Weak and Strong Beta Normalisations in Typed λ-Calculi

Hongwei Xi

Department of Mathematical Sciences
Carnegie Mellon University
Pittsburgh, PA 15213, USA

Abstract. We present a technique to study relations between weak and strong β-normalisations in various typed λ-calculi. We first introduce a translation which translates a λ-term into a λI-term, and show that a λ-term is strongly β-normalisable if and only if its translation is weakly β-normalisable. We then prove that the translation preserves typability of λ-terms in various typed λ-calculi. This enables us to establish the equivalence between weak and strong β-normalisations in these typed λ-calculi. This translation can deal with Curry typing as well as Church typing, strengthening some recent closely related results. This may bring some insights into answering whether weak and strong β-normalisations in all pure type systems are equivalent.

1 Introduction

In various typed λ-calculi, one of the most interesting and important properties on λ-terms is *how* they can be β-reduced to β-normal forms. A λ-term M is said to be *weakly β-normalisable* ($\mathcal{WN}_\beta(M)$) if it can be β-reduced to a β-normal form in *some* way while M is *strongly β-normalisable* ($\mathcal{SN}_\beta(M)$) if *every* β-reduction sequence starting from M terminates with a β-normal form. If every λ-term in a λ-calculus λS is weakly/strongly β-normalisable, we say that λS enjoys *weak/strong β-normalisation* ($\lambda S \models \mathcal{WN}_\beta / \lambda S \models \mathcal{SN}_\beta$).

A conjecture presented by Barendregt at *The Second International Conference on Typed Lambda Calculi and Applications* states that $\lambda S \models \mathcal{WN}_\beta$ implies $\lambda S \models \mathcal{SN}_\beta$ for every pure type system λS [2]. One can certainly strengthen the conjecture by asking whether it is provable in the first-order Peano arithmetic. We will argue that this is the case for various typed λ-calculi including all systems in λ-cube [2].

We achieve this via introducing a translation $||\cdot||$ which translates a λ-term M into a λI-term $||M||$. We then show that $\mathcal{WN}_\beta(||M||)$ implies $\mathcal{SN}_\beta(M)$ for every λ-term M and $||\cdot||$ preserves the typability of λ-terms in various typed λ-calculi.

An immediate consequence of our technique is that it can also be used to infer $\lambda S \models \mathcal{SN}_\beta$ from $\lambda S \models \mathcal{WN}_\beta$ for many typed λ-calculi λS. In a retrospection, $\lambda S \models \mathcal{SN}_\beta$ were usually proven later than $\lambda S \models \mathcal{WN}_\beta$ for many λS such as $\lambda^{\rightarrow}$, $\lambda\cap^{-}$, $\lambda 2$ and λC, involving more complicated methods. In addition, our technique can help estimate bounds on $\mu(M)$ for certain typable λ-terms M, where $\mu(M)$ is the number of β-reduction steps in a longest β-reduction sequence from M. We will demonstrate this for the simply typed λ-terms.

The next section presents some preliminaries and basic known results. We define translations $||\cdot||$ and $|\cdot|$ recursively in Section 3, and prove some properties on them. We present several applications of $||\cdot||$ in Section 4 involving both Curry typing and Church typing, and point out some available methods for removing type dependencies on terms. Finally, we mention some related work and draw some conclusions on our technique.

2 Preliminaries

We give a brief explanation on the notions and terminology used in this paper. Most details can be found in [1], [19] and [2].

Definition 1. The set Λ of λ-terms is defined inductively as follows.

- (variable) There are infinitely many variables $x, y, z, \ldots$ in Λ; variables are subterms of themselves.
- (abstraction) If $M \in \Lambda$ then $(\lambda x.M) \in \Lambda$; N is a subterm of $(\lambda x.M)$ if N is $(\lambda x.M)$ or a subterm of M.
- (application) If $M_1, M_2 \in \Lambda$ then $M_1(M_2) \in \Lambda$; N is a subterm of $M_1(M_2)$ if N is $M_1(M_2)$ or a subterm of M_i for some $i \in \{1, 2\}$.

Let Var be the set of all variables. The set $\mathbf{FV}(M)$ of free variables in a λ-term M is defined as follows.

$$\mathbf{FV}(M) = \begin{cases} \{M\} & \text{if } M \in \text{Var}; \\ \mathbf{FV}(M_1)\backslash\{x\} & \text{if } M = (\lambda x.M_1); \\ \mathbf{FV}(M_1) \cup \mathbf{FV}(M_2) & \text{if } M = M_1(M_2). \end{cases}$$

An occurrence of a variable x in a term M is bound if it occurs in some term M_1 where $\lambda x.M_1$ is a subterm of M; an occurrence of a variable is free if it is not bound.

The set Λ_I of λI-terms is the maximal subset of Λ such that, for every term $M \in \Lambda_I$, if $(\lambda x.N)$ is a subterm of M then $x \in \mathbf{FV}(N)$.

$[N/x]M$ stands for substituting N for all free occurrences of x in M. α-conversion or renaming bound variables may have to be performed in order to avoid name collisions. Also certain substitution properties, such as the substitution lemma (Lemma 2.1.16 [Bar84]), are assumed.

Definition 2. A term of form $(\lambda x.M)(N)$ is called a β-redex, and $[N/x]M$ is called the contractum of the β-redex; $M_1 \leadsto_\beta M_2$ stands for a β-reduction step in which M_2 is obtained from replacing some β-redex in M_1 with its contractum; a β-reduction sequence is a possibly infinite sequence of form

$$M_1 \leadsto_\beta M_2 \leadsto_\beta \ldots;$$

a λ-term is in β-normal form if it contains no β-redexes; $(\lambda x.M)(N)$ is a β_I-redex if $x \in \mathbf{FV}(M)$.

We use R for β-redexes. $\leadsto^n_\beta$, $\leadsto^+_\beta$ and $\leadsto^*_\beta$ stand for β-reduction sequences consisting of n steps, a positive number of steps and a nonnegative number of steps, respectively.

Definition 3. A λ-term M is weakly β-normalisable ($\mathcal{WN}_\beta(M)$) if there exists a β-reduction sequence $M \leadsto^*_\beta N$ such that N is in β-normal form; M is strongly β-normalisable ($\mathcal{SN}_\beta(M)$) if all β-reduction sequences from M are finite.

We write $\lambda S \models \mathcal{WN}_\beta$ ($\lambda S \models \mathcal{SN}_\beta$) if all λ-terms in λS are weakly (strongly) normalisable.

Let $\mu(M) = \infty$ if there exists an infinite β-reduction sequence from M, and let $\mu(M)$ be the number of β-reduction steps in a longest β-reduction sequence from M if $\mathcal{SN}_\beta(M)$, which is guaranteed to exist by König's Lemma.

We adopt

$$n + \infty = \infty + n = \infty + \infty = \infty$$

in the following arithmetic involving ∞.

Lemma 4. *Given $M_R = R(M_1)\dots(M_n)$ and $M_N = N(M_1)\dots(M_n)$, where $R = (\lambda x.N_1)(N_2)$ and $N = [N_2/x]N_1$; then $\mu(M_R) \le 1 + \mu(M_N) + \mu(N_2)$.*

Proof. It suffices to prove this when $\mu(M_N) < \infty$ and $\mu(N_2) < \infty$. Please see the proof of Lemma 4.3.3 (1) in [2].

Now let us state the conservation theorem for λI-calculus, upon which this paper is established.

Theorem Church *For every λI-term M, $\mathcal{WN}_\beta(M)$ if and only if $\mathcal{SN}_\beta(M)$.*

Proof. See the proof of Theorem 11.3.4 in [1] or the proof of Theorem 21 in [33].

3 Translations

Translations $|\cdot|$ and $||\cdot||$ translate λ-terms into λI-terms.

$$||x|| = x(\bullet) \qquad ||M_1 M_2|| = ||M_1||(|M_2|)$$
$$||(\lambda x.M)|| = (\lambda x.|M|(\langle x, \bullet\rangle)) \qquad |M| = (\lambda \bullet .||M||)$$

Note that $\bullet$ and $\langle , \rangle$ are two distinct fresh variables and $\langle x, \bullet\rangle$ stands for $\langle , \rangle(x)(\bullet)$.

Proposition 5. *$||\cdot||$ and $|\cdot|$ have the following properties.*

1. *Given any λ-term M; $||M||$ and $|M|$ are λI-terms and there exists only one free occurrence of $\bullet$ in $||M||$.*
2. *If M_s is a subterm of M then $||M_s||$ is a subterm of $||M||$.*
3. *$[|N|/x]||M|| \leadsto^*_\beta ||[N/x]M||$ for any M and N.*

Proof. (1) and (2) can be readily proven by a structural induction on λ-terms. Since $|N|(\bullet) \rightsquigarrow_\beta \|N\|$ for every λ-term N, (3) can also be proven by a structural induction on M.

Lemma 6. *(Main Lemma) For every λ-term M, $\mu(M) \leq \mu(\|M\|)$.*

Proof. It suffices to show this when $\mu(\|M\|) < \infty$. We proceed by an induction on $\mu(\|M\|)$ and the structure of M, lexicographically ordered.

- M is of form $(\lambda x.N)$. Then $\mu(N) \leq \mu(\|N\|)$ by induction hypothesis. Hence,

$$\mu(M) = \mu(N) \leq \mu(\|N\|) < \mu(|N|(\langle x, \bullet\rangle)) = \mu(\|M\|).$$

- M is of form $x(M_1)\dots(M_n)$ for some $n \geq 0$. Then

$$\|M\| = \|x\|(|M_1|)\dots(|M_n|).$$

  By induction hypothesis, $\mu(M_i) \leq \mu(\|M_i\|) = \mu(|M_i|)$ for $i = 1, \dots, n$. This yields

$$\begin{aligned}\mu(M) &= \mu(M_1) + \dots + \mu(M_n) \leq \mu(\|M_1\|) + \dots + \mu(\|M_n\|) \\ &= \mu(|M_1|) + \dots + \mu(|M_n|) = \mu(\|M\|).\end{aligned}$$

- M is of form $R(M_1)\dots(M_n)$ for some β-redex $R = (\lambda x N_1)(N_2)$. By definition, $\|M\| = \|R\|(|M_1|)\dots(|M_n|)$. Let

$$N = [N_2/x]N_1 \quad \text{and} \quad M_N = N(M_1)\dots(M_n).$$

  With Proposition 5,

$$\begin{aligned}\|R\| &= \|(\lambda x.N_1)\|(|N_2|) = (\lambda x.|N_1|(\langle x, \bullet\rangle))(|N_2|) \\ &\rightsquigarrow_\beta ([|N_2|/x]|N_1|)(\langle |N_2|, \bullet\rangle) \rightsquigarrow_\beta^* |[N_2/x]N_1|(\langle |N_2|, \bullet\rangle) \\ &= |N|(\langle |N_2|, \bullet\rangle) \rightsquigarrow_\beta [\langle |N_2|, \bullet\rangle/\bullet]\|N\|.\end{aligned}$$

  We can take $\mu(\|N_2\|)$ β-reduction steps to reduce $\|N_2\|$ to some λ-term $\star$ in β-normal form. Since there is one free occurrence of $\bullet$ in $\|N\|$, we can take at least $2 + \mu(\|N_2\|)$ β-reduction steps to reduce $\|R\|$ to $\|N\|^+ = [\langle \star, \bullet\rangle]\|N\|$. Let $\|M_N\|^+ = \|N\|^+(|M_1|)\dots(|M_n|)$, then

$$2 + \mu(\|N_2\|) + \mu(\|M_N\|^+) \leq \mu(\|M\|).$$

  Notice that $\mu(\|M_N\|) \leq \mu(\|M_N\|^+)$ since we can simply treat $\langle \star, \bullet\rangle$ as if it were $\bullet$. By induction hypothesis, $\mu(N_2) \leq \mu(\|N_2\|)$ and $\mu(M_N) \leq \mu(\|M_N\|)$ since $\mu(\|N_2\|) \leq \mu(\|M_N\|) < \mu(\|M\|)$. With Lemma 4,

$$\begin{aligned}\mu(M) &\leq 1 + \mu(N_2) + \mu(M_N) < 2 + \mu(\|N_2\|) + \mu(\|M_N\|) \\ &\leq 2 + \mu(\|N_2\|) + \mu(\|M_N\|^+) \leq \mu(\|M\|).\end{aligned}$$

Therefore, $\mu(M) \leq \mu(\|M\|)$ for every λ-term M.

Corollary 7. *For every λ-term M, $\mathcal{WN}_\beta(||M||)$ implies $\mathcal{SN}_\beta(M)$.*

Proof. Note that $||M||$ is a λI-term. By Theorem Church, $\mathcal{WN}_\beta(||M||)$ implies $\mathcal{SN}_\beta(||M||)$, namely, $\mu(||M||) < \infty$. Therefore, $\mu(M) \leq \mu(||M||) < \infty$ by Lemma 6, and this yields $\mathcal{SN}_\beta(M)$.

Lastly, we mention a translation $(\!|\cdot|\!)$ which is a minor variant of $|\cdot|$. $(\!|\cdot|\!)$ can also be used to establish a version of Lemma 6.

$$
\begin{aligned}
&(\!|x|\!) = x\\
&(\!|(\lambda x.M)|\!) = (\lambda \bullet.\lambda x.(\!|M|\!)(\langle x, \bullet\rangle))\\
&(\!|M(N)|\!) = (\lambda \bullet.(\!|M|\!)(\bullet)((\!|N|\!)))
\end{aligned}
$$

It becomes clear that the use of $\bullet$ is to collect λ-terms which might be thrown away in β-reductions.

4 Applications

With the help of Corollary 7, we are now ready to prove the equivalence between $\lambda S \models \mathcal{WN}_\beta$ and $\lambda S \models \mathcal{SN}_\beta$ in various typed λ-calculi. This amounts to typing $||M||$ in these systems when given a typed term M. Since the method that we use cannot deal with all *pure type systems* uniformly, we choose some appropriate λ-calculi to illustrate our technique. We also mention that translation $||\cdot||$ enables us to easily assign a bound for the length of β-reduction sequences from simply typed λ-terms.

4.1 λ-Curry and λ-Church

We assume the familiarity of the reader with $\lambda^\to$-Curry ($\lambda^\to$ with Curry typing) and $\lambda^\to$-Church ($\lambda^\to$ with Church typing). Translations $||\cdot||$ and $|\cdot|$ on simple types are given as follows.

$$||b|| = b \qquad ||A \to B|| = |A| \to ||B|| \qquad |A| = [] \to ||A||$$

Note that b ranges over base types and $[]$ is some base type. Transforms $||\cdot||$ and $|\cdot|$ on terms in $\lambda^\to$-Church are defined below.

$$
\begin{aligned}
&||x|| = x(\bullet) && ||M_1(M_2)|| = ||M_1||(|M_2|)\\
&||(\lambda x : A.M)|| = \lambda x : |A|.|M|(\langle x, \bullet\rangle) && |M| = \lambda\bullet : [].||M||
\end{aligned}
$$

Lemma 8. *If M is a term in $\lambda^\to$-Curry ($\lambda^\to$-Church) with type A, then $||M||$ is a term in $\lambda^\to$-Curry ($\lambda^\to$-Church) with type $||A||$.*

Proof. Note that $\langle,\rangle$'s are given types of form $A \to ([] \to [])$, and they are different variables if their types are different. A structural induction on M yields the results.

By a method invented by Turing [8] and, independently, by Prawitz [25], it can be readily proven that every term M in $\lambda^\to$-Curry($\lambda^\to$-Church) is $\mathcal{WN}_\beta$, and therefore $\mathcal{SN}_\beta(M)$ by Corollary 7 since Lemma 8 yields $\mathcal{WN}_\beta(||M||)$. This is clearly an arithmetisable proof. We pursue this further in the next section.

4.2 An Upper Bound for β-Reduction Sequences in $\lambda^{\rightarrow}$-Church

$\lambda^{\rightarrow}$ means $\lambda^{\rightarrow}$-Church in this section. The size of a $\lambda^{\rightarrow}$-term is given below.

$$\mathbf{size}[x] = 1 \quad \mathbf{size}[(\lambda x : A.M)] = 1 + \mathbf{size}[M] \quad \mathbf{size}[M(N)] = \mathbf{size}[M] + \mathbf{size}[N]$$

The rank $\rho(A)$ of a simple type A is defined as follows.

$$\rho(A) = \begin{cases} 0 & \text{if } A \text{ is a base type;} \\ 1 + \max\{\rho(A_0), \rho(A_1)\} & \text{if } A = A_0 \to A_1. \end{cases}$$

The rank $\rho(R)$ of a β-redex $R = (\lambda x : A.M)(N)$ of type B is $\rho(A \to B)$, and the rank $\hat{\rho}(M)$ of a $\lambda^{\rightarrow}$-term M is

$$\hat{\rho}(M) = \max\{\rho(R) : R \text{ is a } \beta\text{-redex in } M\}.$$

We define functions $2_0(n) = n$ and $2_{k+1}(n) = 2^{2_k(n)}$ for $k = 0, 1, 2, \ldots$.

Theorem 9. *$\mu(M) < 2_{k+1}(n)$ for every $\lambda^{\rightarrow}I$-term M, where $k = \hat{\rho}(M)$ and $n = \mathbf{size}[M]$.*

Proof. Please see [33] for a proof using the standardisation theorem in the untyped λ-calculus. A similar result can also be found in [28].

Corollary 10. *$\mu(M) < 2_{2k+1}(5n)$ for every $\lambda^{\rightarrow}$-term M, where $k = \hat{\rho}(M)$ and $n = \mathbf{size}[M]$.*

Proof. It can be readily verified that $\rho(|A|) \leq 2\rho(A)$ for every simple type A, yielding $\hat{\rho}(\|M\|) \leq 2\hat{\rho}(M)$. Also notice $\mathbf{size}[\|M\|] \leq 5 \cdot \mathbf{size}[M]$ for every $\lambda^{\rightarrow}$-term M. Since $\|M\|$ is a $\lambda^{\rightarrow}I$-term, $\mu(\|M\|) < 2_{2k+1}(5n)$ by Theorem 9. Therefore, $\mu(M) \leq \mu(\|M\|) < 2_{2k+1}(5n)$ by Lemma 6.

A result of a similar form is mentioned in [9], and is proven in [28].

The following translation $[\![\cdot]\!]$ translates $\lambda^{\rightarrow}$-terms into $\lambda^{\rightarrow}I$-terms of the same types.

$$\begin{aligned} [\![x]\!] &= x; \\ [\![M(N)]\!] &= [\![M]\!]([\![N]\!]) \\ [\![(\lambda x : A.M)]\!] &= (\lambda x : A.\lambda x_1 : A_1. \ldots \lambda x_n : A_n.\langle [\![M]\!](x_1) \ldots (x_n), x\rangle), \\ &\quad \text{where } M \text{ is of type } A_1 \to (\cdots (A_n \to B) \cdots) \\ &\quad \text{and } B \text{ is a base type;} \end{aligned}$$

Note that $\langle , \rangle$ is a fresh variable of type $B \to (A \to B)$ in the definition. It is interesting to know that $\mu(M) \leq \mu([\![M]\!])$ for every $\lambda^{\rightarrow}$-term M. This is intimately related to the translation used in [28]. $[\![\cdot]\!]$ is far less flexible than $\|\cdot\|$ since it can only translate $\lambda^{\rightarrow}$-terms. For the reader who is interested in this subject, we point out that our technique can also be applied to the typed λ-calculus with let-polymorphism.

4.3 $\lambda 2$ with Curry Typing

The second-order polymorphic typed λ-calculus $\lambda 2$ is originally introduced in [10] and [27], where Church typing is involved. $\lambda 2$ with Curry typing can be formulated as follows.

Syntax

$$\begin{array}{lll} \text{Types} & A & ::= X \mid A_1 \rightarrow A_2 \mid \forall X.A \\ \text{Contexts} & \Gamma & ::= \cdot \mid \Gamma, x : A \\ \text{Terms} & M & ::= x \mid (\lambda x.M) \mid M_1(M_2) \end{array}$$

We use X, Y for type variables, A, B for types, Γ for contexts, M, N for terms and x for variables assuming that any variable can be declared at most once in a context. We write $[B/X]A$ for the type obtained by substituting B for all free occurrences of X in A.

Typing Rules

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A}(var)$$

$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash (\lambda x.M) : A \rightarrow B}(\rightarrow I) \qquad \frac{\Gamma \vdash M : A \rightarrow B \qquad \Gamma \vdash N : A}{\Gamma \vdash M(N) : B}(\rightarrow E)$$

$$\frac{\Gamma \vdash M : A}{\Gamma \vdash M : \forall X.A}(\forall I)^* \qquad \frac{\Gamma \vdash M : \forall X.A}{\Gamma \vdash M : [B/X]A}(\forall E)$$

Γ must contain no free occurrences of X in rule $\forall I$. $\mathcal{D} :: \Gamma \vdash M : A$ denotes a typing derivation with conclusion $\Gamma \vdash M : A$.

$\lambda 2 \models \mathcal{WN}_\beta$ is equivalent to $\lambda 2 \models \mathcal{SN}_\beta$

Definition 11. M is a $\lambda 2$-term if $\Gamma \vdash M : A$ is derivable for some Γ and A.

Let $\Lambda 2$ be the set of all $\lambda 2$-terms, and we show that $\Lambda 2$ is closed under $||\cdot||$ and $|\cdot|$. Translations $||\cdot||$ and $|\cdot|$ on types are given below, where $\bot = \forall X.X$.

$$\begin{array}{ll} ||X|| = X & ||A \rightarrow B|| = |A| \rightarrow ||B|| \\ ||\forall X.A|| = \forall X.||A|| & |A| = \bot \rightarrow ||A|| \end{array}$$

Given $\Gamma = x_1 : A_1, \ldots, x_n : A_n$, then $|\Gamma| = \langle,\rangle : (\forall X.X \rightarrow (\bot \rightarrow \bot)), x_1 : |A_1|, \ldots, x_n : |A_n|$, and $||\Gamma|| = |\Gamma|, \bullet : \bot$. It can be readily verified that $|\Gamma, x : A|, \bullet : \bot \vdash \langle x, \bullet \rangle : \bot$ is derivable for any Γ and A.

Lemma 12. *Given $M \in \Lambda 2$, then $||M|| \in \Lambda 2$.*

Proof. By definition, there exists a derivation $\mathcal{D}$ with conclusion $\Gamma \vdash M : A$ for some Γ and A. We proceed by induction on the structure of $\mathcal{D}$ to show that there exists $\|\mathcal{D}\| :: \|\Gamma\| \vdash \|M\| : \|A\|$. Since $\|\Gamma\| = |\Gamma|, \bullet : \bot$, we can define $|\mathcal{D}| :: |\Gamma| \vdash |M| : |A|$ as follows when $\|\mathcal{D}\|$ is constructed.

$$\frac{\|\mathcal{D}\| :: |\Gamma|, \bullet : \bot \vdash \|M\| : \|A\|}{|\Gamma| \vdash |M| = (\lambda \bullet . \|M\|) : |A| = \bot \to \|A\|}(\to I).$$

We do a case analysis according to the last typing rule applied in $\mathcal{D}$.

- If $\mathcal{D}$ ends with (var), $(\to E)$ or $(\forall I)$, the case is trivial.
- $\mathcal{D}$ is of form

$$\frac{\mathcal{D}_1 :: \Gamma_1 \vdash M_1 : A_2}{\mathcal{D} :: \Gamma \vdash (\lambda x.M_1) : A_1 \to A_2}(\to I),$$

  where $\Gamma_1 = \Gamma, x : A_1$ and $M = (\lambda x.M_1)$ and $A = A_1 \to A_2$. By induction hypothesis, we have

$$\frac{\dfrac{|\mathcal{D}_1| :: |\Gamma_1|, \bullet : \bot \vdash |M_1| : |A_2| = \bot \to \|A_2\| \qquad |\Gamma_1|, \bullet : \bot \vdash \langle x, \bullet \rangle : \bot}{|\Gamma_1|, \bullet : \bot \vdash |M_1|(\langle x, \bullet \rangle) : \|A_2\|}(\to E)}{|\Gamma|, \bullet : \bot \vdash (\lambda x.|M_1|(\langle x, \bullet \rangle)) : \|A\| = |A_1| \to \|A_2\|}(\to I).$$

  Since $\|\Gamma\| = |\Gamma|, \bullet : \bot$ and $\|M\| = (\lambda x.|M_1|(\langle x, \bullet \rangle))$, we have derived $\|\Gamma\| \vdash \|M\| : \|A\|$.
- $\mathcal{D}$ is of form

$$\frac{\mathcal{D}_1 :: \Gamma \vdash M : \forall X.A_1}{\mathcal{D} :: \Gamma \vdash M : [B/X]A_1}(\forall E),$$

  where $A = [B/X]A_1$. By induction hypothesis, we have

$$\frac{\|\mathcal{D}_1\| :: \|\Gamma\| \vdash \|M\| : \|\forall X.A_1\| = \forall X.\|A_1\|}{\|\Gamma\| \vdash \|M\| : [\|B\|/X]\|A_1\|}(\forall E).$$

  Notice $[\|B\|/X]\|A_1\| = \|[B/X]A_1\| = \|A\|$. Hence, we have derived $\|\Gamma\| \vdash \|M\| : \|A\|$.

Therefore, $\|M\|$ is a $\lambda 2$-term.

Theorem 13. *$\lambda 2 \models \mathcal{WN}_\beta$ if and only if $\lambda 2 \models \mathcal{SN}_\beta$.*

Proof. Given any term $M \in \Lambda 2$, we have $\|M\| \in \Lambda 2$ by Lemma 12. If $\lambda 2 \models \mathcal{WN}_\beta$ then $\mathcal{WN}_\beta(\|M\|)$. This implies that $\mathcal{SN}_\beta(M)$ by Corollary 7. Therefore, if $\lambda 2 \models \mathcal{WN}_\beta$ then $\lambda 2 \models \mathcal{SN}_\beta$. The other direction is trivial.

Note that this is a recent result, which can be formulated in the first-order Peano arithmetic. On the other hand, no proofs of $\lambda 2 \models \mathcal{SN}_\beta$ can be established in the second-order Peano arithmetic. After Girard had proven that $\lambda 2 \models \mathcal{WN}_\beta$, several people (including Girard himself) modified his definition of reducibility candidates and proved that $\lambda 2 \models \mathcal{SN}_\beta$, but none of these proofs achieve this by showing the equivalence between $\lambda 2 \models \mathcal{WN}_\beta$ and $\lambda 2 \models \mathcal{SN}_\beta$.

4.4 $\lambda\omega$ with Church Typing

We choose $\lambda\omega$, the higher order polymorphic typed λ-calculus, to show that our technique can also work with Church typing. Here we can see how to handle types which contain β-redexes. Types depending on terms are avoided intentionally in this case for some reason which will be explained later.

Syntax

$$
\begin{array}{lll}
\text{Kinds} & \kappa & ::= * \mid \kappa_1 \to \kappa_2 \\
\text{Constructors} & \tau & := t \mid (\lambda t : \kappa.\tau) \mid \tau_1(\tau_2) \mid \tau_1 \to \tau_2 \mid \forall t : \kappa.\tau \\
\text{Kind Contexts} & \Delta & ::= \cdot \mid \Delta, t : \kappa \\
\text{Type Contexts} & \Gamma & ::= \cdot \mid \Gamma, x : \tau \\
\text{Terms} & e & ::= x \mid (\lambda x : \tau.e) \mid (\lambda t : \kappa.e) \mid e_1(e_2) \mid e(\tau)
\end{array}
$$

We use k for kinds, τ for constructors, t for variables over constructors, Δ for kind contexts, Γ for type contexts, e for terms and x for variables. Variable can be declared at most once in a context. τ is a type if $\Delta \vdash \tau : *$ is derivable for some Δ.

Typing Rules

$$\frac{t : \kappa \in \Delta}{\Delta \vdash t : \kappa}$$

$$\frac{\Delta, t : \kappa_1 \vdash \tau : \kappa_2}{\Delta \vdash (\lambda t : \kappa_1.\tau) : \kappa_1 \to \kappa_2} \qquad \frac{\Delta \vdash \tau_1 : \kappa_1 \to \kappa_2 \qquad \Delta \vdash \tau_2 : \kappa_1}{\Delta \vdash \tau_1(\tau_2) : \kappa_2}$$

$$\frac{\Delta, t : \kappa \vdash \tau : *}{\Delta \vdash (\forall t : k.\tau) : *} \qquad \frac{\Delta \vdash \tau_1 : * \qquad \Delta \vdash \tau_2 : *}{\Delta \vdash \tau_1 \to \tau_2 : *}$$

We write $\Delta \vdash \Gamma$ if $\Delta \vdash \tau : *$ for every $x : \tau \in \Gamma$.

$$\frac{\Delta \vdash \Gamma \qquad x : \tau \in \Gamma}{\Gamma \vdash_\Delta x : \tau}(var) \qquad \frac{\Gamma \vdash_\Delta e : \tau_1 \qquad \tau_1 =_\beta \tau_2 \qquad \Delta \vdash \tau_2 : *}{\Gamma \vdash_\Delta e : \tau_2}(=_\beta)$$

$$\frac{\Gamma, x : \tau_1 \vdash_\Delta e : \tau_2}{\Gamma \vdash_\Delta (\lambda x : \tau_1.e) : \tau_1 \to \tau_2}(\to I) \qquad \frac{\Gamma \vdash_\Delta e_1 : \tau_1 \to \tau_2 \qquad \Gamma \vdash_\Delta e_2 : \tau_1}{\Gamma \vdash_\Delta e_1(e_2) : \tau_2}(\to E)$$

$$\frac{\Gamma \vdash_{\Delta, t:k} e : \tau}{\Gamma \vdash_\Delta (\lambda t : k.e) : (\forall t : k.\tau)}(\forall I)^* \qquad \frac{\Gamma \vdash_\Delta e : (\forall t : k.\tau_1) \qquad \Delta \vdash \tau_2 : k}{\Gamma \vdash_\Delta e(\tau_2) : [\tau_2/t]\tau_1}(\forall E)$$

Now we can define transforms $||\cdot||$ and $|\cdot|$ on constructors.

$$
\begin{array}{ll}
||t|| = t & ||\lambda t : k.\tau|| = \lambda t : k.||\tau|| \\
||\tau_1(\tau_2)|| = ||\tau_1||(||\tau_2||) & ||\forall t : k.\tau|| = \forall t : k.||\tau|| \\
||\tau_1 \to \tau_2|| = |\tau_1| \to ||\tau_2|| & |\tau| = \bot \to ||\tau||
\end{array}
$$

Notice that $|\cdot|$ is only defined on types.

Proposition 14. *We have the followings.*

1. $\|[\tau_2/t]\tau_1\| = [\|\tau_2\|/t]\|\tau_1\|$.
2. $\tau_1 =_\beta \tau_2$ *implies* $\|\tau_1\| =_\beta \|\tau_2\|$.

Proof. (1) follows from a structural induction on τ_1, and (2) follows from (1).

Now we define $\|\cdot\|$ and $|\cdot|$ on $\lambda\omega$-terms and type contexts.

$$\begin{array}{ll} \|x\| = x(\bullet) & \|e_1(e_2)\| = \|e_1\|(|e_2|) \\ \|(\lambda x:\tau.e)\| = (\lambda x:|\tau|.|e|(\langle x,\bullet\rangle_{|\tau|})) & \|(\lambda t:\kappa.e)\| = (\lambda t:\kappa.\|e\|) \\ \|e(\tau)\| = \|e\|(\|\tau\|) & |e| = (\lambda\bullet:\bot.\|e\|) \end{array}$$

$$|\cdot| = \langle,\rangle : \forall t:*.\, t\to(\bot\to\bot) \qquad |\Gamma, x:\tau| = |\Gamma|, x:|\tau| \qquad \|\Gamma\| = |\Gamma|, \bullet:\bot$$

Note that $\langle x,\bullet\rangle_{|\tau|}$ stands for $\langle,\rangle(|\tau|)(x)(\bullet)$.

Lemma 15. *If $\Gamma \vdash e:\tau$ is derivable, then both*

$$\|\Gamma\| \vdash \|e\| : \|\tau\| \quad \textit{and} \quad |\Gamma| \vdash |e| : |\tau|$$

are also derivable.

Proof. This follows from a structural induction on the derivation of $\Gamma \vdash e:\tau$.

Theorem 16. $\lambda\omega \models \mathcal{WN}_\beta$ *if and only if* $\lambda\omega \models \mathcal{SN}_\beta$.

Proof. Following the proof of Lemma 6, we can establish $\mu(e) \le \mu(\|e\|)$ accordingly. It is easy to observe that $\mathcal{SN}_\beta(\tau)$ for every constructor τ since τ corresponds to a λ-term in some simply typed λ-calculus with constants. Given a $\lambda\omega$-term e and $\|e\| \leadsto_\beta^* e_*$; it is easy to note that every β-redex r in e_* is either a β_I-redex or of form $(\lambda t:k.e_0)\tau$. Hence, r can always be regarded as a perpetual β-redex [3] since one can only substitute constructors for the variables in τ. Therefore, $\mathcal{WN}_\beta(\|e\|)$ implies $\mathcal{SN}_\beta(\|e\|)$. The rest is straightforward.

Through $\lambda 2$ with Curry typing and $\lambda\omega$ with Church typing we can see that $\|\cdot\|$ handles Curry typing and Church typing with virtually no difference. This is quite desirable. We will return to this point when we mention another translation $[\cdot]$ in Section 5.

4.5 Types Depending on Terms and λ-cube

There exists a serious problem in handling dependent types if we apply our technique directly. Given two distinct λ-terms a and b such that $a =_\beta b$; let e be a term of type $P(a)$, then e is also of type $P(b)$ since $P(a) =_\beta P(b)$; if we assign type $P(\|a\|)$ to $\|e\|$, then type $P(\|b\|)$ can also be assigned to $\|e\|$; when $\|a\| \neq_\beta \|b\|$, this may not be possible since $P(\|a\|)$ and $P(\|b\|)$ are two different types; $\|a\| \neq_\beta \|b\|$ often holds when $a \neq b$(try $a = (\lambda x.x)$ and $b = a(a)$).

Fortunately, there exist some methods in [12] and [11] to remove type dependencies on terms for certain typed λ-calculi. A proof in [11] shows that $\lambda\omega \models \mathcal{SN}_\beta$ implies $\lambda C \models \mathcal{SN}_\beta$, where λC stands for the construction of calculus. Following this example, we can verify that $\lambda S \models \mathcal{WN}_\beta$ if and only if $\lambda S \models \mathcal{SN}_\beta$ for every system λS in λ-cube [2]. Again we point out that this is a result which can be formulated in the first-order Peano arithmetic.

5 Related Work

The research on deriving strong normalisation (SN) from weak normalisation (WN) has lasted for at least thirty years. Nederpelt[21], Klop[17], Karr[16], de Groot[7], and Kfoury and Wells[20] have all invented techniques to infer SN from WN. Their techniques all require introducing some notions of reduction different from β-reduction, deriving strong β-normalisation from weak normalisation of these newly introduced notions of reduction. For example, Klop's technique amounts to introducing a pairing constant $[\cdot,\cdot]$, a π-reduction $\leadsto_\pi$ and a κ-reduction $\leadsto_\kappa$ as follows [29].

$$[M_1, M_2]N \leadsto_\pi [M_1N, M_2] \qquad\qquad [M_1, M_2] \leadsto_\kappa M_1$$

These techniques are successful when applied to λ-calculi λS for which there exist syntactic proofs of $\lambda S \models \mathcal{WN}_\beta$. If one tries to prove $\lambda 2 \models \mathcal{SN}_\beta$, it is doubtful that one can gain much (if there is any) by arguing that $\lambda 2$ with some newly introduced reductions enjoys weak normalisation. In addition, these techniques does not help much on establishing the equivalence between $\mathcal{WN}_\beta$ and $\mathcal{SN}_\beta$ in various typed λ-calculi.

Gandy[9] interprets simply typed λ-terms as strictly increasing functionals. His method, now called *functional interpretations*, can yield an upper bound for the lengths of β-reduction sequences from simply typed λ-terms. In this direction further work can be found in [23] and [24] and [15]. One can somewhat view our technique as a syntactic realisation of Gandy's idea.

Schwichtenberg[28] shows an upper bound for the lengths of β-reduction sequences from simply typed λ-terms. In his proof, he uses a translation closely related to translation $\cdot$, which translates simply typed λ-terms into simply typed λI-terms. This translation can only work with $\lambda^{\rightarrow}$-terms. Translating λP-terms into $\lambda^{\rightarrow}$-terms [12], Springintveld[30] then uses Schwichtenberg's upper bound to establish an upper bound for the lengths of β-reduction sequences from λP-terms.

Sørensen and the author independently discovered the following translation $[\cdot]$, which translates a λ-term M into a λI-term $[M]$ such that $\mathcal{WN}_\beta([M])$ implies $\mathcal{SN}_\beta(M)$.

$$\begin{aligned}
&[x] = x\\
&[(\lambda x.M)] = (\lambda k_1.k_1(\lambda x.\lambda k_2.\langle [M](k_2), x\rangle))\\
&[M_1(M_2)] = (\lambda k_1.[M_0](\lambda k_2.k_2([M_1])(k_1)))
\end{aligned}$$

Note that $\langle , \rangle$ is a fresh variable and a term of form $\langle M, N \rangle$ stands for $\langle , \rangle(M)(N)$. $[\cdot]$ is a minor variant of Plotkin's call-by-name continuation passing style translation. Sørenson[29] then proved the equivalence between $\mathcal{WN}_\beta$ and $\mathcal{SN}_\beta$ in various typed λ-calculi including the simply typed λ-calculus, the simply typed λ-calculus with positive recursive types, and $\lambda\omega$. The author[32] showed the equivalence between $\mathcal{WN}_\beta$ and $\mathcal{SN}_\beta$ in $\lambda^{\rightarrow}$ and $\lambda 2$ with Church-typing, and mentioned that $[\cdot]$ can also be applied to $\lambda\omega$.

$[\cdot]$ cannot be applied to terms in $\lambda 2$ with Curry typing since $[M]$ may not be a well-typed $\lambda 2$-term for some $M \in \Lambda 2$. Some explanation can be found in [13] and [14]. Also $[\cdot]$ has trouble working with $\lambda\cap^-$-terms. Reasoning with $[\cdot]$ is usually more difficult than with $||\cdot||$ since one has to deal with a lot of administrative β-redexes introduced by CPS-transformation [22].

6 Conclusion and Future Work

We have demonstrated some applications of our technique. The reader is encouraged to verify that this technique also works on $\lambda\cap^-$ with Curry typing [19], $\lambda\cap^-$ with Church typing [20] and the λ-calculi with positive recursive types [31]. On the other hand, translation $[\cdot]$, which exploits continuations, has troubles handling Curry typing [29], and is less robust than translation $||\cdot||$. Besides, $||\cdot||$ — in the author's opinion — leads to much more straightforward reasoning than $[\cdot]$ does.

All the presented proofs of equivalence between $\mathcal{WN}_\beta$ and $\mathcal{SN}_\beta$ in various typed λ-calculi can be formulated in the first-order Peano arithmetic. Therefore, the following conjecture seems to have some grounds.

Conjecture 17. *For every pure type system λS, the equivalence between $\lambda S \models \mathcal{WN}_\beta$ and $\lambda S \models \mathcal{SN}_\beta$ can be established in the first-order Peano arithmetic.*

Like $[\cdot]$, $||\cdot||$ has a serious problem handling types depending on terms since it does not preserve the β-equivalence between terms(it would be useless for our purpose if it did). We are exploring the possibility to generalise $||\cdot||$ to a family of translations, making the technique applicable to dependent types. Another direction is to adapt the techinique into systems such as Gödel's T, where constants and δ-reduction rules are allowed.

7 Acknowledgement

I thank Frank Pfenning, Peter Andrews and Richard Statman for their support and for providing me a nice work environment. I also thank some anonymous referees for their constructive comments on this paper.

References

1. H.P. Barendregt (1984), The Lambda Calculus: Its Syntax and Semantics, *North-Holland publishing company, Amsterdam.*

2. H.P. Barendregt (1992), Lambda calculi with types, *Handbook of Logic in Computer Science edited by S. Abramsky, Dov M. Gabbay and T.S.E. Maibaum*, Clarendon Press, Oxford, pp. 117-414.
3. J.A. Bergstra and J.W. Klop (1982), Strong normalization and perpetual reductions in the lambda calculus, *J. Inform. Process. Cybernet. 18 (718)*, pp. 403-417.
4. A. Church, (1941), The calculi of lambda conversion, *Princeton University Press, Princeton.*
5. M.Coppo and M.Dezani-Ciancaglini (1980), An Extension of Basic Functionality Theory for the Lambda-calculus, *Notre Dame Journal of Formal Logic*, 21(4), pp. 685-693.
6. M.Coppo, M.Dezani-Ciancaglini and B.Venneri (1981), Functional Characters of Solvable Terms, *Zeitschrift fur Mathematische Logik und Grundlagen der Mathematik*, 27(1), pp. 45-58.
7. P. de Groote (1993), The conservation theorem revisited, *Int'l conf. Typed lambda calculi and applications*, vol. 664 of LNCS, pp. 163-178.
8. R.O. Gandy (1980), An early proof of normalisation by A.M. Turing, *To: H.B. Curry: Essays on combinatory logic, lambda calculus and formalism*, Academic press, pp. 453-456.
9. R.O. Gandy (1980), Proofs of Strong Normalisation, *To: H.B. Curry: Essays on Combinatory logic, lambda calculus and formalism*, Academic press, pp. 457-478.
10. J.-Y. Girard (1972), Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur, *Thèse de doctorat d'etat, Université Paris VII.*
11. H. Geuvers and M.J. Nederhof (1991), A modular proof of strong normalisation for the calculus of constructions, *Journal of Functional Programming.*
12. R. Harper, F. Honsell and G. Plotkin (1987), A framework for defining logics. In *Proc. Second Symposium of Logic in Computer Science*, pages 194-204, Ithaca, N.Y.
13. R. Harper and M. Lillibridge (1993), Explicit polymorphism and CPS conversion, In Proceedings of the Twentieth ACM Symposium on Principles of Programming Languages, pp. 206-209
14. R. Harper and M. Lillibridge (1993), Polymorphic type assignment and CPS conversion, *LISP and Symbolic Computation*, vol. 6, 3(4), pp. 361-380.
15. Stefan Kahrs (1995), Towards a Domain Theory for Termination Proofs, *Laboratory for Foundation of Computer Science*, 95-314, Department of Computer Science, The University of Edinburgh.
16. M. Karr (1985), "Delayability" in proofs of strong normalisabilities in the typed lambda-calculus. In H. Ehrig, C. Floyd, M. Nïvat and J. Thatcher, editors, *Mathematical Foundation of Computer Software*, vol. 185 of LNCS, pp. 208-222.
17. J.W. Klop (1980), Combinatory Reduction Systems, *Ph.D. thesis, CWI, Amsterdam, Mathematical center tracts, No. 127.*
18. J.W. Klop, (1992), Term Rewriting Systems, *Handbook of Logic in Computer Science edited by S. Abramsky, Dov M. Gabbay and T.S.E. Maibaum*, Clarendon Press, Oxford, pp. 1-116.
19. J.L. Krivine (1990), *Lambda-calcul, Types et Modèles*, Masson, Paris.
20. A.J. Kfoury and J.B. Wells (1994), New notions of reduction and non-semantic proofs of β-strong normalisation in typed λ-calculi, *Tech. Rep.*

94-104, Computer Science Department, Boston University.
21. R.P. Nederpelt (1973), Strong normalization in a typed lambda calculus with lambda structured types, *Ph.D. thesis, Technische Hogeschool Eindhoven.*
22. G. Plotkin (1975), Call-by-name, call-by-value, and the lambda calculus, *Theoretical Computer Science*, vol 1, pp. 125-159.
23. J. van de Pol (1994), Termination proofs for higher-order rewrite systems. In J. Heering, K. Meinke, B. Möller and T. Nipkow, editors, *Higher Order Algebra, Logic and Term Rewriting*, vol. 816 of LNCS, pp. 305-325.
24. J. van de Pol and H. Schwichtenberg (1995), Strict functionals for termination proofs, *Int'l conf. Typed lambda calculi and applications*, vol. 902 of LNCS, pp. 350-364.
25. D. Prawitz (1965), Natural Deduction: A proof theoretical study, Almquist & Wiksell publishing company.
26. D. Prawitz (1971), Ideas and results of proof theory, Proceedings of the 2nd scandinavian logic symposium, *editor J.E. Fenstad, North-Holland Publishing Company, Amsterdam.*
27. J. Reynolds (1974), Towards a theory of type structure, *Colloquium sur la Progrmmation*, vol. 19 of LNCS, pp. 408-423.
28. H. Schwichtenberg (1991), An upper bound for reduction sequences in the typed lambda-calculus, *Archive for Mathematical Logic*, 30:405-408.
29. M.H. Sørensen (1996), Strong Normalization from Weak Normalization by A-Translation in Typed lambda-Calculi, *Manuscript announced on the types mailing list*, February.
30. J. Springintveld (1993), Lower and upper bounds for reductions of types in $\lambda\underline{\omega}$ and λP. *Int'l conf. Typed lambda calculi and applications*, vol. 664 of LNCS, pp. 391-405.
31. P. Urzyczyn (1995), Positive recursive type assignment, *Mathematical Foundations of Computer Science*, vol. 969 of LNCS, pp. 382-391.
32. H. Xi (1996), On weak and strong normalisations, *Research Report 96-187, Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh.*
33. H. Xi (1996), Upper bounds for standardisations and an application, *Research Report 96-189, Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh.*
34. H. Xi (1996), An induction measure on λ-terms and its applications, *Research Report 96-192, Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh.*

This article was processed using the LaTeX macro package with LLNCS style