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 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).

↑ arrows give terms depending on types (polymorphism)

→ arrows give types depending on terms (dependent types)

↗ arrows give types depending on types (type constructors)
The Lambda Cube (2/3)

The Grammar

\[ T ::= \star | \square | V | TT | \lambda V^T T | \Pi V^T T \]

The Derivation Rules

(Ax) \quad \Gamma \vdash \star : \square

(Vr) \quad \frac{\Gamma \vdash A : s}{\Gamma, x : A \vdash x : A}

(Wk) \quad \frac{\Gamma \vdash M : A \quad \Gamma \vdash B : s}{\Gamma, x : B \vdash M : A}

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

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

(Ap) \quad \frac{\Gamma \vdash M : \Pi x^A B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B[N/x]}

(Cv) \quad \frac{\Gamma \vdash M : A \quad \Gamma \vdash B : s}{\Gamma \vdash M : B} \quad A = \beta B
\((\lozenge, \star)\) gives \textbf{terms} depending on \textbf{types} \\
(\textit{polymorphism})

\((\star, \lozenge)\) gives \textbf{types} depending on \textbf{terms} \\
(\textit{dependent types})

\((\lozenge, \lozenge)\) gives \textbf{types} depending on \textbf{types} \\
(\textit{type constructors})

<table>
<thead>
<tr>
<th>System</th>
<th>Rules ((\mathcal{R}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\lambda_\rightarrow)</td>
<td>((\star, \star))</td>
</tr>
<tr>
<td>(\lambda 2)</td>
<td>((\star, \star), (\lozenge, \star))</td>
</tr>
<tr>
<td>(\lambda P)</td>
<td>((\star, \star), (\star, \lozenge))</td>
</tr>
<tr>
<td>(\lambda \omega)</td>
<td>((\star, \star), (\lozenge, \lozenge))</td>
</tr>
<tr>
<td>(\lambda \omega)</td>
<td>((\star, \star), (\lozenge, \star), (\lozenge, \lozenge))</td>
</tr>
<tr>
<td>(\lambda P2)</td>
<td>((\star, \star), (\lozenge, \star), (\star, \lozenge))</td>
</tr>
<tr>
<td>(\lambda P\omega)</td>
<td>((\star, \star), (\star, \lozenge), (\lozenge, \lozenge))</td>
</tr>
<tr>
<td>(\lambda C)</td>
<td>((\star, \star), (\star, \lozenge), (\lozenge, \star), (\lozenge, \lozenge))</td>
</tr>
</tbody>
</table>
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 ($S$), axioms ($A \subset S \times S$) and rules ($R \subset S \times S \times S$).

The Grammar

$$T ::= S \mid V \mid TT \mid \lambda V^T T \mid \Pi V^T T$$

The Derivation Rules

(Ax) $\vdash s : s' \ (s, s') \in A$

(Vr) $\Gamma \vdash A : s$

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

(Wk) $\Gamma \vdash M : A \quad \Gamma \vdash B : s$

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

(Fn) $\Gamma \vdash A : s \quad \Gamma, x : A \vdash B : s'$

$$\frac{}{\Gamma \vdash \Pi x^A B : s''}$$

(s, s', s'') $\in R$

(Ab) $\Gamma, x : A \vdash M : B \quad \Gamma \vdash \Pi x^A B : s$

$$\frac{}{\Gamma \vdash \lambda x^A M : \Pi x^A B}$$

(Ap) $\Gamma \vdash M : \Pi x^A B \quad \Gamma \vdash N : A$

$$\frac{}{\Gamma \vdash MN : B[N/x]}$$

(Cv) $\Gamma \vdash M : A \quad \Gamma \vdash B : s$

$$\frac{}{A =_\beta B}$$
A PTS is **weak normalizing** (WN) if all typable terms have normal forms, *i.e.*, if $\Gamma \vdash M : A$, then $M \rightarrow^\beta N$ where $N$ is a normal form.
A PTS is **weak normalizing** (WN) if all typable terms have normal forms, *i.e.*, if $\Gamma \vdash M : A$, then $M \rightarrow^{\beta} N$ where $N$ is a normal form.

A PTS is **strongly normalizing** (SN) if there is no infinite $\beta$-reduction sequence of typable terms, *i.e.*, 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.
A PTS is **weak normalizing** (WN) if all typable terms have normal forms, *i.e.*, if $\Gamma \vdash M : A$, then $M \rightarrow^\beta N$ where $N$ is a normal form.

A PTS is **strongly normalizing** (SN) if there is no infinite $\beta$-reduction sequence of typable terms, *i.e.*, 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.
A PTS is **weak normalizing** (WN) if all typable terms have normal forms, *i.e.*, if $\Gamma \vdash M : A$, then $M \rightarrow^\beta N$ where $N$ is a normal form.

A PTS is **strongly normalizing** (SN) if there is no infinite $\beta$-reduction sequence of typable terms, *i.e.*, 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?

PTTs 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 PTTs with more than a couple sorts, and certainly not ones with bizarro structure.
Why is this problem so hard?

PTTs 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 PTTs with more than a couple sorts, and certainly not ones with bizarro structure.

An Idea. Consider a subclass of PTTs which generalize the lambda cube but maintain useful meta-theoretic properties.
An $n$-tiered PTS is specified by

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

$$S = [n]$$

$$A = \{(i, i + 1) \mid i \in [n - 1]\}$$

$$R \subseteq \{(i, j, j) \mid (i, j) \in S \times S\}$$

**Classification Lemma.** There is a measure $\text{deg}$ on terms s.t.

- $\text{deg } A = n + 1$ if and only if $A = s_n$
- $\text{deg } A = n$ if and only if $\Gamma \vdash A : s_n$ for some context $\Gamma$
- For $i \in [n - 1]$, we have $\text{deg } A = i$ if and only if $\Gamma \vdash A : B$ and $\Gamma \vdash B : s_{i+1}$ for some context $\Gamma$ and term $B$
An \( n \)-tiered PTS is specified by

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

**Classification Lemma.** There is a measure \( \text{deg} \) on terms s.t.

\begin{itemize}
  \item \( \text{deg} A = n + 1 \) if and only if \( A = s_n \)
  \item \( \text{deg} A = n \) if and only if \( \Gamma \vdash A : s_n \) for some context \( \Gamma \)
  \item For \( i \in [n - 1] \), we have \( \text{deg} A = i \) if and only if \( \Gamma \vdash A : B \) and \( \Gamma \vdash B : s_{i+1} \) for some context \( \Gamma \) and term \( B \)
\end{itemize}

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

\[ R_{\lambda C} = \{ (\ast, \ast), (\ast, \square), (\square, \ast), (\square, \square) \} \]
\[ R_{\lambda U^-} = \{ (s_1, s_1), (s_2, s_2), (s_2, s_1), (s_3, s_2) \} \]
An Alternative Question. Does the BGK conjecture hold for tiered PTSs? Furthermore, can we classify those tiered PTSs that are SN?
Theorem. (Girard, Coquand) $\lambda U^-$ 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)* $\lambda U^-$ 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 $\lambda U$ and Coquand improves this to $\lambda U^-$ be encoding Reynold’s Theorem.
Girard’s Paradox

**Theorem.** (Girard, Coquand) $\lambda U^-$ 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 $\lambda U$ and Coquand improves this to $\lambda U^-$ be encoding Reynold’s Theorem.

**Question.** $\lambda U^-$ is tiered. Are there any other non-normalizing tiered PTSs?
Theorem. (Girard, Coquand) $\lambda U^-$ 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 $\lambda U$ and Coquand improves this to $\lambda U^-$ be encoding Reynold’s Theorem.

Question. $\lambda U^-$ 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
Lou proved that ECC is SN (a reasonable subsystem of ECC can be viewed as a PTS).
Lou proved that ECC is SN (a reasonable subsystem of ECC can be viewed as a PTS).

Mellèis and Werner generalized the $\Lambda$-set method of Altenkirch for a generic strong normalization proof for PTSs.
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.
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.
To prove $\lambda S$ 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.
To prove $\lambda S$ 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)* $\lambda P$ is SN if $\lambda \rightarrow$ is SN.

**Theorem.** *(Geuvers, Nederhof)* $\lambda C$ is SN if $\lambda \omega$ is SN.
To prove $\lambda S$ 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) $\lambda P$ is SN if $\lambda \rightarrow$ is SN.

**Theorem.** (Geuvers, Nederhof) $\lambda C$ 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 \( \bot \), 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 $\lambda C$. Define three translations $\rho_1$, $\rho_0$ and $\gamma$ 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 $\lambda C$. Define three translations $\rho_1$, $\rho_0$ and $\gamma$ such that

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

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

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.
Consider translating a predicate on $A$.\textsuperscript{1} We translate $*$ to 0 and translate functions and applications inductively on structure.

\begin{align*}
\text{A Dependent Case} \\
\Gamma \vdash P : A \rightarrow \ast & \quad \Gamma \vdash M : A \\
\Gamma \vdash PM : \ast
\end{align*}

\textsuperscript{1}These example are taken from Geuvers and Nederhof.
Consider translating a predicate on $A$. We translate $\star$ to 0 and translate functions and applications inductively on structure.

**A Dependent Case**

\[
\Gamma \vdash P : A \rightarrow \star \quad \Gamma \vdash M : A \\
\Gamma \vdash PM : \star
\]

**After Translation**

\[
\rho_0(\Gamma) \vdash \gamma(P) : \rho_0(A) \rightarrow 0 \quad \rho_0(\Gamma) \vdash \gamma(M) : \rho_0(A) \\
\rho_0(\Gamma) \vdash \gamma(P)\gamma(M) : 0
\]

---

¹These example are taken from Geuvers and Nederhof.
In the non-dependent case, polymorphism makes things tricky.

**A Non-Dependent Case**

\[
\Gamma \vdash P : \Pi A^* A \to A \quad \Gamma \vdash B : \star \\
\Gamma \vdash PB : B \to B
\]
In the non-dependent case, polymorphism makes things tricky.

*A Non-Dependent Case*

\[
\Gamma \vdash P : \Pi A^* A \rightarrow A \quad \Gamma \vdash B : \star \\
\Gamma \vdash PB : B \rightarrow B
\]

*After Translation*

\[
\rho_0(\Gamma) \vdash \gamma(P) : \text{???} \quad \rho_0(\Gamma) \vdash \gamma(B) : 0 \\
\rho_0(\Gamma) \vdash \gamma(P)\gamma(B) : \rho_0(B) \rightarrow \rho_0(B)
\]
In the non-dependent case, polymorphism makes things tricky.

\[ \Gamma \vdash P : \Pi A^* A \rightarrow A \quad \Gamma \vdash B : \star \]
\[ \Gamma \vdash PB : B \rightarrow B \]

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

\[ \rho_0(\Gamma) \vdash \gamma(P) : \Pi A^* \Pi x^0 A \rightarrow A \quad \rho_0(\Gamma) \vdash \gamma(B) : 0 \]

\[ \rho_0(\Gamma) \vdash \gamma(P) \rho_0(B) \gamma(B) : \rho_0(B) \rightarrow \rho_0(B) \]
Updated Translation

\[
\begin{align*}
\rho_0(\Gamma) &\vdash \gamma(P) : \Pi A^* \Pi x^0 A \rightarrow A &\rho_0(\Gamma) &\vdash \gamma(B) : 0 \\
\frac{}{\rho_0(\Gamma) \vdash \gamma(P) \rho_0(B) \gamma(B) : \rho_0(B) \rightarrow \rho_0(B)}
\end{align*}
\]

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.
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 \(\lambda S\) is 
\((j, i)\)-full.
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 \(\lambda S\) is \((j, i)\)-full.

Fullness is a very strong restriction. Most full systems contain a \(\lambda U^-\) (so are non-normalizing) and the small class of systems that do not contain \(\lambda U^-\) are subsystems of ECC. The strongest non-trivial system is the following.
Challenge 3: Dealing with \( \bot \) (1/2)

For any tiered PTS \( \lambda S \), define its non-dependent restriction, denoted \( \lambda S^* \), to be the tiered system with rules
\[
\{(s_i, s_j, s_j) \in R_{\lambda S} \mid i \leq j\}.
\]
Challenge 3: Dealing with ⊥ (1/2)

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

In the last stage of the translation for $\lambda C$ 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 $\Pi$-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 $\lambda S^*$. 
The idea. All higher sorts are very scarcely inhabited, e.g.,

So we define one more intermediate translation that eliminates the uses of \( \bot \)'s in these cases.
Theorem. For any full tiered PTS $\lambda S$, there are two functions $\tau : T \rightarrow T$ and $\llbracket - \rrbracket : T \rightarrow T$ on terms such that the following hold.

1. If $\Gamma \vdash A : B$ then there is a context $\Gamma'$ such that $\Gamma' \vdash_{\lambda S^*} \llbracket A \rrbracket : \tau(B)$. That is, $\llbracket - \rrbracket$ preserves typability.

2. For any term $A$ typable in $\lambda S$, if $A \rightarrow^\beta B$, then $\llbracket A \rrbracket \rightarrow^\beta_+ \llbracket B \rrbracket$. In particular, $\llbracket - \rrbracket$ preserves infinite reduction paths.

Corollary. Let $\lambda S$ be a full PTS. Then $\lambda S$ is SN if $\lambda S^*$ is SN.

Corollary. For any full PTS, WN is equivalent to SN is provable in Peano Arithmetic.
Theorem. For any full tiered PTS $\lambda S$, there are two functions $\tau : T \rightarrow T$ and $\llbracket - \rrbracket : T \rightarrow T$ on terms such that the following hold.

1. If $\Gamma \vdash A : B$ then there is a context $\Gamma'$ such that $\Gamma' \vdash_{\lambda S^*} \llbracket A \rrbracket : \tau(B)$. That is, $\llbracket - \rrbracket$ preserves typability.

2. For any term $A$ typable in $\lambda S$, if $A \rightarrow^\beta B$, then $\llbracket A \rrbracket \rightarrow^+_{\beta} \llbracket B \rrbracket$. In particular, $\llbracket - \rrbracket$ preserves infinite reduction paths.

Corollary. Let $\lambda S$ be a full PTS. Then $\lambda S$ is SN if $\lambda S^*$ is SN.
Results

Theorem. For any full tiered PTS $\lambda S$, there are two functions $\tau : T \to T$ and $\llbracket - \rrbracket : T \to T$ on terms such that the following hold.

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

Corollary. Let $\lambda S$ be a full PTS. Then $\lambda S$ is SN if $\lambda S^*$ is SN.

Corollary. For any full PTS, WN is equivalent to SN is provable in Peano Arithmetic.
A Simple Bootstrapping Argument (Xi)

1. If $\lambda S$ is WN then so is $\lambda S^*$. 

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.
A Simple Bootstrapping Argument (Xi)

1. If $\lambda S$ is WN then so is $\lambda S^*$.
2. Since $\lambda S^*$ is non-dependent, if the CPS translation of Barthe et al. can be applied to it, then $\lambda S^*$ is SN.
1. If $\lambda S$ is WN then so is $\lambda S^*$. 
2. Since $\lambda S^*$ is non-dependent, if the CPS translation of Barthe et al. can be applied to it, then $\lambda S^*$ is SN. 
3. Using the dependency eliminating translation, it follows that $\lambda S$ is SN.
A Simple Bootstrapping Argument (Xi)

1. If $\lambda S$ is WN then so is $\lambda S^*$.

2. Since $\lambda S^*$ is non-dependent, if the CPS translation of Barthe et al. can be applied to it, then $\lambda S^*$ is SN.

3. Using the dependency eliminating translation, it follows that $\lambda S$ 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)

<table>
<thead>
<tr>
<th>Sorts</th>
<th>( \rho_i(s_j) \triangleq \begin{cases} 0 &amp; i = 0 \ s_i &amp; \text{otherwise} \end{cases} ) (where ( j \geq i ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Var.</td>
<td>( \rho_i(s_j x) \triangleq s_{i+2} x ) (where ( j \geq i + 2 ))</td>
</tr>
<tr>
<td>( \Pi )-terms</td>
<td>( \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) &amp; \deg A \geq i + 1 \ \rho_i(B) &amp; \text{otherwise} \end{cases} )</td>
</tr>
<tr>
<td>( \lambda )-terms</td>
<td>( \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) &amp; \deg A \geq i + 2 \ \rho_i(M) &amp; \text{otherwise} \end{cases} )</td>
</tr>
<tr>
<td>App.</td>
<td>( \rho_i(MN) \triangleq \begin{cases} \rho_i(M) \rho_{\deg N - 1}(N) \ldots \rho_i(N) &amp; \deg N \geq i + 1 \ \rho_i(M) &amp; \text{otherwise} \end{cases} )</td>
</tr>
</tbody>
</table>
### The Translation (2/2)

<table>
<thead>
<tr>
<th>Sorts</th>
<th>$\gamma(s_i) \triangleq c^0$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\gamma(s_i x) = s_1 x$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Variables</th>
<th>$\Pi$-terms</th>
<th>$\lambda$-terms</th>
<th>App.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\gamma \Pi x^A B \triangleq$</td>
<td>$\gamma(\lambda x^A M) \triangleq$</td>
<td>$\gamma(MN) \triangleq$</td>
</tr>
<tr>
<td></td>
<td>$c^0 \rightarrow 0 \rightarrow 0 \gamma(A)(\gamma(B)[c_\text{deg } A-1(A) / s_\text{deg } A x] \ldots [c_0(A) / s_1 x])$</td>
<td>$(\lambda z^0 \lambda x^{\text{deg } A-1(A)} \ldots \lambda x^{\text{deg } A}(\gamma(M)) \gamma(A)$</td>
<td>$\begin{cases} \gamma(M) \rho_{\text{deg } N-1}(N) \ldots \rho_0(N) \gamma(N) &amp; \text{deg } N \geq 1 \ \gamma(M) \gamma(N) &amp; \text{deg } N = 0 \end{cases}$</td>
</tr>
</tbody>
</table>

(I have not included the last stage which eliminates the uses of $\bot$-terms)
Conclusions
What is the normalization behavior of tiered PTSs? Even the case of $n = 3$ is not known, I would guess $\lambda S$ is SN if and only if it does not contain $\lambda U^-$. 
Open Questions

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

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

- Are there any more examples of non-normalizing tiered PTSs besides $\lambda U^-$?

- What other normalization techniques are amenable to the PTS setting?
  - perpetual reductions
  - $\beta$-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?
Open Questions

- What is the normalization behavior of tiered PTSs? Even the case of $n = 3$ is not known, I would guess $\lambda S$ is SN if and only if it does not contain $\lambda U^-$. 
- Are there any more examples of non-normalizing tiered PTSs besides $\lambda U^-$?
- What other normalization techniques are amenable to the PTS setting?
  - perpetual reductions
  - $\beta$-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