An Irrelevancy-Eliminating Translation of Pure Type Systems

Nathan Mull
University of Chicago, IL, USA

Abstract

I present an infinite-reduction-path-preserving typability-preserving translation of pure type systems which eliminates rules and sorts that are in some sense irrelevant with respect to normalization. This translation can be bootstrapped with existing results for the Barendregt-Geuvers-Klop conjecture, extending the conjecture to a larger class of systems. Performing this bootstrapping with the results of Barthe et al. [4] yields a new class of systems with dependent rules and non-negatable sorts for which the conjecture holds. To my knowledge, this is the first improvement in the state of the conjecture since the results of Roux and van Doorn [16] (which can be used for the same sort of bootstrapping argument) albeit a somewhat modest one; in essence, the translation eliminates clutter in the system that does not affect normalization. This work is done in the framework of *tiered* pure type systems, a simple class of persistent systems which is sufficient to study when concerned with questions about normalization.

2012 ACM Subject Classification Theory of computation → Type theory

Keywords and phrases pure type systems, normalization, reduction-path-preserving translations, Barendregt-Geuvers-Klop conjecture

Digital Object Identifier 10.4230/LIPIcs.TYPES.2022.7

Acknowledgements I'd like to thank Stuart Kurtz for many useful discussions as well as the reviewers for their incredibly helpful comments.

1 Introduction

The class of pure type systems [2, 3, 5, 8, 9, 18] was introduced as a natural generalization of the lambda cube which includes systems with more complex sort structure and product type formation rules. The study of pure type systems is primarily concerned with how this sort structure affects meta-theoretic properties (especially given the minimal collection of type formers). One such property is normalization: a type system is *weakly normalizing* if every typable term has a normal form and *strongly normalizing* if no typable term appears in an infinite reduction sequence.

Despite the fact that weak normalization is, of course, the weaker of the two properties, it is often sufficient for proving other important meta-theoretic properties, e.g., consistency and decidability of type checking in the presence of dependent types. Observations to this effect were made by Geuvers in his PhD thesis [9], where he also conjectured that weak normalization implies strong normalization for *all* pure type systems (Conjecture 8.1.12). This conjecture has come to be known as the *Barendregt-Geuvers-Klop conjecture*.[1]

Little progress has been made on this conjecture, in part because pure type systems in general are not always 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

[1] Sørensen submitted the conjecture by this name to the *TLCA List of Open Problems* [1]. He has also referred to it as the *Barendregt-Geuvers conjecture* [17]. Barendregt is noted by Barthe et al. [4] to have presented the conjecture at the *Second International Conference on Typed Lambda Calculi and Applications (1995)*, and Klop is the co-author of a preprint which refers to the Barendregt-Geuvers-Klop conjecture by name ([13], Conjecture 1.1).

basic syntactic ambiguation of the inference rules which allows for maximal freedom in sort structure. The resulting systems may fail to have the meta-theoretic properties one might expect (e.g., type unicity) so it is common to consider classes of systems which do maintain these properties. The state of the art of the conjecture is the result of Barthe et al. [4], which proves strong normalization from weak normalization for a class of non-dependent pure type systems (see Definition 11) by generalizing Xi's [19] and Sørensen's [17] CPS translation.

I propose revisiting the Barendregt-Geuvers-Klop conjecture in a slightly simpler framework. I begin by presenting a class of basic, concrete pure type systems I call *tiered* pure type systems. Despite their simplicity, they can be used to characterize a general class of pure type systems; so called *bounded separable persistent* pure type systems (and, in particular, bounded non-dependent systems) are disjoint unions of tiered systems.

Being concrete, the conjecture restricted to this setting is that *weak normalization implies strong normalization for all* ***tiered*** *pure type systems.* This re-framing of the problem 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 are many systems to consider, some 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 some of 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 $\triangle$ which allows for the introduction of kind variables that can appear in expressions but cannot be abstracted over. In $\lambda\mathsf{HOL}$, it is possible to derive

$$\mathfrak{A} : \square \vdash_{\lambda\mathsf{HOL}} \lambda A^{\mathfrak{A}}.\ \lambda x^{A}.\ x : \Pi A^{\mathfrak{A}}.\ A \to A.$$

A judgment of this form cannot derived in $\lambda\omega$ because the variable $\mathfrak{A}$ cannot be introduced without the axiom $\vdash_{\lambda\mathsf{HOL}} \square : \triangle$. Thus, the introduction of $\triangle$ is meaningful with respect to what expressions can be derived. But both $\lambda\mathsf{HOL}$ and $\lambda\omega$ are strongly normalizing. One basic observation is that there is a single expression inhabiting $\triangle$, namely $\square$. 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. In the case of the judgment above, the variable $\mathfrak{A}$ can be instantiated at $*$ yielding the judgment

$$\vdash_{\lambda\omega} \lambda A^{*}.\ \lambda x^{A}.\ x : \Pi A^{*}.\ A \to A$$

derived which can be derived in $\lambda\omega$.

I generalize this observation 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 in some rules. This translation can be iteratively applied to $\lambda\mathcal{S}$ until a fixed point $\lambda\mathcal{S}^{\downarrow}$ is reached. Thus, it can be used to prove the strong normalization of systems $\lambda\mathcal{S}$ for which $\lambda\mathcal{S}^{\downarrow}$ is known to be strongly normalizing. It can also be bootstrapped with existing results for the Barendregt-Geuvers-Klop conjecture. The argument is simple: if $\lambda\mathcal{S}$ is weakly normalizing, then so is $\lambda\mathcal{S}^{\downarrow}$ since it can be embedded in $\lambda\mathcal{S}$. By assumption, $\lambda\mathcal{S}^{\downarrow}$ is strongly normalizing, and so $\lambda\mathcal{S}$ 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. In particular, on a technical note, $\lambda\mathcal{S}$ may have *dependent* rules and *non-negatable* sorts (see ([4], Definition 2.23, Definition 3.1) and Definition 11 for details).

This technique bears a resemblance to the one used by Roux and van Doorn [16] in their structural theory of pure type systems, which in turn resembles the techniques of Geuvers and Nederhof [8] and Harper et al. [10]. 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. In fact, their translation can be used to eliminate some rules *between* tiered systems in a disjoint union whereas the translation presented here eliminates some rules *within* the individual summands in a disjoint union of tiered systems (all while preserving strong normalization).

It is important to emphasize that this result depends on the fact that the additional structure that can be handled is irrelevant and, in particular, irrelevant with respect to normalization, not derivability or expressibility. But if we do want to prove the full conjecture, we also have to prove it for "junk" systems, ones which may not be interesting in their own right and may have rules which don't add much to the system. This result is perhaps more meaningfully interpreted in the reverse direction: the systems $\lambda\mathcal{S}^{\downarrow}$ 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 as of now it may be useful to further develop the theory regarding what barriers exist, and what systems beyond the lambda cube – natural or not – may be 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.

2 Preliminaries

The class of pure type systems is the basis of a very general framework for describing type systems and their meta-theory. These systems vary in their sort structure and their product type formation rules, and include the entire lambda cube. Barendregt cites Berardi [5] and Terlouw [18] for their conception, though Geuvers and Nederhof [8] are cited as having given the first explicit definition, based on the previous two works. The presentations of Barendregt [2, 3] are perhaps the best known sources.[2]

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.[3]

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. This observation regarding variable annotations was first made by Geuvers ([9], Definition 4.2.9).

[2] I am of the opinion that, after the development of the lambda cube, the notion of pure type systems was soon to follow, and that all aforementioned should be cited as originators.

[3] 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.

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 [3]). 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-contexts are called **declarations**. I use capital Greek letters like 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 declaration $(x : A)$ I write $(x : A) \in \Gamma$ if that declaration 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.

The pure type system $\lambda\mathcal{S}$ 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.** $\vdash_{\lambda\mathcal{S}} s : s'$ for any axiom (s, s').
- **Variable Introduction.** For a variable ${}^{s}x$ which is fresh with respect to Γ

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

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

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

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

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

- **Abstraction.**

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

- **Application.**

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

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

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

The subscript on the turnstile is dropped when there is no fear of ambiguity. The annotations on bound 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.

An expression M is said to be **derivable** in $\lambda\mathcal{S}$ if there is some context Γ and expression A such that $\Gamma \vdash_{\lambda\mathcal{S}} M : A$. 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 2), a judgment that is not a type judgment is a term judgment, though some judgments are both type and term judgments.

2.1 Meta-Theory

I collect here the meta-theoretic lemmas necessary for the subsequent results. Much of the meta-theory of pure type systems was worked out by Geuvers and Nederhof [8], and can be found in several of the great available resources on pure type systems ([3, 4, 9, 12], among others) so proofs are omitted. For the remainder of the section, fix a pure type system $\lambda\mathcal{S}$.

▶ **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 $\Gamma \vdash B : s$ and $\Gamma, {}^s x : B \vdash C : s'$ 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 $\Gamma \vdash \Pi^s x^B.\ C : s'$ and $\Gamma, {}^s x : B \vdash M : C$ 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** (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 $\Gamma \vdash A : s$.*

▶ **Definition 3.** *A pure type system is **functional** if the following hold.*

- *If $(s, t) \in \mathcal{A}$ and $(s, t') \in \mathcal{A}$ then $t = t'$.*
- *If $(s, t, u) \in \mathcal{R}$ and $(s, t, u') \in \mathcal{R}$, then $u = u'$.*

▶ **Lemma 4** (Type Unicity). *If $\lambda\mathcal{S}$ 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 5.** *A sort s is a **top-sort** if there is no sort s' such that $(s, s') \in \mathcal{A}$. It is a **bottom-sort** if there is no sort s' such that $(s', s) \in \mathcal{A}$.*

▶ **Lemma 6** (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. $\Gamma \nvdash AB : s$

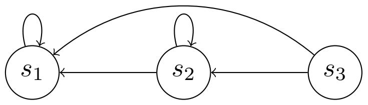

Figure 1 A visual representation of the system λU.

2.2 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., persistence as subsequently defined.

▶ **Definition 7.** *A pure type system $\lambda\mathcal{S}$ is* ***persistent*** *if it is functional (Definition 3) and*

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

From this point forward, I freely use the notation (s, s') for the rule (s, s', s'). One minor issue with properties like this is that it is often difficult to envisage the systems which satisfy them. In particular, the results tailored to a class of systems defined as such may use more meta-theoretic machinery than necessary. I choose, instead, to work with a simple class of systems I call *tiered* pure type systems, which have a very concrete description.

▶ **Definition 8.** *Let n be a non-negative integer. A pure type system is* ***n-tiered*** *if its has the form*

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

where $[n] \triangleq \{1, \ldots, n\}$.

A couple remarks about these systems:

- these systems can be envisaged as graphs as in Figure 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 only 0-tiered system is the empty pure type system; there are two 1-tiered systems, specified by either $(\{s_1\}, \emptyset, \emptyset)$ or $(\{s_1\}, \emptyset, \{(s_1, s_1)\})$, neither of which have derivable expressions; the 2-tiered systems which contain the rule (s_1, s_1) are exactly the lambda cube;
- the n-tiered systems are considered in passing by Barthe et al. ([4], Remark 2.39). They include natural subsystems of ECC^n (as defined in [15]) with only the two-sorted rules;

A large class of natural pure type systems can be classified as disjoint unions of tiered systems. In order to state this equivalence, I work in the structural theory of pure type systems presented by Roux and van Doorn [16].

▶ **Definition 9.** *For pure type systems $\lambda\mathcal{S}$ and $\lambda\mathcal{S}'$, the* ***disjoint union*** *$\lambda\mathcal{S} \sqcup \lambda\mathcal{S}'$ is specified by*

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

A class of systems which can be characterized by disjoint unions must be partitionable into atoms which can be analyzed individually. Let '$<_{\mathcal{A}}$', '$\leq_{\mathcal{A}}$', and '$\approx_{\mathcal{A}}$' denote the transitive, reflexive-transitive, and equivalence closure of $\mathcal{A}$, respectively.

▶ **Definition 10.** *A pure type system $\lambda\mathcal{S}$ is **separable** if $(s, s') \in \mathcal{R}_{\lambda\mathcal{S}}$ implies $s \approx_{\mathcal{A}} s'$. It is **atomic** if $s \approx_{\mathcal{A}} s'$ for all sorts s and s'.*

There are, of course, many examples of important *non-separable* persistent pure type systems, e.g., systems from the logic cube [2, 3, 6, 9, 7] like Berardi's formulation of λPREDω which is specified by

$$\mathcal{S} \triangleq \{*^s, \Box^s, *^p, \Box^p\}$$
$$\mathcal{A} \triangleq \{(*^s, \Box^s), (*^p, \Box^p)\}$$
$$\mathcal{R} \triangleq \{(*^p, *^p), (\Box^p, *^p), (\Box^p, \Box^p), (*^s, *^p), (*^s, \Box^p)\}$$

The rules $(*^s, *^p)$ and $(*^s, \Box^p)$ "cross" between two tiered systems.[4] Despite this, there are also useful classes of systems which *are* separable, e.g., generalized non-dependent systems are separable by fiat.

▶ **Definition 11.** *Let $\lambda\mathcal{S}$ be a pure type system.*

- *$\lambda\mathcal{S}$ satisfies the **ascending chain condition** if '$<_{\mathcal{A}}$' does, i.e., there is no infinite sequence of sorts $s, s', s'', \dots$ such that $s < s' < s'' \dots$; it satisfies the **descending chain condition** if there is no infinite sequence of sorts $s, s', s'', \dots$ such that $s > s' > s'' \dots$; it is **bounded** if it satisfies both the ascending and descending chain conditions.*
- *$\lambda\mathcal{S}$ is **weakly non-dependent** if $(s, s', s'') \in \mathcal{R}$ implies $s \geq s' \geq s''$.*
- *$\lambda\mathcal{S}$ is **stratified** if it satisfies the ascending chain condition and is non-dependent.*
- *$\lambda\mathcal{S}$ is **generalized non-dependent** if it is stratified and persistent. If $\lambda\mathcal{S}$ is also bounded, I will write that it is **bounded non-dependent**, and if it is tiered, I will just write that it is non-dependent.*

We can now characterize disjoint unions of tiered systems in terms of the above properties. The proof of Lemma 12 is omitted, but it follows roughly by showing that '$\leq_{\mathcal{A}}$' is a total order.

▶ **Lemma 12.** *A pure type system is tiered if and only if it is persistent, bounded, and atomic.*

▶ **Lemma 13.** *A pure type system is persistent, bounded, and separable if and only if is the disjoint union of tiered pure type systems.*

Proof. It is straightforward to verify that tiered systems are persistent, bounded, and atomic, and so their disjoint unions are persistent, bounded, and separable. In the other direction, let $\lambda\mathcal{S}$ be a pure type system that is persistent, bounded, and separable and consider the partition $\mathcal{P}$ of $\mathcal{S}$ into $\approx_{\mathcal{A}}$-equivalence classes. Let $\mathcal{S}_p$ be such an equivalence class and let $\lambda\mathcal{S}_p$ denote the pure system specified by

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

[4] I'd like to specifically thank the reviewer who reminded me of this example.

The system $\lambda\mathcal{S}_p$ is persistent and bounded because $\lambda\mathcal{S}$ is, and it is atomic by definition, so by Lemma 12 it is tiered. We can then view $\lambda\mathcal{S}$ as the system $\bigsqcup_{\mathcal{S}_p \in \mathcal{P}} \lambda\mathcal{S}_p$.[5] Note that all axioms are accounted for by fiat and all rules are accounted for by separability. ◀

▶ **Corollary 14.** *A pure type system is bounded non-dependent if and only if it is the disjoint union of non-dependent tiered pure types systems.*

Roux and van Doorn [16] 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, bounded, separable systems it suffices to consider tiered systems.

▶ **Proposition 15.** *If weak normalization implies strong normalization for all tiered pure type systems, then the same is true for all persistent, bounded, separable pure type systems. In particular, if weak normalization implies strong normalization for all non-dependent pure type systems, then the same is true for all bounded non-dependent pure type systems.*[6]

I close this section with some useful features of 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 expressions and classifying expressions according to their degree. This result is due to Berardi [6] and Geuvers and Nederhof [8], and the presentation here roughly follows the same course.

▶ **Definition 16.** *The **degree** of an expression is given by the following function* $\deg : \mathsf{T} \to \mathbb{N}$.

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

Let T_j *denote* $\{M \in \mathsf{T} \mid \deg(M) = j\}$ *and let* $\mathsf{T}_{\geq j}$ *denote* $\{M \in \mathsf{T} \mid \deg(M) \geq j\}$.

▶ **Lemma 17** (Classification)**.** *Let $\lambda\mathcal{S}$ 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* $\Gamma \vdash_{\lambda\mathcal{S}} A : s_n$ *for some context* Γ.
- *For* $i \in [n-1]$, *we have* $\deg(A) = i$ *if and only if* $\Gamma \vdash_{\lambda\mathcal{S}} A : B$ *and* $\Gamma \vdash_{\lambda\mathcal{S}} B : s_{i+1}$ *for some context* Γ *and expression* B.

In particular, for context Γ and expressions M and A, if $\Gamma \vdash_{\lambda\mathcal{S}} M : A$ then $\deg(A) = \deg(M) + 1$.

[5] Formally, they are isomorphic pure type systems. The definition of a pure type system homomorphism is as one might expect, see the definition of Geuvers ([9], Definition 4.2.5) – which is also used by Roux and van Doorn [16] – for more details.

[6] This all sits in a more general theory. A natural extension of tiered systems includes infinite tiered systems and even cyclic systems, which can help better characterize classes of systems, like generalized non-dependent systems and persistent separable systems.

Finally, some useful facts about degree. See the presentation by Barendregt [3] for proofs in the 2-tiered case.

▶ **Lemma 18.** *Let $\lambda\mathcal{S}$ be an n-tiered pure type system and let A and B be expressions derivable in $\lambda\mathcal{S}$.*

- *If* $\deg(B) = j - 1$ *then* $\deg(A[B/^{s_j}x]) = \deg(A)$.
- *If* $A \twoheadrightarrow_\beta B$, *then* $\deg(A) = \deg(B)$.

3 Irrelevancy-Eliminating Translation

Fix an n-tiered pure type system $\lambda\mathcal{S}$. I first describe the sorts which are *top-sort-like*. Recall that s is a top-sort if there is no sort s' such that $(s, s') \in \mathcal{A}$, so s_n is the only top-sort of $\lambda\mathcal{S}$. Top-sorts are interesting in part because they tend to be sparsely inhabited. A top-sort-like sort s_i which is not a top-sort has the sort s_{i+1} above it, but to ensure s_i is sparsely inhabited, s_{i+1} should not appear in any rules. We will also be interested in top-sort-like sorts which themselves do not appear in any rules.

▶ **Definition 19.**

- *A sort s_i is **rule-isolated** if for all j, neither (s_j, s_i) nor (s_i, s_j) appear in $\mathcal{R}_{\lambda\mathcal{S}}$.*
- *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).*
- *A sort s_i is **completely isolated** if it is top-sort-like and rule-isolated.*

Next, I describe the structure that will be considered irrelevant with respect to normalization. Roughly speaking, this includes rules on top-sort-like sorts which allow for the derivation of redexes on expressions from sparsely inhabited types. It will be possible to essentially pre-reduce these redexes in the translation, eliminating the need for the rules in the target system of the translation. In what follows, it will be convenient to consider *sets* of top-sort-like sorts. I call a subset $\mathcal{I}$ of $[n]$ an *index set* for $\lambda\mathcal{S}$, and denote by $\mathcal{S}_\mathcal{I}$ the set $\{s_i \mid i \in \mathcal{I}\}$.

▶ **Definition 20.**

- *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 $(s_j, s_i) \in \mathcal{R}_{\lambda\mathcal{S}}$. A sort s_i is **irrelevant** if it is $[n]$-irrelevant.*
- *An index set $\mathcal{I}$ is **completely irrelevant** in $\lambda\mathcal{S}$, if for each i in $\mathcal{I}$,*
  - *s_i is top-sort-like and irrelevant;*
  - *s_{i-1} is $([n] \setminus \mathcal{I})$-irrelevant.*

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 irrelevant after removing the rules associated with sorts in $\mathcal{S}_\mathcal{I}$. Note also that if $(s_i, s_i) \in \mathcal{R}_{\lambda\mathcal{S}}$, then any completely irrelevant index set cannot contain $i - 1$, i or $i + 1$. 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.

3.1 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 21.** *Let s_i be an irrelevant sort such that s_i is a top-sort or s_{i+1} is irrelevant. For every derivable expression A, if $\deg(A) = i$ 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 6) 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 derivations. The cases in which the last inference is an axiom, variable introduction, weakening, or conversion are straightforward. The last inference cannot be a product type formation because s_i is irrelevant. The last inference cannot be an abstraction or application because s_{i+1} is irrelevant. ◀

This does not hold if s_{i+1} is not irrelevant. If $(s_{i+1}, s_{i+1}) \in \mathcal{R}_{\lambda\mathcal{S}}$, for example, then $\varnothing \vdash (\lambda x^{s_i}.\ x)s_{i-1} : s_i$ is derivable. This is why we require *both* s_i and s_{i+1} to be irrelevant.

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 22.** *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 $(\mathcal{S}_{\lambda\mathcal{S}} \setminus \mathcal{I})$-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 21, which means the inductive hypothesis can be applied directly to the left antecedent judgment. ◀

▶ **Lemma 23.** *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 24.** *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 [17].

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 sparsely inhabited, it is unclear *a priori* what the value of the expression will be after a series of reductions. By Lemma 23, 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 25.** *Define the context-indexed function $\tau_\Gamma : \mathsf{Ctx} \times \mathsf{T} \to \mathsf{T}$ by induction on both arguments as follows.*

$$\tau_\Gamma(s_i) \triangleq s_i$$

$$\tau_\Gamma({}^{s_i}x) \triangleq \begin{cases} s_{i-2} & \textit{if } i \in \mathcal{I} \textit{ and } ({}^{s_i}x : s_{i-1}) \in \Gamma \\ \bullet_z & \textit{if } i \in \mathcal{I} \textit{ and } ({}^{s_i}x : {}^{s_{i+1}}z) \in \Gamma \\ {}^{s_i}x & \textit{otherwise} \end{cases}$$

$$\tau_\Gamma(\Pi x^A.\ B) \triangleq \begin{cases} \tau_{\Gamma,x:A}(B) & \deg(A) \in \mathcal{I} \\ \Pi x^{\tau_\Gamma(A)}.\ \tau_{\Gamma,x:A}(B) & \textit{otherwise} \end{cases}$$

$$\tau_\Gamma(\lambda x^A.\ M) \triangleq \begin{cases} \tau_{\Gamma,x:A}(M) & \deg(A) \in \mathcal{I} \\ \lambda x^{\tau_\Gamma(A)}.\ \tau_{\Gamma,x:A}(M) & \textit{otherwise} \end{cases}$$

$$\tau_\Gamma(MN) \triangleq \begin{cases} \tau_\Gamma(M) & \deg(N)+1 \in \mathcal{I} \\ \tau_\Gamma(M)\tau_\Gamma(N) & \textit{otherwise} \end{cases}$$

where $\bullet_z$ is a distinguished variable. This function is used to define a function on contexts as

$$\tau(\varnothing) \triangleq \varnothing$$

$$\tau(\Gamma, {}^{s_j}x : A) \triangleq \begin{cases} \tau(\Gamma) & j \in \mathcal{I} \\ \tau(\Gamma), {}^{s_j}x : s_{j-1}, \bullet_x : {}^{s_j}x & \textit{if } j-1 \in \mathcal{I} \textit{ and } A = s_{j-1} \\ \tau(\Gamma), {}^{s_j}x : \tau_\Gamma(A) & \textit{otherwise.} \end{cases}$$

As for proving the desired features of this translation, first note if $i \in \mathcal{I}$, then the translation maps expressions of degree $i-1$ (where $i \in \mathcal{I}$) to a sort or a $\bullet$-variable.

▶ **Proposition 26.** *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 24, 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 argument of the translation can be weakened when the last variable does not appear in the expression argument.

▶ **Proposition 27.** *For any context Γ, expressions M, A, and B, and variable x, if $\Gamma \vdash M : A$ and $\Gamma \vdash B : s_i$ then $\tau_{\Gamma,x:B}(M) = \tau_\Gamma(M)$.*

We now prove the standard substitution-commutation and β-preservation lemmas for this translation.

▶ **Lemma 28.** *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 21. If $A = s_{i-1}$, then by Proposition 26 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 27 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 29.** *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 30.** *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 28). 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 21 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 24 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 31.** *The* ***irrelevance reduction*** *of $\lambda\mathcal{S}$, denoted here by $\lambda\mathcal{S}^-$, is the n-tiered system specified by the rules $\mathcal{R}_{\lambda\mathcal{S}} \setminus \{(s_i, s_j) \mid i \in \mathcal{I}$ and $j \in [n]\}$.*

▶ **Lemma 32.** *For context Γ and expressions M and A, if*

$$\Gamma \vdash_{\lambda\mathcal{S}} M : A \qquad \textit{then} \qquad \tau(\Gamma) \vdash_{\lambda\mathcal{S}^-} \tau_\Gamma(M) : \tau_\Gamma(A).$$

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 26, 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

$$\begin{aligned} \tau(\Gamma) &= \tau(\Gamma, {}^{s_i}x : A) \\ \tau_\Gamma(\lambda x^A.\ M) &= \tau_{\Gamma, x:A}(M) \\ \tau_\Gamma(\Pi x^A.\ B) &= \tau_{\Gamma, x:A}(B) \end{aligned}$$

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 28, 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 30. ◀

It remains to show that this translation is path-preserving. The guiding observation is that β_2-reductions cannot make more "complex" redexes. We define a complexity measure which captures this observation by its being monotonically decreasing in β_2-reductions.

▶ **Definition 33.** *The* ***shallow λ-depth*** *of an expression 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 34.** *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 domain-full version of a result by Lévy for the untyped lambda calculus about the creation of new redexes [14]. I give the statement of the result here without proof (See [11, 20] among others for the standard definition of a residual).

▶ **Lemma 35.** *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 created 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 36.** *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 24, the expression Q is of the form $\Pi {x_1}^{A_1}.\ \dots \Pi {x_k}^{A_k}.\ B$ 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 35, 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 section is standard.

▶ **Theorem 37.** *If $\lambda\mathcal{S}^-$ is strongly normalizing, then $\lambda\mathcal{S}$ is strongly normalizing.*

Proof. Suppose there is an infinite reduction sequence in $\lambda\mathcal{S}$

$$M_1 \to_\beta M_2 \to_\beta \dots$$

where M_1 is derivable in $\lambda\mathcal{S}$ from the context Γ. Since μ is monotonically decreasing in β_2-reductions (Lemma 36), 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 30 implies there infinitely many β-reductions in the reduction path

$$\tau_\Gamma(M_1) \twoheadrightarrow_\beta \tau_\Gamma(M_2) \twoheadrightarrow_\beta \dots$$

where $\tau_\Gamma(M_1)$ is derivable in $\lambda\mathcal{S}^-$ by Lemma 32. ◀

3.2 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.

Fix an n-tiered pure type system $\lambda\mathcal{S}$ with $n > 2$, and a completely isolated sort s_i.[7] 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.

[7] The restriction on n is a technicality that ensures the target system is nontrivial. See, for example, the variable case of Definition 38.

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. It will be useful to be more careful about variable annotations in the following definitions and lemmas.

▶ **Definition 38.** *Define the context-indexed function $\theta_\Gamma : \mathsf{Ctx} \times \mathsf{T} \to \mathsf{T}$ inductively on both arguments as follows.*

$$\theta_\Gamma(s_j) \triangleq \begin{cases} s_j & j < i \\ s_{j-1} & \textit{otherwise} \end{cases}$$

$$\theta_\Gamma({}^{s_j}x) \triangleq \begin{cases} s_{i-2} & \textit{if } j = i \textit{ and } ({}^{s_i}x : s_{i-1}) \in \Gamma \\ {}^{s_j}x & j < i \\ {}^{s_{j-1}}x & \textit{otherwise} \end{cases}$$

$$\theta_\Gamma(\Pi^{s_j}x^A.\ B) \triangleq \Pi^{\theta_\Gamma(s_j)}x^{\theta_\Gamma(A)}.\ \theta_{\Gamma,x:A}(B)$$

$$\theta_\Gamma(\lambda^{s_j}x^A.\ M) \triangleq \lambda^{\theta_\Gamma(s_j)}x^{\theta_\Gamma(A)}.\ \theta_{\Gamma,x:A}(M)$$

$$\theta_\Gamma(MN) \triangleq \theta_\Gamma(M)\theta_\Gamma(N)$$

This function is used to define a function on contexts as

$$\theta(\varnothing) \triangleq \varnothing$$

$$\theta(\Gamma, {}^{s_j}x : A) \triangleq \begin{cases} \theta(\Gamma) & \textit{if } j = i \textit{ and } A = s_{i-1} \\ \theta(\Gamma), {}^{\theta_\Gamma(s_j)}x : \theta_\Gamma(A) & \textit{otherwise.} \end{cases}$$

As with the previous translation, contexts can be weakened without changing the value of the function (in analogy with Proposition 27 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 39.** *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 40.** *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 39, 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 completely isolated sort s_i is removed and potential sorts above it are shifted down.

▶ **Definition 41.** *The i-**collapse** of $\lambda\mathcal{S}$, denote here by $\lambda\mathcal{S}^*$, is the $(n-1)$-tiered systems specified by the rules $\{(\theta_\varnothing(s_j), \theta_\varnothing(s_k)) \mid (s_j, s_k) \in \mathcal{R}_{\lambda\mathcal{S}}\}$.*

▶ **Lemma 42.** *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 axiom (s_{i-1}, s_i) for the translated derivation *in the system* $\lambda\mathcal{S}^*$, 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 to derive

$$\theta(\Gamma) \vdash \theta_\Gamma(M)\theta_\Gamma(N) : \theta_{\Gamma,x:A}(B)[\theta_\Gamma(N)/x]$$

where $\theta_{\Gamma,x:A}(B)[\theta_\Gamma(N)/x] = \theta_\Gamma(B[N/x])$ by Lemma 39.

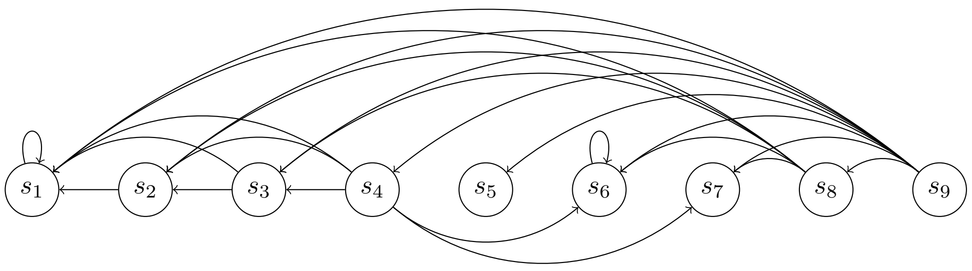

Figure 2 A system with a non-trivial sequence of irrelevance reductions.

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 21, in fact $A = B$. If $B = s_{i-1}$, then by Corollary 24 we again have $A = B$. Otherwise, by Lemma 40, $\theta_\Gamma(A) =_\beta \theta_\Gamma(B)$ 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 straightforward.

▶ **Theorem 43.** *If $\lambda\mathcal{S}^*$ is strongly normalizing then $\lambda\mathcal{S}$ is strongly normalizing.*

3.3 The Final Translation

We can now consider the fixed-points of the above translations.

▶ **Definition 44.** *Let $\tau(\lambda\mathcal{S})$ denote the fixed-point of taking the irrelevance reduction of $\lambda\mathcal{S}$ with respect to maximum completely irrelevant index sets. That is, repeat $\lambda\mathcal{S} := \lambda\mathcal{S}^-$ taken with respect to the maximum completely irrelevant index set of $\lambda\mathcal{S}$, until its maximum completely irrelevant index set is empty.*

▶ **Definition 45.** *Let $\theta(\lambda\mathcal{S})$ denote the fixed-point of taking the i-collapse of $\lambda\mathcal{S}$, where i is the maximum index of a complete isolated sort in $\lambda\mathcal{S}$, if one exists. That is, repeat $\lambda\mathcal{S} := \lambda\mathcal{S}^*$ taken with respect to the maximum index of a completely isolated sort of $\lambda\mathcal{S}$ until it has no completely isolated sort or is 2-tiered.*

Note that a sort which does not appear in the maximum completely irrelevant index set of $\lambda\mathcal{S}$ may appear in the maximum completely irrelevant set of $\lambda\mathcal{S}^-$. See Figure 2 for a tiered system with a non-trivial sequence of irrelevance reductions. 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 $\{4, 8\}$. One can then imagine how this effect can be scaled up to larger systems.

I will write $\lambda\mathcal{S}^\downarrow$ for $\tau(\lambda\mathcal{S})$ and $\lambda\mathcal{S}^\Downarrow$ for $\theta(\tau(\lambda\mathcal{S}))$. Since no rules are removed by an i-collapse (only shifted), no sort can become completely isolated and no new completely irrelevant index set can be created, so in fact $\lambda\mathcal{S}^\Downarrow$ is the fixed-point of $\theta \circ \tau$.

The main two theorems are as follows.

▶ **Theorem 46.** *For any tiered pure type system $\lambda\mathcal{S}$, if $\lambda\mathcal{S}^\Downarrow$ is strongly normalizing, then $\lambda\mathcal{S}$ is strongly normalizing.*

▶ **Theorem 47.** *For any tiered pure type system $\lambda\mathcal{S}$, if weak normalization implies strong normalization for $\lambda\mathcal{S}^{\downarrow}$, then weak normalization implies strong normalization for $\lambda\mathcal{S}$.*

In particular, if $\lambda\mathcal{S}^{\downarrow}$ satisfies the conditions of Barthe et al. ([4], Theorem 5.21), then weak normalization implies strong normalization in $\lambda\mathcal{S}$. This does not immediately apply to $\lambda\mathcal{S}^{\Downarrow}$ since it is not immediate that weak normalization is preserved from $\lambda\mathcal{S}$ to $\lambda\mathcal{S}^{*}$; the sorts are not preserved. Note that it is immediate in the case that the completely isolated sort is a top-sort. Given the scope of this work, I leave this to be verified, it is a natural step in extending these results.

4 Conclusions

I have presented a path-preserving translation which eliminates some irrelevant structure. Again, this structure is irrelevant with respect to normalization, not derivability. When combined with results for the Barendregt-Geuvers-Klop conjecture, it widens the class of systems for which the conjecture applies. 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, it appears that dealing with circular rules is one of the clear barriers in strengthening these results. For 3-tiered systems, we extend the conjecture to (and can prove strong normalization of) the system[8]

$$\begin{tikzcd}
s_1 \arrow[loop above] & s_2 \arrow[l] & s_3 \arrow[l] \arrow[ll, bend right]
\end{tikzcd}$$

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.

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. Regardless, I hope to have demonstrated with this translation that, despite the full Barendregt-Geuvers-Klop conjecture seeming quite far from being resolved, there are still a number of approachable questions and avenues for further development.

References

1 TLCA List of Open Problems, 2014. `http://tlca.di.unito.it/opltlca/`.

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

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

4 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.

[8] This system is not covered by the Barthe et al. result because s_2 is not negatable.

5 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.

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

7 Herman Geuvers. The Calculus of Constructions and Higher Order Logic. In *The Curry-Howard Isomorphism*, volume 8 of *Cahiers du Centre de Logique (Universit'e catholique de Louvain), Academia, Louvain-la-Neuve (Belgium)*, pages 131–191, 1995.

8 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.

9 Jan Herman Geuvers. *Logic and Type Systems*. PhD thesis, University of Nijmegen, 1993.

10 Robert Harper, Furio Honsell, and Gordon Plotkin. A Framework for Defining Logics. *Journal of the ACM*, 40(1):143–184, 1993.

11 Gérard Huet. Residual Theory in λ-Calculus: A Formal Development. *Journal of Functional Programming*, 4(3):371–394, 1994.

12 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.

13 Jeroen Ketema, Jan Willem Klop, and V van Oostrom. Vicious Circles in Rewriting Systems. *Artificial Intelligence Preprint Series*, 52, 2004.

14 Jean-Jacques Lévy. *Réductions Correctes et Optimales dans le Lambda-Calcul*. PhD thesis, L'Université Paris VII, 1978.

15 Zhaohui Luo. *An extended calculus of constructions*. PhD thesis, University of Edinburgh, 1990.

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

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

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

19 Hongwei Xi. On Weak and Strong Normalisations. Technical report, Carnegie Mellon University, Department of Mathematics, 1996.

20 Hongwei Xi. Development Separation in Lambda-Calculus. *Electronic Notes in Theoretical Computer Science*, 143:207–221, 2006.