An Irrelevancy-Eliminating Translation for Pure Type Systems

Nathan Mull
University of Chicago, USA

Abstract

I present an infinite-reduction-path-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.* [3] 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 [13]—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, ruling out any unhelpful structure that does not affect normalization. This work is done in the framework of *tiered* pure type systems, a straightforward simplification of stratified persistent systems I introduce 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.CVIT.2016.23

1 Introduction

The class of pure type systems [15, 4, 1, 2] was introduced as a natural generalization of the lambda cube which also 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 the meta-theoretic properties of the system (especially given the minimal collection of type formers). One such meta-theoretic 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. The well-known *Barendregt-Geuvers-Klop conjecture* posits that weak normalization implies strong normalization for all pure type systems.

Very little progress has been made on this conjecture, in part because pure type systems in general are too unstructured to be amenable to standard techniques. Though natural, the generalization to pure type systems from the lambda cube is in some sense the most obvious one, a basic syntactic ambiguation of the inference rules to allow for maximal freedom where the lambda cube imposes its sort-structural restrictions. The resulting systems may fail to have the meta-theoretic properties one might expect (*e.g.*, type unicity). It is, therefore, common to consider classes of pure type systems that maintain these meta-theoretic properties. The state of the art of the conjecture is the result of Barthe *et al.* [3], which states that weak normalization implies strong normalization for *generalized non-dependent, clean, negatable*[1] pure type systems. The proof of this result depends on a very nice—though somewhat complicated—generalization of Sørensen's CPS translation [14] for $\lambda\omega$ (among other systems).

[1] These are technical restrictions, discussed further in the later exposition.

I have recently given a slightly simpler presentation of the same result [12] but based on Xi's Thunkification translation [16].

I propose revisiting the Barendregt-Geuvers-Klop conjecture in a slightly simpler framework. I start by presenting a class of basic, concrete pure type systems I call *tiered* pure type systems. Despite their simplicity they are sufficient to consider with regards to questions about normalization. Stratified persistent systems (and generalized non-dependent systems) are disjoint unions of tiered systems, so tiered systems are in some sense the *atoms* of these systems. For concreteness, the conjecture restricted to this setting is that *weak normalization implies strong normalization in all* ***tiered*** *pure type systems.* The result of Barthe *et al.* implies the conjecture for non-dependent, clean, negatable tiered systems, and so it remains to remove any and all of these restrictions.

This simple reframing of the problem (a moving of the goalposts, so to speak) is a minor though I believe important step towards making further progress on the full version of the conjecture. But even in this setting, there is a huge number of systems to consider, many of which contain what amounts to "junk" structure. The primary contribution of this paper is a translation of pure type systems which preserves typability and infinite reduction paths (I will simply write "path-preserving" from this point forward) and removes this irrelevant structure. By removing structure here, I mean that the target system of the translation is the same as the source system but with some sorts and rules removed.

Consider, for example, the system $\lambda\mathsf{HOL}$, which may be thought of as the system $\lambda\omega$ with an additional *superkind* sort $\triangle$ that allows for the introduction of kind variables that can appear in expressions but cannot be abstracted over. The introduction of $\triangle$ is meaningful with respect to what expressions can be derived, but unsurprisingly both $\lambda\mathsf{HOL}$ and $\lambda\omega$ are strongly normalizing. One basic observation here is that there is a single expression inhabiting $\triangle$, namely the *kind* sort $\Box$. This sparsity of inhabitation can be leveraged to define a path-preserving translation from $\lambda\mathsf{HOL}$ to $\lambda\omega$ and, in fact, from any pure type system with an isolated top-sort to the same system but without the top-sort.

I generalize this basic idea in two ways. First, I define a path-preserving translation that eliminates not just top-sorts but also any sort which is *top-sort-like*. Second, I extend this translation to eliminate not just isolated sorts, but also sorts which may appear as sources of Π-types. In particular, this allows for the elimination of some *dependent* rules in the system. This translation can be iteratively applied to a system λS until a fixed point $\lambda S^{\downarrow}$ is reached, which is equivalent to λS with respect to strong normalization. Thus, this translation can be used to prove the strong normalization of systems λS for which $\lambda S^{\downarrow}$ is known to be strongly normalizing. But perhaps more importantly, it can be bootstrapped with existing results for the Barendregt-Geuvers-Klop conjecture. The argument is simple: if λS is weakly normalizing, then so is $\lambda S^{\downarrow}$ since it can be embedded in λS. Then $\lambda S^{\downarrow}$ is strongly normalizing by assumption, and so λS is strongly normalizing by the path-preserving translation. Bootstrapping with the result of Barthe *et al.* yields a proof of the Barendregt-Geuvers-Klop conjecture for a larger class of systems.

This technique bears a resemblence to the one used by Roux and van Doorn [13] in their structural theory of pure type systems, which in turn resembles the techniques of Geuvers and Nederhof [6] and Harper *et al.* [7]. In all these works, a translation is defined from one pure type system into another which has fewer rules. And though it is not explicitly stated, the translation of Roux and van Doorn can be bootstrapped in the same way as described above to push the state of the conjecture. In fact, their translation can be used to eliminate rules *between* disjoint systems in order to, say, make it stratified and persistent (*i.e.*, a disjoint unions of tiered systems) whereas the translation presented here eliminates

rules *within* the individual summands in a disjoint union of tiered systems.

It is important to emphasize that this results depends on the fact that the additional structure that can be handled is in some sense irrelevant. But if we do want to prove the full conjecture, we also have to prove it for a class of "junk" systems, ones which may not be interesting in their own right and may have rules which don't add much to the system. In a way, this result is perhaps more meaningfully interpreted in the reverse direction: the systems $\lambda S^{\downarrow}$ for which the conjecture is *not* known to hold are targets for the developments of better techniques. Ideally, some technique could handle all these systems uniformly, but for now it may be useful to further develop the theory regarding what barriers exist, what systems beyond the lambda cube, natural or not, may be most important to study.

In what follows I present some preliminary material, which includes some exposition on tiered systems. I then define the irrelevancy-eliminating translation in two parts: one part for eliminating rules and one for eliminating sorts. The final translation will be taken as the composition of these two translations. Finally, I present its application to the Barendregt-Geuvers-Klop conjecture and conclude with a short section on what it implies about the systems which remain to be studied.

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.[2]

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 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, Q, R, S, A, B, C and D 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 the statements appearing in proto-contexts **declarations**. We use Γ, Δ and Υ as meta-variables for proto-contexts. The sequence braces of proto-contexts are often dropped and concatenation of contexts is denoted by comma-separation. Substitution and the β-equality relation are extend to contexts element-wise. For a proto-context Γ and statement $(x : A)$ we write $(x : A) \in \Gamma$ if $(x : A)$ appears in Γ. We define the subset relation $\Gamma \subset \Delta$ for proto-contexts Γ and Δ analogously. Let $\mathcal{C}$ denote the set of all proto-contexts.

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

[2] 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 refers to a particular sort in tiered systems. I will try to be as clear as possible when distinguishing between these two cases of notation.

to the rules below. Likewise, the designation "context" is reserved for proto-contexts that appear in some (derivable) judgment.[3]

▶ **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') we can derive $\vdash_{\lambda S} s : s'$.*
- ***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'')*

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

- ***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 expressions 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 2). 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 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 $\Gamma \vdash_{\lambda S} A : s$ where $s \in \mathcal{S}$, and a **term judgment** if it is of the form $\Gamma \vdash_{\lambda S} M : A$ where $\Gamma \vdash_{\lambda S} A : s$ for some sort s. We also say that M is a term and A is a type.

To conclude this part of the section, I collect here the minimal collection of meta-theoretic lemmas and definitions explicitly necessary for the subsequent results. I do not present any proofs, and instead refer the reader to any of the nice resources on pure type systems ([2, 3, 9], among others). In what follows, fix a pure type system λS.

▶ **Lemma 2.** *(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 $(s, s') \in \mathcal{A}$.*

[3] 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').

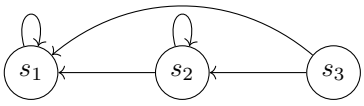


Figure 1 A visual representation of the system λU, where an arrow (s_i, s_j) indicates the presence of the rule (s_i, s_j, s_j) (axioms are not represented in the graph except in the order the nodes are presented).

- *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 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 $\Gamma \vdash A : s$.*

▶ **Definition 4.** *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 $s'' = t''$.*

▶ **Definition 5.** *A sort s is a **top-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. $\Gamma \not\vdash s : A$
2. $\Gamma \not\vdash x : s$
3. $\Gamma \not\vdash AB : s$
4. $\Gamma \not\vdash \lambda x^A.\ B : s$.

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

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

*A tiered system is **based** if it has the rule (s_1, s_1, s_1).*

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 based 2-tiered systems are exactly the lambda cube.
- The n-tiered systems are considered in passing by Barthes *et al.* (Remark 2.39, [3]). They include the natural subsystems of ECC^n (as defined in [11]) which have only rules of the form (s_i, s_j, s_j) (*i.e.*, excluding rules of the form (s_i, s_j, s_i)).

Working in tiered systems simplifies the arguments in the subsequent sections 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 4) in the previous section.

▶ **Definition 8.** *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 9.** *A pure type system is* ***weakly stratified*** *if there is no infinite sequence of sorts $s, s', s'', \ldots$ such that*

$$s <_{\mathcal{A}} s' <_{\mathcal{A}} s'' <_{\mathcal{A}} \ldots \qquad \text{or} \qquad s >_{\mathcal{A}} s' >_{\mathcal{A}} s'' >_{\mathcal{A}} \ldots$$ [4]

In order to state the following equivalence, we work in the structural theory of pure type systems of Roux and van Doorn [13] (albeit, not the particularly interesting part of it).

▶ **Definition 10.** *For pure type systems λS and $\lambda S'$, the* ***disjoint union*** *$\lambda S \sqcup \lambda S'$ is specified by*

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

▶ **Lemma 11.** *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 $\mathcal{S}_t = \{s \mid s \leq_{\mathcal{A}} t\}$. We say a **chain** from s to s' is a sequence of sorts $(v_1, \ldots, v_k)$ such that $v_1 = s$ and $v_k = s'$ and $(v_i, v_{i+1}) \in \mathcal{A}$ 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. Functionality then 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 $\mathcal{S}_t$ 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.[5] ◀

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

▶ **Definition 12.** *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 $s' \leq_{\mathcal{A}} s$. A tiered pure type system λS is **non-dependent** its rules are non-dependent.*

▶ **Corollary 13.** *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 [13] 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 14.** *The **degree** of an expression is given according to the following function* $\deg : \mathsf{T} \to \mathbb{N}$.

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

[5] The definition of a pure type system homomorphism is as one might expect, see the work by Roux and van Doorn [13] for further details.

▶ **Lemma 15.** *(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.

▶ **Lemma 16.** *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* $\deg A = \deg B$.

3 Irrelevancy-Eliminating Translation

We begin with a couple definitions. Fix an n-tiered pure type system λS. We first describe the sorts which are sufficiently *top-sort-like*, as well as the properties of these sorts that induce irrelevant structure. In what follows, it will be convenient to consider *sets* of top-sort-like sorts. We call a subset $\mathcal{I}$ of $[n]$ an **index set** for λS, and denote by $\mathcal{S}_\mathcal{I}$ the set $\{s_i \mid i \in \mathcal{I}\}$.

▶ **Definition 17.**

- *A sort s_i is* ***rule-isolated*** *if it does not appear in any rules, i.e., for all j, neither (s_j, s_i) nor (s_i, s_j) appear in $\mathcal{R}_{\lambda S}$.*
- *A sort s_i is* ***top-sort-like*** *if $i < n$ implies s_{i+1} is rule-isolated (i.e., s_i is a top sort or its succeeding sort is rule-isolated).*
- *For any index set $\mathcal{J}$, a sort s_i is* ***$\mathcal{J}$-irrelevant*** *if there is no sort s_j such that $j \in \mathcal{J}$ and $(s_j, s_i) \in \mathcal{R}_{\lambda S}$. We say s_i is* ***irrelevant*** *if it is $[n]$-irrelevant.*
- *An index set $\mathcal{I}$ is* ***completely irrelevant*** *in λS, if for each i in $\mathcal{I}$,*
  - *s_i is top-sort-like and irrelevant;*
  - *s_{i-1} is $([n] \setminus \mathcal{I})$-irrelevant.*
- *A sort s_i is* ***completely isolated*** *if s_i is top-sort-like and rule-isolated.*

In the case of complete irrelevance, if $\mathcal{I}$ is a singleton set $\{i\}$, then the only rule with s_{i-1} appearing second is (s_i, s_{i-1}). By considering sets of indices simultaneously, we can make weaker assumptions on these preceding sorts. The condition of $([n] \setminus \mathcal{I})$-irrelevance ensures that s_{i-1} becomes isolated after removing the rules associated with sorts in $\mathcal{S}_\mathcal{I}$. Note also that if $(s_i, s_i) \in \mathcal{R}_{\lambda S}$, then any completely irrelevant index set cannot contain $i-1$, i or $i+1$. In particular, for based systems, 1 and 2 cannot appear in any completely irrelevant index set. Finally, it is important that there is a *unique maximum completely irrelevant index set*. In particular, the union of any two completely irrelevant index sets is completely irrelevant.

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 18.** *Let s_i be an irrelevant sort such that s_i is a top-sort or s_{i+1} is irrelevant. For context Γ and expression A, if $\Gamma \vdash A : s_i$, then $A = s_{i-1}$ or $A \in \mathsf{V}_{s_{i+1}}$.*

Proof. If $i = n$, then this follows directly from the top-sort lemma (Lemma 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 A. By the generation lemma (Lemma 2), A cannot be a Π-expression as s_i is irrelevant. Likewise, A cannot be a λ-expression since $s_i \neq_\beta \Pi x^B.\ C$ for any expressions B and C. Finally, A cannot be an application MN since the generation lemma would imply that there is a context Γ and expressions B and C such that $\Gamma \vdash M : \Pi x^B.\ C$ and $\Gamma \vdash \Pi x^B.\ C : s_{i+1}$. But this is not possible since $i + 1$ is irrelevant. ◀

This does not hold if s_{i+1} is not irrelevant. If $(s_{i+1}, s_{i+1}) \in \mathcal{R}_{\lambda S}$, for example, then $\varnothing \vdash s_i \to s_i : s_{i+1}$ and $\varnothing \vdash \lambda x^{s_i}.\ x : s_i \to s_i$ so $\varnothing \vdash (\lambda x^{s_i}.\ x) s_{i-1} : s_i$. This is why we require *both* s_i and s_{i+1} to be irrelevant. Similarly, this does not hold for all expressions of degree i. If $(s_{i+2}, s_{i+2}) \in \mathcal{R}_{\lambda S}$ then $\varnothing \vdash s_{i+1} \to s_{i+1} : s_{i+2}$ and $\varnothing \vdash \lambda x^{s_{i+1}}.\ x : s_{i+1} \to s_{i+1}$.

The primary challenge moving forward is dealing with the fact that variables may appear as types of sort s_i. These variables are what will necessitate s_{i+1} being not just irrelevant, but also isolated. Regardless, the sparsity of types of sort s_i induces sparsity of expressions of degree $i - 1$.

▶ **Lemma 19.** *For index i in $\mathcal{I}$, context Γ and expression M, if $\Gamma \vdash M : s_{i-1}$, then M is of the form $\Pi {x_1}^{A_1}. \ldots \Pi {x_k}^{A_k}.\ B$ where $\deg(A_j) \in \mathcal{I}$ for all j and either $B = s_{i-2}$ or $B \in \mathsf{V}_{s_i}$.*

Proof. By induction on the structure of derivations. The cases in which the last inference is an axiom, variable introduction, or weakening are straightforward. The last inference clearly cannot be an abstraction, and it cannot be an application since s_i is irrelevant. What follows are the remaining two cases.

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

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

Since s_{i-1} is $(\mathcal{S}_{\lambda S} \setminus \mathcal{I})$-irrelevant, it must be that $j \in \mathcal{I}$. The desired result holds after applying the inductive hypothesis to the right antecedent judgment.

Conversion. Suppose the last inference is of the form

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash s_{i-1} : s_i}{\Gamma \vdash M : s_{i-1}}$$

where $A =_\beta s_{i-1}$. Note that $\deg(A) = i$ so $\Gamma \vdash A : s_i$ by type correctness. Thus, $A = s_{i-1}$ by Lemma 18, which means the inductive hypothesis can be applied directly to the left antecedent judgment. ◀

▶ **Lemma 20.** *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 21.** *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 [14].

The other wrinkle in defining this translation is that it is difficult to pre-reduce expressions of variable type because even though such types are sparse-inhabitance, it is unclear *a priori* what the value of the expression will be after a series of reductions. By Lemma 20, 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 22.** *The context-indexed family of functions $(\tau_\Gamma : \mathsf{T} \to \mathsf{T})_{\Gamma \in \mathcal{C}}$ as given as follows.*

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

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

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

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

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

where $\bullet_z$ is a distinguished variable. This family of function is extended to a single function on contexts as

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

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

Note that this class of functions is well-defined with respect to the dependence on contexts; each function can be defined simultaneously, inductively on the structure of the input. As for proving the desired features of this translation, first note if $i \in \mathcal{I}$, then the translation maps terms of degree $i-1$ (where $i \in \mathcal{I}$) to canonical atomic terms.

▶ **Proposition 23.** *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 21, for which the above fact clearly holds. This turns out to be a key feature of the translation. Because the translation is able to drop so much information about these expressions, we can pre-reduce redexes in which they appear on the right.

We also use the fact that the context associated with a translation can be weakened when the variable does not appear in its argument.

▶ **Proposition 24.** *For any index i, context Γ, expressions M, A, and B, and variable ${}^{s_i}x$, if $\Gamma \vdash M : A$ and $\Gamma \vdash B : s_i$ then $\tau_{\Gamma,{}^{s_i}x:B}(M) = \tau_\Gamma(M)$.*

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

▶ **Lemma 25.** *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*

$$\tau_\Gamma(M[N/{}^{s_i}x]) = \begin{cases} \tau_{\Gamma,{}^{s_i}x:A}(M) & i \in \mathcal{I} \\ \tau_{\Gamma,{}^{s_i}x:A}(M)[\tau_\Gamma(N)/{}^{s_i}x] & \text{otherwise.} \end{cases}$$

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 18. If $A = s_{i-1}$, then by Proposition 23 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 24 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 26.** *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 27.** *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 25). 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 18 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 21 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 28.** *The **irrelevance reduction** of an n-tiered pure type system λS, denoted here by λS^-, is the n-tiered system specified by the rules*

$$\mathcal{R}_{\lambda S} \setminus \{(s_i, s_j) \mid i \in \mathcal{I} \text{ and } j \in [n]\}.$$

▶ **Lemma 29.** *For context Γ and expressions M and A, if*

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

Proof. By induction on the structure of derivations.
Axiom. If the derivation is a single axiom $\vdash s_i : s_{i+1}$ then the translated derivation is the same axiom.
Variable Introduction. Suppose the last inference is of the form

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

First suppose $i \in \mathcal{I}$. If $A = s_{i-1}$, then $\tau_{\Gamma, x:s_{i-1}}(x) = s_{i-2}$ and $\tau(\Gamma) \vdash s_{i-2} : s_{i-1}$ where $\tau(\Gamma)$ is well-formed by the inductive hypothesis; that is, $\tau(\Gamma) \vdash \tau_\Gamma(A) : s_i$ implies $\tau(\Gamma)$ is well-formed. If A is of the form ${}^{s_{i+1}}y$, then $({}^{s_{i+1}}y : s_{i-1}) \in \Gamma$, which implies $(\bullet_y : {}^{s_{i+1}}y) \in \tau(\Gamma)$ and $\tau(\Gamma) \vdash \bullet_y : {}^{s_{i+1}}y$ where $\tau(\Gamma)$ is again well-formed by the inductive hypothesis.

Next suppose $i - 1 \in \mathcal{I}$ and $A = s_{i-1}$. By the inductive hypothesis, we can derive

$$\frac{\tau(\Gamma) \vdash s_{i-1} : s_i}{\tau(\Gamma), {}^{s_i}x : s_{i-1} \vdash {}^{s_i}x : s_{i-1}}$$

and so by weakening,

$$\frac{\tau(\Gamma), {}^{s_i}x : s_{i-1} \vdash {}^{s_i}x : s_{i-1} \qquad \tau(\Gamma), {}^{s_i}x : s_{i-1} \vdash {}^{s_i}x : s_{i-1}}{\tau(\Gamma), {}^{s_i}x : s_{i-1}, \bullet_x : {}^{s_i}x \vdash {}^{s_i}x : s_{i-1}}$$

The remaining cases are straightforward.
Weakening. Suppose the last inference is of the form

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

By Proposition 23, 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 25, 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 27. ◀

It remains to show that this translation is path-preserving. The guiding observation is that β_2-reductions cannot make more redexes, and what redexes may persist must be simpler in some sense. We define a complexity measure which captures this observation by its being monotonically decreasing in β_2-reductions.

▶ **Definition 30.** *The* ***shallow λ-depth*** *of a term M is the number of top-level λ's appearing in it, i.e., the function $\delta : \mathsf{T} \to \mathbb{N}$ is given by $\delta(\lambda x^A.\ N) \triangleq 1 + \delta(N)$ and $\delta(M) \triangleq 0$ otherwise. The* ***shallow λ-depth of a redex*** *$(\lambda x^A.\ M)N$ is the shallow λ-depth of its left term $\lambda x^A.\ M$.*

I will simply write "depth" from this point forward.

▶ **Definition 31.** *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.,*

$$\mu(s_i) = \mu(x) \triangleq 0$$
$$\mu(\Pi x^A.\ B) = \mu(\lambda x^A.\ B) \triangleq \mu(A) + \mu(B)$$
$$\mu(MN) \triangleq \begin{cases} \mu(M) + \mu(N) + \delta(MN) & MN \text{ is a } \beta_2\text{-redex} \\ \mu(M) + \mu(N) & \text{otherwise.} \end{cases}$$

Finally, we prove the monotonicity lemma. It depends on the typed-version of a result by Lévy for the untyped lambda calculus about the creation of new redexes [10]. I give the statement of the result here without proof. I also include a definition of the standard notion of a *residual* redex, adapted from [17] as well as [8].

▶ **Definition 32.** *Let M and N be expressions such that $M \to_\beta N$ by reducing the redex $(\lambda x^A.\ P)Q$, and let R be a redex in M. The **residuals** of R in N are the copies of R which appear in N after reducing $(\lambda x^A.\ P)Q$. That is,*

- *if R is $(\lambda x^A.\ P)Q$, then R has no residuals.*
- *if $R \subset Q$, then the residuals are R in N are the copies of R in $P[Q/x]$;*
- *if $R \subset P$, then the copy of R in P after substitution of the form $R[Q/x]$ is the residual of R in N;*
- *if the redex $(\lambda x^A.\ P)Q$ is contained in R, then R after substitution (i.e., R with $(\lambda x^A.\ P)Q$ replaced by $P[Q/x]$) is the residual of R in N*
- *if R is disjoint from $(\lambda x^A.\ P)Q$ then R itself is the residual of R in N.*

▶ **Lemma 33.** *(Lévy, 1978) For expressions M and N such that $M \to_\beta N$, if $(\lambda x^A.\ P)Q$ is a redex of N which is not a residual of a redex in M, then it is creates in one of the following ways.*

1. $(\lambda y^B.\ y)(\lambda x^A.\ P)Q \to_\beta (\lambda x^A.\ P)Q$;
2. $(\lambda y^C.\ \lambda x^D.\ R)SQ \to_\beta (\lambda x^{D[S/y]}.\ R[S/y])Q$ *where* $A = D[S/y]$ *and* $P = R[S/y]$;
3. $(\lambda y^B.\ R)(\lambda x^A.\ P) \to_\beta R[\lambda x^A.\ P/y]$ *where yQ is a sub-expression of R.*

▶ **Lemma 34.** *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 21, the expression Q is of the form $\Pi {x_1}^{A_1}. \ldots \Pi {x_k}^{A_k}.\ B$ where $\deg(A_j) \in \mathcal{I}$ for all j and either $B = s_{i-2}$ or $B \in \mathsf{V}_{s_i}$. This means reducing a β_2-redex cannot duplicate existing redexes in M, so every redex has at most one residual in N. Furthermore, if N has a new β_2-redex, it is by item 2 of Lemma 33, *i.e.*, there are expressions C, D, R, and S, and variable z such that $P = \lambda z^D.\ R$ and

$$(\lambda x^C.\ \lambda z^D.\ R)QS \to_\beta (\lambda z^{D[Q/x]}.\ R[Q/x])S.$$

It is easy to verify that, because of the form of Q, only one new β-redex is created and, furthermore, $\delta(R[Q/x]) \leq \delta(R)$. This implies the new redex has smaller depth than the redex that was reduced, so even if it is a β_2-redex, the complexity of M decreases. ◀

The proof of the main theorem of this sub-section is standard.

▶ **Theorem 35.** *If λS^- is strongly normalizing, then λS is strongly normalizing.*

Proof. Suppose there is an infinite reduction sequence in λS

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

where M_1 is derivable from the context Γ. Since μ is monotonically decreasing in β_2-reductions (Lemma 34), 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 27 implies there infinitely many β-reductions in the reduction path

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

which is in λS^- by Lemma 29. ◀

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. The two translations can, in fact, be collapsed into a single translation, but it is convenient to separate it into two parts, especially since the result can be made slightly stronger; the conditions on s_{i-1} are weaker for completely isolation than for complete irrelevancy.

Fix an n-tiered pure type system λS with $n > 2$, and a completely isolated sort s_i.[6] In essence, the following translation removes the completely isolated sort and shifts down all the sorts that might be above it. Because isolated sorts can only really be used to introduce variables into the context, the translation pre-substitutes those variables with dummy values that won't affect the normalization behavior of the expression after translation.

One notable feature of this translation is that it does not preserve the number of sorts in the system and, furthermore, does not preserve degree. Thus, it will be useful to be more careful about variable annotations in the following definitions and lemmas.

▶ **Definition 36.** *The context-indexed family of function* $\{\theta_\Gamma : \mathsf{T} \to \mathsf{T}\}_{\Gamma \in \mathcal{C}}$ *is given as follows.*

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

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

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

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

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

This family of functions is extended to a single function on contexts as

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

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

[6] The restriction on n is a technicality that ensures the target system is nontrivial. See, for example, the variable case of Definition 36.

As with the previous translation, this one is well-defined with respect to its dependence on context, and the contexts can be weakened without changing the value of the function (in analogy with Proposition 24 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 37.** *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 38.** *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 37, keeping in mind that $j \neq i$ since i is isolated, so the lemma can be safely applied. ◀

Finally, typability preservation. The target system is as expected, the isolated sort is removed and potential sorts above it are shifted down.

▶ **Definition 39.** *The **i-collapse** of an n-tiered pure type system λS, denote here by λS^*, is the $(n-1)$-tiered systems specified by the rules*

$$\mathcal{R}_{\lambda S^*} \triangleq \{(\theta_\varnothing(s_j), \theta_\varnothing(s_k)) \mid (s_j, s_k) \in \mathcal{R}_{\lambda S} \text{ and } j \neq i \text{ and } k \neq i\}.$$

▶ **Lemma 40.** *For context Γ and expressions M and A where $M \neq s_{i-1}$, if*

$$\Gamma \vdash M : A \qquad \textit{then} \qquad \theta(\Gamma) \vdash \theta_\Gamma(M) : \theta_\Gamma(A).$$

Proof. By induction on the structure of derivations. The proof differs slightly depending on whether or not s_i is a top-sort. I make clear below which cases differ.
Axiom. Since $M \neq s_{i-1}$, the judgment $\varnothing \vdash \theta_\varnothing(s_j) : \theta_\varnothing(s_{j+1})$ is still an axiom.
Variable Introduction. Suppose the last inference is of the form

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

If $j = i$ and $A = s_{i-1}$, then $\theta(\Gamma) \vdash s_{i-2} : s_{i-1}$ is still derivable. Note that $\theta(\Gamma)$ can be proved to be well-formed by the inductive hypothesis. If $j < i$, then we have

$$\frac{\theta(\Gamma) \vdash \theta_\Gamma(A) : s_j}{\theta(\Gamma), {}^{s_j}x : \theta_\Gamma(A) \vdash {}^{s_j}x : \theta_\Gamma(A)}$$

If $j > i$, then in particular s_i is not a top-sort. This case is then similar to the previous one, keeping in mind that this might use the rule (s_{i-1}, s_i) for the translated derivation *in the system* λS^*, but not in the case that s_i is a top-sort.
Weakening. This case follows directly from the fact that $\theta_{\Gamma,x:B}(M) = \theta_\Gamma(M)$ whenever M and B are derivable from Γ. It is also similar to the analogous case in the previous sub-section.
Product Type Formation. Suppose the last inference is of the form

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

Note that $j \neq i$ and $k \neq i$ since s_i is rule-isolated. In particular, neither A nor B are s_{i-1}. Therefore, we can apply the inductive hypothesis directly to each antecedent judgment and derive the desired consequent judgment.
Abstraction. Suppose the last inference is of the form

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

Note that $j \neq i$ since s_i is rule-isolated, and so $\Pi x^A.\ B$ would not be derivable. Furthermore, $B \neq s_i$ (so $M \neq s_{i-1}$) since s_i is irrelevant. Therefore, we can apply the inductive hypothesis directly to each antecedent judgment and derive the desired consequent judgment.
Application. Suppose the last inference is of the form

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

Note that $\deg(A) \neq i{+}1$ (and in particular $N \neq s_{i-1}$), since s_{i+1} is rule-isolated. Furthermore, $\deg(A) \neq i$ (and $\deg(N) \neq i - 1$) since s_i is rule-isolated. Therefore, we can apply the inductive hypothesis directly to each antecedent judgment.

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

and $\theta_{\Gamma,x:A}(B)[\theta_\Gamma(N)/x] = \theta_\Gamma(B[N/x])$ by Lemma 37.
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 18, in fact $A = B$. If $B = s_{i-1}$, then by Corollary 21 we again have $A = B$. Otherwise, by Lemma 38, $\theta_\Gamma(A) =_\beta \theta_\Gamma(B)$ and we can derive $\theta(\Gamma) \vdash \theta_\Gamma(M) : \theta_\Gamma(B)$ by the inductive hypothesis and conversion. ◀

Since β-reductions are simulated directly, the argument for the final theorem is simple.

▶ **Theorem 41.** *If λS^* is strongly normalizing then λS is strongly normalizing.*

Proof. Suppose there is an infinite reduction path in λS starting at M, which is derivable from the context Γ. Note that this sequence cannot contain the term s_{i-1} since this is a normal form. Therefore, applying θ_Γ to each term yields an infinite reduction path in λS^*. ◀

It is also important to note that weak normalization is preserved in the opposite direction.

▶ **Proposition 42.** *If λS is weakly normalizing then so is λS^* is weakly normalizing.*

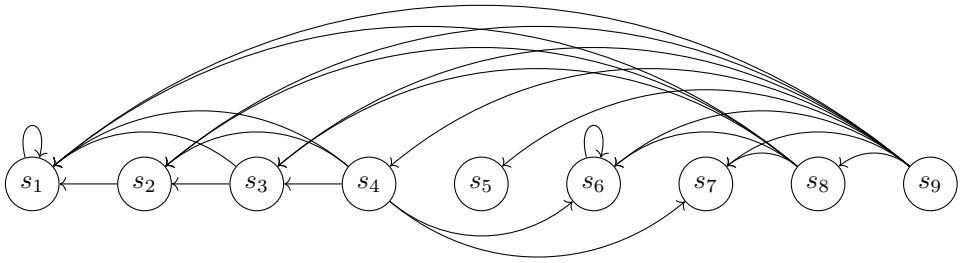


Figure 2 The iteration of irrelevance reductions can be fairly complex; it may be that a sort cannot appear in a completely irrelevant index set until after several iterations. In the system above, the maximum completely irrelevant index set of this system is $\{9\}$, but after eliminating the rules associated with s_9, both s_9 and s_5 become rule-isolated, and so the next maximum completely irrelevant index set is $\{4, 8\}$. It is not difficult to imagine how this effect can be scaled up to larger systems.

This is trivial in the case that the sorts are maintained—as for the irrelevance reduction—and also in the case that the eliminated sort is a top-sort, but requires, in the case of a top-sort-like sort, noting that any term in λS^* can be embedded into λS up to sort renaming.

3.3 The Final Translation

With these two translations, we can now define one final translation, which is the composition of the fixed-points of these to translations.

▶ **Definition 43.** *For any tiered pure type system λS, let $\mathcal{I}_{\lambda S}$ denote its unique maximum completely irrelevant index set and let $i_{\lambda S}$ denote the maximum index of a completely isolated sort in λS, if one exists. Let $\tau(\lambda S)$ denote the fixed-point of taking the irrelevance reduction of λS with respect to its maximum completely irrelevant set $\mathcal{I}_{\lambda S}$ (i.e., until $\mathcal{I}_{\lambda S} = \emptyset$) and let $\theta(\lambda S)$ denote the fixed-point of taking the $i_{\lambda S}$-collapse of λS (i.e., until λS has no completely isolated sort or is 2-tiered). The **irrelevance elimination of** λS, denoted as $\lambda S^{\downarrow}$ is the system $\theta(\tau(\lambda S))$.*

See Figure 2 for an example of this transformation. The main theorem of this paper is as follows. It is simply the observation that the translations from the previous sub-sections can be composed.

▶ **Theorem 44.** *For any tiered pure type system λS, if $\lambda S^{\downarrow}$ is strongly normalizing, then λS is strongly normalizing.*

And, as prefaced above, this result can be bootstrapped with existing results for the Barendregt-Geuvers-Klop conjecture in the expected way.

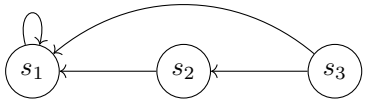
▶ **Corollary 45.** *For any tiered pure type system λS, if weak normalization implies strong normalization for $\lambda S^{\downarrow}$, then weak normalization implies strong normalization for λS.*

Proof. If λS is weakly normalizing, then $\lambda S^{\downarrow}$ is weakly normalizing, since any term in $\lambda S^{\downarrow}$ can be embedded in λS. But then $\lambda S^{\downarrow}$ is strongly normalizing by assumption, which implies λS is strongly normalizing by Theorem 44. ◀

By the result of Barthe *et al.*, if $\lambda S^{\downarrow}$ is non-dependent, clean, and negatable, then weak normalization implies strong normalization in λS. Since $\lambda S^{\downarrow}$ can be non-dependent even if λS has dependent rules, this extension allows us to prove the conjecture of some systems with dependencies. Cleanliness is a technical restriction that I won't go into here, but a pure type system is **negatable** if it has the rule (s_i, s_i) whenever s_i is relevant. So this extension also allows us to consider systems with non-negatable sorts.

4 Conclusions

I have presented a path-preserving translation from a tiered pure type system λS to a weaker system $\lambda S^{\downarrow}$ which is, in essence, λS without its irrelevant structure. When combined with results for the Barendregt-Geuvers-Klop conjecture, it widens the class of systems for which the conjecture applies, most notably to systems with dependent rules, but also to systems with non-negatable sorts. This is a step towards proving the conjecture for all tiered systems, in particular because it highlights those systems which require further analysis. For example, besides dealing with systems that have more dependences, it appears that dealing with circular rules is one of the clear barriers in strengthening these results. For 3-tiered systems, we extend existing results to include the system specified by



but not to the same system with the additional rule (s_3, s_3). Circular rules break irrelevancy and, consequently, induce much more complicated structure in the system. There is also likely other forms of irrelevant structure that are not covered by the translation in this paper.

Additionally, it is worth noting that the conditions on completely irrelevant index sets cannot be trivially weakened. If, for example the irrelevance condition on preceding sorts was removed, this technique would apply to λU (*i.e.*, the same system presented above but with the additional rule (s_2, s_2)), leading to a contradiction since λU is non-normalizing. Circular rules again seem to be at the core of this issue. More carefully considering λU and related non-normalizing systems through the lens of these results—particularly why the techniques don't apply to these systems—may yield a more structural understanding of the non-normalization of λU, independent of Girard's paradox. Regardless, I hope to have demonstrated with this translation that, despite the full Barendregt-Geuvers-Klop conjecture seeming quite far from being solved, there are still a number of approachable questions and avenues for further development.

References

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 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.

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

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

9 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.

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

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

12 Nathan Mull. Strong Normalization from Weak Normalization in Non-Dependent Pure Type Systems via Thunkification. Research Report, `https://nmmull.github.io/thunk.pdf`, 2022.

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

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

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

16 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.

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