A Generalized Translation of Pure Type Systems

Nathan Mull

University of Chicago
Department of Computer Science
June 22, 2022

TYPES 2022

Overview

- Background
  - (Tiered) Pure Type Systems
  - The Barendregt-Geuvers-Klop Conjecture
- Dependency Eliminating Translations
- Conclusions/Open Questions

Background

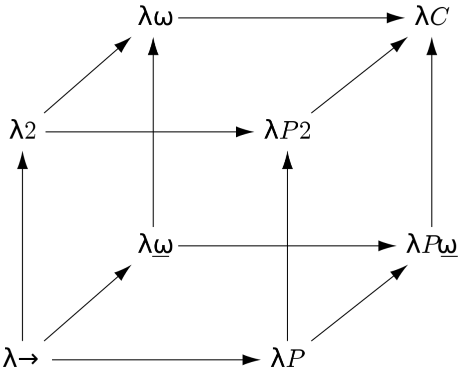
The Lambda Cube (1/3)

The lambda cube is introduced by Barendregt around 1990 as a way of classifying a collection of type systems based on the common features they had (the kinds of abstractions they allowed).

$\uparrow$ arrows give terms depending on types (*polymorphism*)

$\rightarrow$ arrows give types depending on terms (*dependent types*)

$\nearrow$ arrows give types depending on types (*type constructors*)



The Lambda Cube (2/3)

The Grammar

$$\mathsf{T} ::= \star \mid \square \mid \mathsf{V} \mid \mathsf{TT} \mid \lambda \mathsf{V}^{\mathsf{T}}\mathsf{T} \mid \Pi \mathsf{V}^{\mathsf{T}}\mathsf{T}$$

The Derivation Rules

$$(\mathsf{Ax})\ \vdash \star : \square \qquad (\mathsf{Vr})\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A} \qquad (\mathsf{Wk})\frac{\Gamma \vdash M : A \quad \Gamma \vdash B : s}{\Gamma, x : B \vdash M : A}$$

$$(\mathsf{Fn})\frac{\Gamma \vdash A : s \quad \Gamma, x : A \vdash B : s'}{\Gamma \vdash \Pi x^A B : s'}\ (s, s') \in \mathcal{R}$$

$$(\mathsf{Ab})\frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash \Pi x^A B : s}{\Gamma \vdash \lambda x^A M : \Pi x^A B}$$

$$(\mathsf{Ap})\frac{\Gamma \vdash M : \Pi x^A B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B[N/x]} \qquad (\mathsf{Cv})\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s}{\Gamma \vdash M : B}\ A =_\beta B$$

The Lambda Cube (3/3)

$(\Box, \star)$ gives terms depending on types (*polymorphism*)

$(\star, \Box)$ gives types depending on terms (*dependent types*)

$(\Box, \Box)$ gives types depending on types (*type constructors*)

System	Rules ($\mathcal{R}$)
$\lambda_{\rightarrow}$	$(\star, \star)$
$\lambda 2$	$(\star, \star)$, $(\Box, \star)$
λP	$(\star, \star)$, $(\star, \Box)$
$\lambda \underline{\omega}$	$(\star, \star)$, $(\Box, \Box)$
$\lambda \omega$	$(\star, \star)$, $(\Box, \star)$, $(\Box, \Box)$
$\lambda P2$	$(\star, \star)$, $(\Box, \star)$, $(\star, \Box)$
$\lambda P\underline{\omega}$	$(\star, \star)$, $(\star, \Box)$, $(\Box, \Box)$
λC	$(\star, \star)$, $(\star, \Box)$, $(\Box, \star)$, $(\Box, \Box)$

Pure Type Systems

Pure Type Systems (PTSs) are introduced by Terlouw, Berardi, and Barendregt also around 1990. A PTS is specified by a set of sorts ($\mathcal{S}$), axioms ($\mathcal{A} \subset \mathcal{S} \times \mathcal{S}$) and rules ($\mathcal{R} \subset \mathcal{S} \times \mathcal{S} \times \mathcal{S}$).

The Grammar

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

The Derivation Rules

$$(\mathsf{Ax})\ \vdash s : s' \quad (s, s') \in \mathcal{A} \qquad (\mathsf{Vr})\frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A} \qquad (\mathsf{Wk})\frac{\Gamma \vdash M : A \quad \Gamma \vdash B : s}{\Gamma, x : B \vdash M : A}$$

$$(\mathsf{Fn})\frac{\Gamma \vdash A : s \quad \Gamma, x : A \vdash B : s'}{\Gamma \vdash \Pi x^A B : s''} \quad (s, s', s'') \in \mathcal{R}$$

$$(\mathsf{Ab})\frac{\Gamma, x : A \vdash M : B \qquad \Gamma \vdash \Pi x^A B : s}{\Gamma \vdash \lambda x^A M : \Pi x^A B}$$

$$(\mathsf{Ap})\frac{\Gamma \vdash M : \Pi x^A B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B[N/x]} \qquad (\mathsf{Cv})\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s}{\Gamma \vdash M : B} \quad A =_\beta B$$

The Barendregt-Geuvers-Klop Conjecture

A PTS is weak normalizing (WN) if all typable terms have normal forms, *i.e.*, if $\Gamma \vdash M : A$, then $M \twoheadrightarrow_\beta N$ where N is a normal form.

The Barendregt-Geuvers-Klop Conjecture

A PTS is weak normalizing (WN) if all typable terms have normal forms, *i.e.*, if $\Gamma \vdash M : A$, then $M \twoheadrightarrow_\beta N$ where N is a normal form.

A PTS is strongly normalizing (SN) if there is no infinite β-reduction sequence of typable terms, ie, if $\Gamma \vdash M : A$, then *every* reduction path starting from M terminates at a normal form.

The Barendregt-Geuvers-Klop Conjecture

A PTS is weak normalizing (WN) if all typable terms have normal forms, *i.e.*, if $\Gamma \vdash M : A$, then $M \twoheadrightarrow_\beta N$ where N is a normal form.

A PTS is strongly normalizing (SN) if there is no infinite β-reduction sequence of typable terms, ie, if $\Gamma \vdash M : A$, then *every* reduction path starting from M terminates at a normal form.

Conjecture. WN implies SN for *every* pure type system.

The Barendregt-Geuvers-Klop Conjecture

A PTS is weak normalizing (WN) if all typable terms have normal forms, *i.e.*, if $\Gamma \vdash M : A$, then $M \twoheadrightarrow_\beta N$ where N is a normal form.

A PTS is strongly normalizing (SN) if there is no infinite β-reduction sequence of typable terms, ie, if $\Gamma \vdash M : A$, then *every* reduction path starting from M terminates at a normal form.

Conjecture. WN implies SN for *every* pure type system.

This is motivated primarily by a collection of results that derive SN from WN from systems in the lambda cube.

Why is this problem so hard?

PTSs are in some sense the most obvious generalization of the lambda cube, but they're too unwieldy (*e.g.*, they lose many good meta-theoretic properties).

Why is this problem so hard?

PTSs are in some sense the most obvious generalization of the lambda cube, but they're too unwieldy (*e.g.*, they lose many good meta-theoretic properties).

We don't actually have that many examples of PTSs with more than a couple sorts, and certainly not ones with bizarro structure.

Why is this problem so hard?

PTSs are in some sense the most obvious generalization of the lambda cube, but they're too unwieldy (*e.g.*, they lose many good meta-theoretic properties).

We don't actually have that many examples of PTSs with more than a couple sorts, and certainly not ones with bizarro structure.

An Idea. Consider a subclass of PTSs which generalize the lambda cube but maintain useful meta-theoretic properties.

Tiered Pure Type Systems (1/2)

An *n*-tiered PTS is specified by

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

Tiered Pure Type Systems (1/2)

An n-tiered PTS is specified by

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

Classification Lemma. There is a measure $\deg$ on terms s.t.

- $\deg A = n + 1$ if and only if $A = s_n$
- $\deg A = n$ if and only if $\Gamma \vdash A : s_n$ for some context Γ
- For $i \in [n-1]$, we have $\deg A = i$ if and only if $\Gamma \vdash A : B$ and $\Gamma \vdash B : s_{i+1}$ for some context Γ and term B

Tiered Pure Type Systems (1/2)

An n-tiered PTS is specified by

$$\begin{aligned}\mathcal{S} &= [n] \\ \mathcal{A} &= \{(i, i+1) \mid i \in [n-1]\} \\ \mathcal{R} &\subset \{(i, j, j) \mid (i, j) \in \mathcal{S} \times \mathcal{S}\}\end{aligned}$$

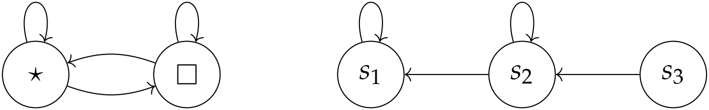
Classification Lemma. There is a measure deg on terms s.t.

- $\deg A = n + 1$ if and only if $A = s_n$
- $\deg A = n$ if and only if $\Gamma \vdash A : s_n$ for some context Γ
- For $i \in [n-1]$, we have $\deg A = i$ if and only if $\Gamma \vdash A : B$ and $\Gamma \vdash B : s_{i+1}$ for some context Γ and term B

With respect to normalization, these are equivalent to stratified persistent pure type systems, but simpler and more explicitly presented.

Tiered Pure Type Systems (2/2)

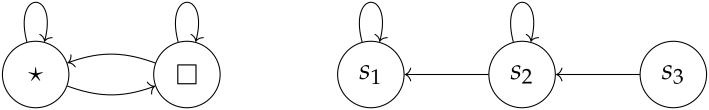
A Graphical Representation



$$\mathcal{R}_{\lambda C} = \{(\star,\star),(\star,\square),(\square,\star),(\square,\square)\}$$
$$\mathcal{R}_{\lambda U^-} = \{(s_1,s_1),(s_2,s_2),(s_2,s_1),(s_3,s_2)\}$$

Tiered Pure Type Systems (2/2)

A Graphical Representation



$$\mathcal{R}_{\lambda C} = \{(\star,\star),(\star,\Box),(\Box,\star),(\Box,\Box)\}$$
$$\mathcal{R}_{\lambda U^-} = \{(s_1,s_1),(s_2,s_2),(s_2,s_1),(s_3,s_2)\}$$

An Alternative Question. Does the BGK conjecture hold for tiered PTSs? Furthermore, can we classify those tiered PTSs that are SN?

Girard's Paradox

Theorem. *(Girard, Coquand)* λU^- is not WN. In particular, there is a term of type $\Pi x^{s_1} x$ so all types (of sort s_1) are inhabited.

Girard's Paradox

Theorem. *(Girard, Coquand)* λU^- is not WN. In particular, there is a term of type $\Pi x^{s_1} x$ so all types (of sort s_1) are inhabited.

Girard proves this by encoding the Burali-Forti paradox in λU and Coquand improves this to λU^- be encoding Reynold's Theorem.

Girard's Paradox

Theorem. *(Girard, Coquand)* λU^- is not WN. In particular, there is a term of type $\Pi x^{s_1} x$ so all types (of sort s_1) are inhabited.

Girard proves this by encoding the Burali-Forti paradox in λU and Coquand improves this to λU^- be encoding Reynold's Theorem.

Question. λU^- is tiered. Are there any other non-normalizing tiered PTSs?

Girard's Paradox

Theorem. *(Girard, Coquand)* λU^- is not WN. In particular, there is a term of type $\Pi x^{s_1} x$ so all types (of sort s_1) are inhabited.

Girard proves this by encoding the Burali-Forti paradox in λU and Coquand improves this to λU^- be encoding Reynold's Theorem.

Question. λU^- is tiered. Are there any other non-normalizing tiered PTSs?

Girard's paradox implies that the question on the previous slide should have a fairly interesting answer.

Dependency Eliminating Translations

Objectives

- Outline existing generalizations of normalization results to the PTS setting
- Discuss dependency eliminating translations
- Briefly touch on the challenge of generalizing these translations
- Describe the current state of my work, with an application to the stronger version of the BGK conjecture

Normalization in the PTS Setting

Lou proved that ECC is SN (a reasonable subsystem of ECC can be viewed as a PTS).

Normalization in the PTS Setting

Lou proved that ECC is SN (a reasonable subsystem of ECC can be viewed as a PTS).

Mellèis and Werner generalized the Λ-set method of Altenkirch for a generic strong normalization proof for PTSs.

Normalization in the PTS Setting

Lou proved that ECC is SN (a reasonable subsystem of ECC can be viewed as a PTS).

Mellèis and Werner generalized the Λ-set method of Altenkirch for a generic strong normalization proof for PTSs.

Barthe, Hatcliff and Sørensen extended CPS translations to a class of non-dependent PTSs and showed that the BGK conjecture holds on this class of systems.

Normalization in the PTS Setting

Lou proved that ECC is SN (a reasonable subsystem of ECC can be viewed as a PTS).

Mellèis and Werner generalized the Λ-set method of Altenkirch for a generic strong normalization proof for PTSs.

Barthe, Hatcliff and Sørensen extended CPS translations to a class of non-dependent PTSs and showed that the BGK conjecture holds on this class of systems.

The Goal of this Work. Generalize the dependency eliminating translation of Geuvers and Nederhof (based on the one of Harper) to a class of PTSs.

Dependency Eliminating Translations

To prove λS is SN, it suffices to give a translation of terms that is infinite-reduction-path preserving and typability-preserving into a weaker system that is SN.

Dependency Eliminating Translations

To prove λS is SN, it suffices to give a translation of terms that is infinite-reduction-path preserving and typability-preserving into a weaker system that is SN.

Theorem. *(Harper)* λP is SN if $\lambda_{\rightarrow}$ is SN.

Theorem. *(Geuvers, Nederhof)* λC is SN if $\lambda \omega$ is SN.

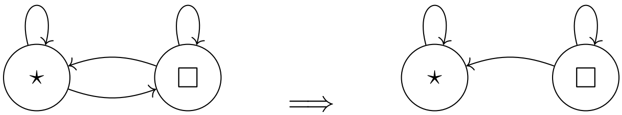
Dependency Eliminating Translations

To prove λS is SN, it suffices to give a translation of terms that is infinite-reduction-path preserving and typability-preserving into a weaker system that is SN.

Theorem. *(Harper)* λP is SN if $\lambda_{\rightarrow}$ is SN.

Theorem. *(Geuvers, Nederhof)* λC is SN if $\lambda\omega$ is SN.

Both of the translations used in these results can be seen as eliminating the dependent rules in the system, the forward facing arrows in the graphic representation, *e.g.*,



The Approach

Basic Idea. Map all types and kinds down to terms of a fixed type variable (call it 0).

The Approach

Basic Idea. Map all types and kinds down to terms of a fixed type variable (call it 0).

This eliminates the need for dependent rules; given $\Pi x^A B$, the translated term will have B replaced with 0, which only requires the rule $(s, \star)$.

The Approach

Basic Idea. Map all types and kinds down to terms of a fixed type variable (call it 0).

This eliminates the need for dependent rules; given $\Pi x^A B$, the translated term will have B replaced with 0, which only requires the rule $(s, \star)$.

This approach can be used in the generalization as well.

Main Challenges.

1. preserving well-formedness of the types of translated terms, which may themselves have terms to be typed

The Approach

Basic Idea. Map all types and kinds down to terms of a fixed type variable (call it 0).

This eliminates the need for dependent rules; given $\Pi x^A B$, the translated term will have B replaced with 0, which only requires the rule $(s, \star)$.

This approach can be used in the generalization as well.

Main Challenges.

1. preserving well-formedness of the types of translated terms, which may themselves have terms to be typed
2. preserving infinite reduction paths by not losing subterm information in the translation

The Approach

Basic Idea. Map all types and kinds down to terms of a fixed type variable (call it 0).

This eliminates the need for dependent rules; given $\Pi x^A B$, the translated term will have B replaced with 0, which only requires the rule $(s, \star)$.

This approach can be used in the generalization as well.

Main Challenges.

1. preserving well-formedness of the types of translated terms, which may themselves have terms to be typed
2. preserving infinite reduction paths by not losing subterm information in the translation
3. dealing with uses of $\perp$, which won't be derivable at all levels (only a problem for the generalization)

Challenge 1: Preserving Typability

We can't translate terms down to a fixed type 0 right off the bat because we have to know that these terms are typeable, that types are themselves typeable, and so on.

Challenge 1: Preserving Typability

We can't translate terms down to a fixed type 0 right off the bat because we have to know that these terms are typeable, that types are themselves typeable, and so on.

For λC. Define *three* translations ρ_1, ρ_0 and γ such that

$$\Gamma \vdash A : s \quad \text{implies} \quad \rho_1(\Gamma) \vdash \rho_0(A) : \star$$
$$\Gamma \vdash M : A \quad \text{implies} \quad \rho_0(\Gamma) \vdash \gamma(M) : \rho_0(A)$$

Challenge 1: Preserving Typability

We can't translate terms down to a fixed type 0 right off the bat because we have to know that these terms are typeable, that types are themselves typeable, and so on.

For λC. Define *three* translations ρ_1, ρ_0 and γ such that

$$\Gamma \vdash A : s \quad \text{implies} \quad \rho_1(\Gamma) \vdash \rho_0(A) : \star$$
$$\Gamma \vdash M : A \quad \text{implies} \quad \rho_0(\Gamma) \vdash \gamma(M) : \rho_0(A)$$

For the generalization. Define $n + 1$ translation inductively,

$$\rho_{n-1}, \ldots, \rho_0, \gamma$$

where $\rho_i(s_k) = s_i$ and $\gamma(s_k) = 0$ for some variables 0 of sort s_1. Then use the previously defined translations to type the subsequently defined translations.

Challenge 2: Preserving Infinite Reductions (1/2)

Consider translating a predicate on A.[1] We translate $\star$ to 0 and translate functions and applications inductively on structure.

A Dependent Case

$$\frac{\Gamma \vdash P : A \to \star \qquad \Gamma \vdash M : A}{\Gamma \vdash PM : \star}$$

[1]These example are taken from Geuvers and Nederhof.

Challenge 2: Preserving Infinite Reductions (1/2)

Consider translating a predicate on A.[1] We translate $\star$ to 0 and translate functions and applications inductively on structure.

A Dependent Case

$$\frac{\Gamma \vdash P : A \to \star \qquad \Gamma \vdash M : A}{\Gamma \vdash PM : \star}$$

After Translation

$$\frac{\rho_0(\Gamma) \vdash \gamma(P) : \rho_0(A) \to 0 \qquad \rho_0(\Gamma) \vdash \gamma(M) : \rho_0(A)}{\rho_0(\Gamma) \vdash \gamma(P)\gamma(M) : 0}$$

[1] These example are taken from Geuvers and Nederhof.

Challenge 2: Preserving Infinite Reductions (2/2)

In the non-dependent case, polymorphism makes things tricky.

A Non-Dependent Case

$$\frac{\Gamma \vdash P : \Pi A^{\star} A \to A \qquad \Gamma \vdash B : \star}{\Gamma \vdash PB : B \to B}$$

Challenge 2: Preserving Infinite Reductions (2/2)

In the non-dependent case, polymorphism makes things tricky.

A Non-Dependent Case

$$\frac{\Gamma \vdash P : \Pi A^{\star} A \to A \qquad \Gamma \vdash B : \star}{\Gamma \vdash PB : B \to B}$$

After Translation

$$\frac{\rho_0(\Gamma) \vdash \gamma(P) : ??? \qquad \rho_0(\Gamma) \vdash \gamma(B) : 0}{\rho_0(\Gamma) \vdash \gamma(P)\gamma(B) : \rho_0(B) \to \rho_0(B)}$$

Challenge 2: Preserving Infinite Reductions (2/2)

In the non-dependent case, polymorphism makes things tricky.

A Non-Dependent Case

$$\frac{\Gamma \vdash P : \Pi A^{\star} A \to A \qquad \Gamma \vdash B : \star}{\Gamma \vdash PB : B \to B}$$

After Translation

$$\frac{\rho_0(\Gamma) \vdash \gamma(P) : ??? \qquad \rho_0(\Gamma) \vdash \gamma(B) : 0}{\rho_0(\Gamma) \vdash \gamma(P)\gamma(B) : \rho_0(B) \to \rho_0(B)}$$

The Problem. We need to pass in $\gamma(B)$ so we don't lose subterm information, but how do we get $\rho_0(B)$ from $\gamma(B)$? What would the type of $\gamma(P)$ have to be?

Updated Translation

$$\frac{\rho_0(\Gamma) \vdash \gamma(P) : \Pi A^\star \Pi x^0 A \to A \qquad \rho_0(\Gamma) \vdash \gamma(B) : 0}{\rho_0(\Gamma) \vdash \gamma(P)\rho_0(B)\gamma(B) : \rho_0(B) \to \rho_0(B)}$$

Updated Translation

$$\frac{\rho_0(\Gamma) \vdash \gamma(P) : \Pi A^{\star} \Pi x^0 A \to A \qquad \rho_0(\Gamma) \vdash \gamma(B) : 0}{\rho_0(\Gamma) \vdash \gamma(P)\rho_0(B)\gamma(B) : \rho_0(B) \to \rho_0(B)}$$

For the generalization. We need to include more and more terms in each subsequent translation in order not to lose subterm information with the non-dependent case. *This puts a very strong requirement on the non-dependent part of the PTS.*

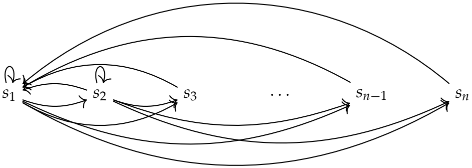
Full Systems

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

Full Systems

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

Fullness is a *very* strong restriction. Most full systems contain a λU^- (so are non-normalizing) and the small class of systems that do not contain λU^- are subsystems of ECC. The strongest non-trivial system is the following.



Challenge 3: Dealing with $\bot$ (1/2)

For any tiered PTS λS, define its non-dependent restriction, denoted λS^*, to be the tiered system with rules $\{(s_i, s_j, s_j) \in \mathcal{R}_{\lambda S} \mid i \leq j\}$.

Challenge 3: Dealing with $\bot$ (1/2)

For any tiered PTS λS, define its non-dependent restriction, denoted λS^*, to be the tiered system with rules $\{(s_i, s_j, s_j) \in \mathcal{R}_{\lambda S} \mid i \leq j\}$.

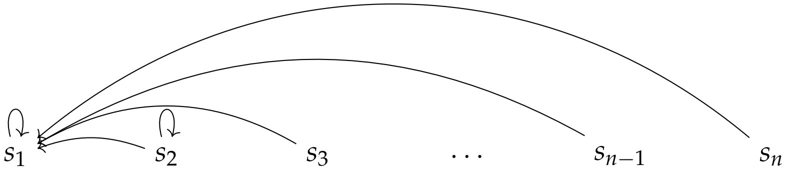
In the last stage of the translation for λC a term of type $\Pi x^{s_1} x$ is put in the context to be able to derive dummy terms for the case of translating Π-terms. This is possible because

$$\vdash_{\lambda C} \Pi x^{s_1} x : s_2$$

But in general, $\Pi x^{s_i} x$ will not be a derivable type in λS^*.

Challenge 3: Dealing with ⊥ (2/2)

The idea. All higher sorts are very scarcely inhabited, *e.g.*,



So we define one more intermediate translation that eliminates the uses of ⊥'s in these cases.

Results

Theorem. For any full tiered PTS λS, there are two functions $\tau : \mathsf{T} \to \mathsf{T}$ and $[\![-]\!] : \mathsf{T} \to \mathsf{T}$ on terms such that the following hold.

1. If $\Gamma \vdash A : B$ then there is a context Γ' such that $\Gamma' \vdash_{\lambda S^*} [\![A]\!] : \tau(B)$. That is, $[\![-]\!]$ preserves typability.
2. For any term A typable in λS, if $A \twoheadrightarrow_\beta B$, then $[\![A]\!] \twoheadrightarrow_\beta^+ [\![B]\!]$. In particular, $[\![-]\!]$ preserves infinite reduction paths.

Results

Theorem. For any full tiered PTS λS, there are two functions $\tau : \mathsf{T} \to \mathsf{T}$ and $[\![-]\!] : \mathsf{T} \to \mathsf{T}$ on terms such that the following hold.

1. If $\Gamma \vdash A : B$ then there is a context Γ' such that $\Gamma' \vdash_{\lambda S^*} [\![A]\!] : \tau(B)$. That is, $[\![-]\!]$ preserves typability.
2. For any term A typable in λS, if $A \twoheadrightarrow_\beta B$, then $[\![A]\!] \twoheadrightarrow_\beta^+ [\![B]\!]$. In particular, $[\![-]\!]$ preserves infinite reduction paths.

Corollary. Let λS be a full PTS. Then λS is SN if λS^* is SN.

Results

Theorem. For any full tiered PTS λS, there are two functions $\tau : \mathsf{T} \to \mathsf{T}$ and $[\![-]\!] : \mathsf{T} \to \mathsf{T}$ on terms such that the following hold.

1. If $\Gamma \vdash A : B$ then there is a context Γ' such that $\Gamma' \vdash_{\lambda S^*} [\![A]\!] : \tau(B)$. That is, $[\![-]\!]$ preserves typability.
2. For any term A typable in λS, if $A \twoheadrightarrow_\beta B$, then $[\![A]\!] \twoheadrightarrow_\beta^+ [\![B]\!]$. In particular, $[\![-]\!]$ preserves infinite reduction paths.

Corollary. Let λS be a full PTS. Then λS is SN if λS^* is SN.

Corollary. For any full PTS, WN is equivalent to SN *is provable in Peano Arithemtic*.

A Simple Bootstrapping Argument (Xi)

1. If λS is WN then so is λS^*.

A Simple Bootstrapping Argument (Xi)

1. If λS is WN then so is λS^*.
2. Since λS^* is non-dependent, if the CPS translation of Barthe et al. can be applied to it, then λS^* is SN.

A Simple Bootstrapping Argument (Xi)

1. If λS is WN then so is λS^*.
2. Since λS^* is non-dependent, if the CPS translation of Barthe et al. can be applied to it, then λS^* is SN.
3. Using the dependency eliminating translation, it follows that λS is SN.

A Simple Bootstrapping Argument (Xi)

1. If λS is WN then so is λS^*.
2. Since λS^* is non-dependent, if the CPS translation of Barthe et al. can be applied to it, then λS^* is SN.
3. Using the dependency eliminating translation, it follows that λS is SN.

The Takeaway. With better translations, both non-dependent systems and from dependent to non-dependent systems, we can expand the class of systems to which the BGK conjecture holds.

The Translation (1/2)

Sorts	$\rho_i(s_j) \triangleq \begin{cases} 0 & i = 0 \\ s_i & \text{otherwise} \end{cases}$ (where $j \geq i$)
Var.	$\rho_i({}^{s_j}x) \triangleq {}^{s_{i+2}}x$ (where $j \geq i + 2$)
Π-*terms*	$\rho_i(\Pi x^A B) \triangleq \begin{cases} \Pi x^{\rho_{\deg A - 1}(A)} \ldots \Pi x^{\rho_i(A)} \rho_i(B) & \deg A \geq i + 1 \\ \rho_i(B) & \text{otherwise} \end{cases}$
λ-*terms*	$\rho_i(\lambda x^A M) \triangleq \begin{cases} \lambda x^{\rho_{\deg A - 1}(A)} \ldots \lambda x^{\rho_{i+1}(A)} \rho_i(M) & \deg A \geq i + 2 \\ \rho_i(M) & \text{otherwise} \end{cases}$
App.	$\rho_i(MN) \triangleq \begin{cases} \rho_i(M) \rho_{\deg N - 1}(N) \ldots \rho_i(N) & \deg N \geq i + 1 \\ \rho_i(M) & \text{otherwise} \end{cases}$

The Translation (2/2)

Sorts	$\gamma(s_i) \triangleq c^0$
Variables	$\gamma({}^{s_i}x) = {}^{s_1}x$
Π-*terms*	$\gamma\Pi x^A B) \triangleq$ $c^{0\to 0\to 0}\gamma(A)(\gamma(B)[c^{\rho_{\deg A-1}(A)}/{}^{s_{\deg A}}x]\ldots[c^{\rho_0(A)}/{}^{s_1}x])$
λ-*terms*	$\gamma(\lambda x^A M) \triangleq (\lambda z^0 \lambda x^{\rho_{\deg A-1}(A)} \ldots \lambda x^{\rho_0(A)}\gamma(M))\gamma(A)$
App.	$\gamma(MN) \triangleq \begin{cases} \gamma(M)\,\rho_{\deg N-1}(N)\ldots\rho_0(N)\gamma(N) & \deg N \geq 1 \\ \gamma(M)\gamma(N) & \deg N = 0 \end{cases}$

(I have not included the last stage which eliminates the uses of $\perp$-terms)

Conclusions

Open Questions

- What is the normalization behavior of tiered PTSs? Even the case of $n = 3$ is not known, I would guess λS is SN if and only if it does not contain λU^-.

Open Questions

- What is the normalization behavior of tiered PTSs? Even the case of $n = 3$ is not known, I would guess λS is SN if and only if it does not contain λU^-.
- Are there any more examples of non-normalizing tiered PTSs besides λU^-?

Open Questions

- What is the normalization behavior of tiered PTSs? Even the case of $n = 3$ is not known, I would guess λS is SN if and only if it does not contain λU^-.
- Are there any more examples of non-normalizing tiered PTSs besides λU^-?
- What other normalization techniques are amenable to the PTS setting?
  - perpetual reductions
  - β-monotonic measures
  - thunkification translation
  - category theoretic arguments
  - normalization by evaluation

Open Questions

- What is the normalization behavior of tiered PTSs? Even the case of $n = 3$ is not known, I would guess λS is SN if and only if it does not contain λU^-.
- Are there any more examples of non-normalizing tiered PTSs besides λU^-?
- What other normalization techniques are amenable to the PTS setting?
  - perpetual reductions
  - β-monotonic measures
  - thunkification translation
  - category theoretic arguments
  - normalization by evaluation
- Can dependency eliminating translations be combined with double negation (CPS) translations to weaken the fullness requirement?

Thank You