdash \tau_i(M) : \rho_i(A) \qquad \rho'_i(\Gamma), \Upsilon_{i,M}, \bullet_i : \bot_i \vdash \rho_i(B) : s_j}{\rho'_i(\Gamma), \Upsilon_{i,M}, \bullet_i : \bot_i \vdash \tau_i(M) : \rho_i(B)}$$

□

3.3 Strong Normalization from Weak Normalization

It remains to derive strong normalization from weak normalization for non-dependent clean negatable tiered systems. This is in essense a rewriting of Xi's proof in the framework of Barthe *et al.* by which, supposing our system is weakly normalizing, we prove by reverse induction on i that all terms in $\mathsf{T}_{\geq i}$ are strongly normalizing for every index i.

Definition 16. *A non-dependent tiered pure type system λS is **i-secure** if for all derivable expressions M, if $\deg(M) \geq i$, then M is strongly normalizing.*

The induction step splits into two cases, depending on whether the sort with the index in question is negatable or irrelevant. By negatablity of the system itself, these are the only two cases. The case of irrelevant sorts is simple and handled directly by Barthe *et al.*

Lemma 18. *(Lemma 5.19, [3]) 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. In particular, λS is $(i-1)$-secure.*

The second case is aided by two lemmas alluded to in the introduction. First, it must be that our term-level translation maps expressions into an appropriate generalization of the λI-calculus. This easy to verify by looking at the way λ-expressions appear in the translation.

Definition 17. *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 19. *Let λS a non-dependent negatable tiered pure type system. For any derivable expression M with $\deg(M) = i-1$, it follows that $\tau_i(M)$ is an I-expression at level $i-1$.*

As proved by Barthe *et al.* this calculus has the same sort of conservation theorem as one proved by Church and Rosser [6] for the standard λI-calculus.

Lemma 20. *(Lemma 5.16, [3]) Let λS be a non-dependent tiered pure type system that is i-secure. Then for every I-expression M at level $i-1$, if M is weakly normalizing then it is strongly normalizing.*

Next, it must be that the term-level translation preserves infinite reductions paths. The proof of this will occupy the following subsection, but taken together we have the implications

$$\begin{aligned} \tau_i(M) \text{ is weakly normalizing} &\Rightarrow \tau_i(M) \text{ is strongly normalizing} \\ &\Rightarrow M \text{ is strongly normalizing} \end{aligned}$$

which we package as a single lemma.

Lemma 21. *Let λS be a non-dependent, clean, negatable tiered pure type systems. For any derivable expression M with $\deg(M) = i - 1$, if $\tau_i'(M)$ is weakly normalizing then M is strongly normalizing.*

Before proving path preservation, the missing component of the above lemma, we can present the proof of the final result, which we have been outlining thus far.

Theorem 1. *Let λS be a non-dependent, clean, negatable n-tiered pure type system. If λS is weakly normalizing then λS is strongly normalizing.*

Proof. We prove by reverse induction on i (from n to 0) 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. By Lemma 18, if s_k is irrelevant, then λS is $(k-1)$-secure. Otherwise, since λS is negatable, s_k must be negatable. Let M be a term in $\mathsf{T}_{\geq k-1}$ derivable in λS. If $\deg(M) \geq k$, then M is strongly normalizing by k-security. Otherwise, by Lemma 17, $\tau_k(M)$ is derivable and so is weakly normalizing. So by Lemma 21, M is strongly normalizing, and consequently, λS is $(k-1)$-secure. $\square$

Preserving Infinite Reduction Paths

To prove that τ_i preserves infinite reduction paths, we show that the length of the longest reduction from path starting at an expression M is at most the length of the longest path from $\tau_i(M)$.

Definition 18. *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).*

First, we to know how τ_i interacts with substitution.

Lemma 22. *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 $\tau_i(M[N/{}^{s_k}x]) \twoheadleftarrow_\beta \tau_i(M)[\tau_i'(N)/{}^{s_k}x]$.*
2. *If $k > i$, then $\tau_i(M[N/{}^{s_k}x]) = \tau_i(M)[\rho_i(N)/{}^{s_k}x]$.*

Proof. We prove item 1 by induction on the structure of M. The case in which M is a sort is straightforward.
Variable. If $M = {}^{s_i}x$ then

$$\tau_i({}^{s_i}x[N/{}^{s_i}x]) = \tau_i(N) \twoheadleftarrow_\beta \tau_i'(N)\bullet_i = \tau_i({}^{s_i}x)[\tau_i'(N)/{}^{s_i}x]$$

Otherwise, M is of the form ${}^{s_i}y$ with $y \neq x$ and

$$\tau_i({}^{s_i}y[N/{}^{s_k}x]) = \tau_i({}^{s_i}y) = \tau_i({}^{s_i}y)[\tau_i'(N)/{}^{s_k}x]$$

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

$$\begin{aligned}\tau_i((\Pi y^A.\ B)[N/x]) &= \tau_i(A[N/x])\bot_{i-1} \to \tau_i(B[N/x]) \\ &\twoheadleftarrow_\beta \tau_i(A\bot_{i-1})[\tau_i'(N)/x] \to \tau_i(B)[\tau_i'(N)/x] \\ &= \tau_i(\Pi y^A.\ B)[\tau_i'(N)/x]\end{aligned}$$

Otherwise, note that if $\deg(A) > i - 1$, then by Lemma 11, ${}^{s_i}x$ does not appear free in A and the free variables of $\rho'_i(A)$ are the same as those of A, so

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

The cases that M is a λ-terms or an application are similar. Furthermore the proof of item 2 is similar as well. □

Now for the upper bound. We'll need a generalization of a basic result from the theory of the untyped lambda calculus (see [14]).

Lemma 23. *For any expressions A, M, $N_1, \ldots, N_k$ and variable x,*

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

Lemma 24. *Let λS be non-dependent tiered pure type system where s_i is negatable. For any expression A with* $\deg(A) \geq i$,

$$\mu(A) \leq \mu(\rho_i(A)) = \mu(\rho'_i(A))$$

Proof. By induction on the structure of A. For illustration, we consider the case that A is a Π-term.
$\underline{\Pi\text{-Term.}}$ Suppose that M is of the form $\Pi x^A.\ N$. By the inductive hypothesis, $\mu(A) \leq \mu(\rho_i(A)) = \mu(\rho'_i(A))$ and $\mu(B) \leq \mu(\rho_i(B))$. Therefore,

$$\begin{aligned}\mu(\Pi x^A.\ B) &= \mu(A) + \mu(B) \\ &\leq \mu(\rho'_i(A)) + \mu(\rho_i(B)) \\ &= \mu(\Pi x^{\rho'_i(A)}.\ \rho_i(B))\end{aligned}$$

□

Lemma 25. *Let λS be a weakly normalizing non-dependent tiered pure type system where s_i is negatable. For any derivable expression M with* $\deg(M) = i - 1$, *we have*

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

Proof. By induction on $\mu(\tau_i(M))$ and the structure of M, lexicographically ordered. The cases in which M is a sort, a variable, or a Π-type are straightforward.
$\underline{\lambda\text{-Expression.}}$ Suppose M is of the form $\lambda x^A.\ N$. By Lemma 24, $\mu(A) \leq \mu(\rho'_i(A))$ and by the inductive hypothesis, $\mu(N) \leq \mu(\tau_i(N)) = \mu(\tau'_i(N))$, so

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

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

Application. We consider two cases. First suppose that M is of the form $xN_1 \ldots, N_k$ where $k \geq 0$. In what follows, let

$$\pi_i(M) = \begin{cases} \tau_i'(M) & \deg M = i-1 \\ \rho_i(M) & \deg M \geq i \end{cases}$$

Then

$$\begin{aligned} \mu(xN_1 \ldots N_k) &= \sum_{j=1}^{k} \mu(N_j) \\ &\leq \sum_{j=1}^{k} \mu(\pi_i(N_j)) \\ &= \mu(\tau_i'(xN_1 \ldots N_k)) \end{aligned}$$

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

$$\begin{aligned} &\tau_i((\lambda x^A.\ P)N_1 \ldots N_k) \\ &= (\lambda x^{\rho_i'(A)}.\ \tau_i'(P)\langle x, \bullet_i\rangle_{\rho_i'(A)})\pi_i(N_1), \ldots \pi_i(N_k) \\ &\twoheadrightarrow_\beta (\lambda x^{\mathsf{nf}(\rho_i'(A))}.\ \tau_i'(P)\langle x, \bullet_i\rangle_{\rho_i'(A)})\pi_i(N_1), \ldots \pi_i(N_k) \\ &\rightarrow_\beta \tau_i'(P)[\pi_i(N_1)/x]\langle \pi_i(N_1), \bullet_i\rangle_{\rho_i'(A)}\pi_i(N_2) \ldots \pi_i(N_k) \\ &\twoheadrightarrow_\beta \tau_i'(P[N_1/x])\langle \pi_i(N_1), \bullet_i\rangle_{\rho_i'(A)}\pi_i(N_2) \ldots \pi_i(N_k) \\ &\twoheadrightarrow_\beta \tau_i'(P[N_1/x])\langle \mathsf{nf}(\pi_i(N_1)), \bullet_i\rangle_{\rho_i'(A)}\pi_i(N_2) \ldots \pi_i(N_k) \\ &\rightarrow_\beta \tau_i(P[N_1/x])[\langle \mathsf{nf}(\pi_i(N_1)), \bullet_i\rangle_{\rho_i'(A)}/\bullet_i]\pi_i(N_2) \ldots \pi_i(N_k) \end{aligned}$$

where $\mathsf{nf}(M)$ denotes the normal form of M, which is guaranteed to exist by the assumption of weak normalization. Note that the fifth line follows from Lemma 22. This reduction has length at least

$$\begin{aligned} &2 + \mu(\rho_i'(A)) \\ &\quad + \mu(\pi_i(N)) \\ &\quad + \mu(\tau_i(P[N_1/x][\langle \mathsf{nf}(N_1), \bullet_i\rangle_{\rho_i'(A)}/\bullet_i])\pi_i(N_2) \ldots \pi_i(N_k)) \end{aligned}$$

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

$$\begin{aligned} &\mu(\tau_i((P[N_1/x])N_2 \ldots N_k)) \\ &= \mu(\tau_i(P[N_1/x])\pi_i(N_2) \ldots \pi_i(N_k)) \\ &\leq \mu(\tau_i(P[N_1/x][\langle \mathsf{nf}(N_1), \bullet_i\rangle_{\rho_i'(A)}/\bullet_i])\pi_i(N_2) \ldots \pi_i(N_k)) \end{aligned}$$

since $\bullet_i$ can be replaced with $\langle \mathsf{nf}(N_1), \bullet_i\rangle_{\rho_i'(A)}$ in any reduction sequence starting at $\tau_i(P[N_1/x])\pi_i(N_2) \ldots \pi_i(N_k)$. So applying the inductive hypothesis, the above expression is lower bounded by

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

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

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

Finally, by Lemma 23, this implies $\mu(M) \leq \mu(\tau_i(M))$. $\square$

4 Conclusions

I've presented a proof that weak normalization implies strong normalization in non-dependent clean negatable tiered pure type systems via thunkification. 'Tiered' can be replaced with 'generalized non-dependent' by considering a disjoint unions of tiered systems. The obvious question that remains is whether it is possible to prove strong normalization from weak normalization for *all* non-dependent tiered systems. Removing cleanliness would seem to require a different form of padding gadget. One potential place to look for this is Loader's translation [9], which attempts to derive padding terms on the fly from a collection of basic padding terms. Regardless, my hope is that, in its simplicity, this proof based on thunkification is amenable to modifications and improvements.

References

[1] Henk Barendregt. Introduction to generalized type systems. *Journal of Functional Programming*, 1(2):125–154, 1991.

[2] Henk Barendregt. Lambda Calculi with Types. In *Handbook of Logic in Computer Science, Volume II*, pages 117–309. Oxford University Press, 1993.

[3] 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.

[4] 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.

[5] Stefano Berardi. *Type Dependence and Constructive Mathematics.* PhD thesis, Dipartimento di Informatica, Torino, Italy, 1990.

[6] Alonzo Church and J. Barkley Rosser. Some Properties of Conversion. *Transactions of the American Mathematical Society*, 39(3):472–482, 1936.

[7] Jean-Yves Girard. *Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur.* PhD thesis, Éditeur inconnu, 1972.

[8] 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.

[9] Ralph Loader. Normalization by Calculation. Strong normalization from weak normalization proof, 1995.

[10] Zhaohui Luo. *An extended calculus of constructions.* PhD thesis, University of Edinburgh, 1990.

[11] Cody Roux and Floris van Doorn. The Structural Theory of Pure Type Systems. In *Rewriting and Typed Lambda Calculi*, pages 364–378. Springer, 2014.

[12] Morten Heine Sørensen. Strong Normalization from Weak Normalization in Typedλ-Calculi. *Information and Computation*, 133(1):35–71, 1997.

[13] Jan Terlouw. Een nadere bewijstheoretische analyse van GSTT's. Technical report, Department of Computer Science, University of Nijmege, 1989.

[14] 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.