Simulating η-Expansions with β-Reductions in the Second-Order Polymorphic λ-Calculus

Hongwei Xi

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

email: hwxi+@cs.cmu.edu

Abstract. We introduce an approach to simulating η-expansions with β-reductions in the second-order polymorphic λ-calculus. This generalizes the work of Di Cosmo and Delia Kesner which simulates η-expansions with β-reductions in simply typed settings, positively solving the conjecture on whether the simulation technique can be extended to polymorphic settings. We then present a *modular* proof that the second-order polymorphic λ-calculus with an expansive version of η-reduction is strong normalizing and confluent. The simulation is also promising to provide modular proofs showing that other rewriting systems are also strongly normalizing after expanded with certain versions of η-expansion.

1 Introduction and Related Work

η-conversion presents an approach to studying extensional equalities for λ-terms. Given an η-equality $\lambda x.M(x) =_\eta M$, where x has no free occurrences in M; one can either say $\lambda x.M(x)$ η-contracts ($\rightarrow_\eta$) to M, or M η-expands ($\rightarrow_{\eta_*}$) to $\lambda x.M(x)$; the former is usually adopted as a rewrite rule since every λ-term can then be η-contracted to an η-normal form. This strategy leads to a confluent untyped λ-calculus $\lambda\beta\eta$, in which every λ-term M has a β-normal form if and only if M has a $\beta\eta$-normal. Carrying η-contraction into the simply typed λ-calculus $\lambda^{\rightarrow}\beta\eta$, one can prove that $\lambda^{\rightarrow}\beta\eta$ is confluent and strongly normalizing. However, problems occur when $\lambda^{\rightarrow}\beta\eta$ is augmented with a unit type $\mathbf{T}$ representing the terminal object of a cartesian closed category, and an extensional rule

$$M \rightarrow_{\mathbf{T}} *,$$

where M is of type $\mathbf{T}$ and $*$ is the only element in $\mathbf{T}$. The confluence breaks down as shown in the following well-known example [16]:

$$\lambda x.f(x) \rightarrow_\eta f \quad \text{and} \quad \lambda x.f(x) \rightarrow_{\mathbf{T}} \lambda x.*,$$

where f is of type $\mathbf{T} \rightarrow \mathbf{T}$ and x of type $\mathbf{T}$. Note that both f and $\lambda x.*$ cannot be contracted further. This is a serious drawback since it can easily occur when one adds η-contraction to algebraic rewriting systems. One immediate remedy is to allow $N \rightarrow_{\mathbf{T} \rightarrow \mathbf{T}} \lambda x.*$ for every term N of type $\mathbf{T} \rightarrow \mathbf{T}$. However, this method can produce an unwieldy system since infinitely many rules have to be included in order to handle terms of other similar types.

The use of η-expansion as a rewrite rule was suggested in [18]. Mints [17] presented a proof for the confluence and (weak) normalization of a simply typed λ-calculus with surjective pairing and η-expansion. A flaw in Mints's proof was later corrected in [6]. Y. Akama [1] proved the confluence and strong normalization of the above system, which is also given by C.B. Jay and N. Ghani [15]. Di Cosmo and Kesner discovered a translation $(\cdot)^{\circ}$ which simulates η-expansion with β-reduction in a simply typed setting [8]. An application of this translation can also be found in [7].

The following example shows that applying η-expansion unconditionally easily leads to infinite reduction sequence.

$$x^{A \rightarrow B} \rightarrow_{\eta_*} \lambda y^A.x^{A \rightarrow B}(y^A) \rightarrow_{\eta_*} \lambda y^A.(\lambda y^A.x^{A \rightarrow B}(y^A))(y^A) \rightarrow_{\eta_*} \cdots$$

However, this can be remedied by applying $\rightarrow_{\eta_*}$ with restrictions; namely, λ-abstraction cannot be η-expanded, nor can terms which are applied to other terms. Systems with such restricted η-expansions have been receiving attentions of a growing number of researchers as shown in the reference.

The translation discovered by Di Cosmo and Kesner provides a modular approach to proving that many rewriting systems are still strong normalizing and confluent after augmented with restricted versions of η-expansion [7]. This is a very useful proof technique in the first-order settings. Unfortunately, their translation cannot be applied to polymorphic settings directly. Although they have proven that the second-order polymorphic λ-calculus with surjective pairing and a version of η-expansion is strongly normalizing and confluent, their proof is not modular. Compared with the corresponding results in simply typed settings, this is less satisfactory.

We will generalize the translation of Di Cosmo and Kesner to the second-order polymorphic λ-calculus. We then present a modular proof, showing the second-order polymorphic λ-calculus with a version of η-expansion is strongly normalizing and confluent. This new translation is also promising to be a powerful technique dealing with η-expansions in other polymorphic settings [5].

Extensional polymorphism is studied in [5, 9, 10, 13]. A modular proof for the strong normalisation and confluence of $\lambda\beta\beta_2\eta_*$ (in our notation) is given in [10], which is rather long and complicated. Since $\lambda\beta\beta_2\eta_*$ is $\lambda\beta\beta_2\eta_*\eta_*^2$ excluding $\rightarrow_{\eta_*^2}$, the result simply follows from our main result (Corollary 9). We point out that proofs for our main result have been already presented in [9] and [13], but those proofs are not modular. They are established upon the well-known *reducibility candidates* method due to Tait [21] and Girard [14].

It is interesting to mention that the author learned the use η-expansion while studying the equivalence between strong and weak normalizations in various typed λ-calculi. Schwichtenberg [20] used η-expansion to translate simply typed λ-terms into simply typed λI-terms so that β-reduction sequences from the former can be simulated by those from the latter. This can yield a bound for the lengths of β-reduction sequence from simply typed terms. Details can also be founded in [22].

2 Preliminaries

The second-order polymorphic typed λ-calculus $\lambda 2$ is originally introduced in [14] and [19], where Church typing is involved. We now give a slightly different formulation of $\lambda 2$.

$$\begin{array}{l} \text{Types } A, B ::= a \mid b \mid X \mid A \to B \mid \forall X.A \\ \text{Terms } M, N ::= x^A \mid (\lambda x^A.M) \mid M(N) \mid (\Lambda X.M) \mid M(A) \end{array}$$

We use a, b for base types, X for type variables, A, B for types, M, N for terms and x^A, y^A for variables of type A. We write $[B/X]A$ for the type obtained by substituting B for every free occurrence of X in A. We often write $\lambda x^A.M$ for $(\lambda x^A.M)$ and $\Lambda X.M$ for $(\Lambda X.M)$, enhancing clarity.

As usual, let $\mathrm{FV}(M)$ be the set of free (term) variables in term M and $\mathrm{FTV}(A)$ be the set of free type variables in type A. We also need the following notions: $\mathrm{FTV}(M)$ is the set of free type variables in term M and $\mathrm{FTFV}(M)$ is the set of free type variables in the types of free variables in M.

$$\begin{array}{ll} \mathrm{FTV}(x^A) = \mathrm{FV}(A) & \mathrm{FTV}(M(N)) = \mathrm{FTV}(M) \cup \mathrm{FTV}(N) \\ \mathrm{FTV}(\lambda x^A.M) = \mathrm{FTV}(A) \cup \mathrm{FTV}(M) & \mathrm{FTV}(\Lambda X.M) = \mathrm{FTV}(M) \backslash \{X\} \\ \mathrm{FTV}(M(A)) = \mathrm{FTV}(A) \cup \mathrm{FTV}(M) & \mathrm{FTFV}(M) = \bigcup \{\mathrm{FTV}(A) : x^A \in \mathrm{FV}(M)\} \end{array}$$

We assume the familiarity of the reader with free and bound occurrences of variables. Term equalities are modulo α-conversions, which are often taken implicitly. The followings are typing rules for $\lambda 2$.

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

$$\frac{\vdash M : B}{\vdash (\lambda x^A.M) : A \to B}(\to I) \qquad \frac{\vdash M : A \to B \qquad \vdash N : A}{\vdash M(N) : B}(\to E)$$

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

Note that the rule $\forall I$ can be applied only if $X \notin \mathrm{FTFV}(M)$. A term M is a $\lambda 2$-term of type A if $\vdash M : A$ is derivable.

An advantage of this formulation is that we can define a function τ which maps terms to their types without bothering their typing derivations.

$$\begin{array}{l} \tau(x^A) = A \qquad\qquad \tau(\lambda x^A.M) = A \to \tau(M) \\ \tau(M(N)) = B, \text{ if } \tau(M) = A \to B \text{ and } \tau(N) = A; \\ \tau(\Lambda X.M) = \forall X.\tau(M)\ \tau(M(B)) = [B/X]A, \text{ if } \tau(M) = \forall X.A \end{array}$$

Since η-expansions needs the guidance of types, this formulation of $\lambda 2$ brings a great deal of convenience in our following development. If we use typing rules with contexts in the formulation of $\lambda 2$, we have to carry contexts around in order to compute the types of open terms, complicating our following presentation *significantly.*

Proposition 1. *For every term M, M is a $\lambda 2$-term of type A if and only if $\tau(M) = A$.*

Proof. A structural induction on M yields the result. $\square$

Given a $\lambda 2$-term M of type A; for every λ-term of type B, $[N/x^B]M$ is the term obtained by substituting N for every free occurrence of x^B in M; for every type B, $[B/X]M$ is the term obtained by substituting B for every free occurrence of X in M; it can be readily verified that $\tau([N/x^B]M) = A$ and $\tau([B/X]M) = [B/X]A$. We now present the reduction rules for the second order extensional λ-calculus $\lambda\beta\beta_2\eta_*\eta_*^2$.

- (β-contraction) $(\lambda x^A.M)(N) \overset{\beta}{\to} [N/x^A]M$
- (β^2-contraction) $(\Lambda X.M)(A) \overset{\beta^2}{\to} [A/X]M$
- (η-expansion) $M \overset{\eta_*}{\to} \lambda x^A.M(x^A)$, if $x^A \notin \mathrm{FV}(M)$ and $\tau(M) = A \to B$ for some B and M is not a λ-abstraction
- (η^2-expansion) $M \overset{\eta_*^2}{\to} \Lambda X.M(X)$, if $X \notin \mathrm{FTV}(M)$ and $\tau(M) = \forall X.B$ for some B and M is not a Λ-abstraction

β-reduction ($\to_\beta$) is given below, and β^2-reduction ($\to_{\beta^2}$) can be given accordingly.

$$\frac{M_1 \overset{\beta}{\to} M_2}{M_1 \to_\beta M_2} \qquad \frac{M_1 \to_\beta M_2}{\lambda x^A.M_1 \to_\beta \lambda x^A.M_2} \qquad \frac{M_1 \to_\beta M_2}{M_1(N) \to_\beta M_2(N)}$$

$$\frac{M_1 \to_\beta M_2}{N(M_1) \to_\beta N(M_2)} \qquad \frac{M_1 \to_\beta M_2}{\Lambda X.M_1 \to_\beta \Lambda X.M_2} \qquad \frac{M_1 \to_\beta M_2}{M_1(A) \to_\beta M_2(A)}$$

η-reduction ($\to_{\eta_*}$) and η^2-reduction ($\to_{\eta_*^2}$) can also be defined accordingly with the following restrictions, respectively.

$$\frac{M_1 \to_{\eta_*} M_2 (\text{except } M_1 \overset{\eta_*}{\to} M_2)}{M_1(N) \to_{\eta_*} M_2(N)} \qquad \frac{M_1 \to_{\eta_*^2} M_2 (\text{except } M_1 \overset{\eta_*^2}{\to} M_2)}{M_1(A) \to_{\eta_*^2} M_2(A)}$$

Now we define reduction $\underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow}$ and $\underset{\beta\beta^2}{\longrightarrow}$ as follows.

$$\underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow} = \to_\beta \cup \to_{\beta^2} \cup \to_{\eta_*} \cup \to_{\eta_*^2} \quad \text{and} \quad \underset{\beta\beta^2}{\longrightarrow} = \to_\beta \cup \to_{\beta^2} .$$

For any decorated reduction notation of $\to$ in this paper, the corresponding decorated reduction notations of $\twoheadrightarrow$ and $\twoheadrightarrow^+$ stand for some (possibly zero) and a positive number of steps of such a reduction, respectively. We shall use *subject reduction* property of $\lambda 2$ implicitly in our following presentation. We assume the familiarity of the reader with this property.

3 Translation and Simulation

For each type variable X, we assign to it a *fresh* term variable $v^{X\to X}$ of type $X \to X$, which will *only* be used for the following translation purpose.

$$
\begin{array}{ll}
|b| = b & |X| = X \\
|A \to B| = |A| \to |B| & |\forall X.A| = \forall X.(X \to X) \to |A| \\
\Delta_b = \lambda x^b.x^b & \Delta_X = v^{X\to X} \\
\multicolumn{2}{l}{\Delta_{A\to B} = \lambda x^{|A\to B|}.\lambda y^{|A|}.\Delta_B(x^{|A\to B|}(\Delta_A(y^{|A|})))} \\
\multicolumn{2}{l}{\Delta_{\forall X.A} = \lambda x^{|\forall X.A|}.\Lambda X.\lambda v^{X\to X}.\Delta_A(x^{|\forall X.A|}(X)(v^{X\to X}))}
\end{array}
$$

Notice $\Delta_{a\to b}(x^{a\to b}) \twoheadrightarrow_\beta (\lambda y^a.x^{a\to b}(y^a))$, which is the $\to_{\eta_*}$-normal form of $x^{a\to b}$. In general, for M of type A, $\Delta_A(M)$ $\beta\beta^2$-reduces a $\to_{\eta_*\eta_*^2}$-normal form which closely relates to the $\to_{\eta_*\eta_*^2}$-normal form of M as shown below. These Δ's are called expansors, which set up the machinery to simulate η-expansions with β-reductions.

Proposition 2. *Given types A and B, the following hold.*

1. $[|B|/X]|A| = |[B/X]A|$.
2. $\tau(\Delta_A) = |A| \to |A|$.
3. $[\Delta_B/v^{|B|\to|B|}][|B|/X]\Delta_A = \Delta_{[B/X]A}$.

Proof. An structural induction on A yields the results. □

We use $\Delta^0_A(M)$ for M and $\Delta^{n+1}_A(M)$ for $\Delta_A(\Delta^n_A(M))$ when $n \geq 0$.

Definition 3. We now define versions of $|M|$ and $|M|^+$ inductively.

$$
\begin{array}{ll}
|x^A| = \Delta^n_A(x^{|A|}) & |x^A|^+ = \Delta_A(|x^A|) \\
|(\lambda x^A.M)| = \Delta^n_{A\to\tau(M)}(\lambda x^{|A|}.|M|^+) & |(\lambda x^A.M)|^+ = |(\lambda x^A.M)| \\
|M(N)| = \Delta^n_{\tau(M(N))}(|M|(|N|^+)) & |M(N)|^+ = \Delta_{\tau(M(N))}(|M(N)|) \\
|(\Lambda X.M)| = \Delta^n_{\forall X.\tau(M)}(\Lambda X.\lambda v^{X\to X}.|M|^+) & |(\Lambda X.M)|^+ = |(\Lambda X.M)| \\
|M(B)| = \Delta^n_{\tau(M(B))}(|M|(|B|)(\Delta_B)) & |M(B)|^+ = \Delta_{\tau(M(B))}(|M(B)|)
\end{array}
$$

Note that n can be any nonnegative integers here. Hence, for every λ2-term, $|M|$ and $|M|^+$ have infinitely many different versions.

We shall try to simulate a $\underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow}$-reduction sequence from M with some $\underset{\beta\beta^2}{\longrightarrow}$-reduction sequence from $|M|^+$. The need of versions can be explained as follows. Given $M \to_\beta N$, we need to have $|M|^+ \twoheadrightarrow_\beta |N|^+$ in order to make the simulation work. Suppose we translate $(\lambda x^A.x^A)(M)$ to $(\lambda x^A.\Delta_A(x^A))(\Delta_A(M))$, which β-reduces to $\Delta^2_A(M)$. Note that $\Delta^2_A(M)$ can not be β-reduced to $\Delta_A(M)$, though they may be β-equivalent. This forces us to adopt that $\Delta^2_A(M)$ is also a version of $|M|^+$ since the simulation would break down otherwise.

Examples Given $M = x^{a\to b}(y^a)$, where a, b are base types; $\Delta_b^2(x^{a\to b}(\Delta_a(y^a)))$ is a version of $|M|$ and a version of $|M|^+$; $\Delta_{a\to b}(x^{a\to b})(\Delta_a^2(y^a))$ is a version of $|M|$, but not a version of $|M|^+$. Let $N = \Lambda X.(\lambda x^X.x^X)$ and $A = a \to b$; note $N(A)$ is in $\eta_*\eta_*^2$-normal form; we have $N(A) \to_{\beta^2} N_1 = (\lambda x^A.x^A)$ but N_1 is not in η_*-normal form; we translate $N(A)$ to

$$(\Lambda X.\lambda v^{X\to X}.(\lambda x^X.v^{X\to X}(x^X)))(A)(\Delta_A);$$

this translation $\beta\beta^2$-reduces to $(\lambda x^A.\Delta_A(x^A))$, which is a version of $|N_1|^+$; then η-expansion from N_1 can be simulated. The use of $v^{X\to X}$ is like an *expansor candidate* which will be instantiated with some expansor when X is instantiated.

We often use $|M|$ ($|M|^+$) for a *version* of $|M|$ ($|M|^+$) in our following writing. This convention brings a great deal of convenience and turns out to be adequately clear, esp. with the help of contexts. For instance, $|M| \underset{\beta\beta^2}{\twoheadrightarrow} |N|$ means that *every* version of $|M|$ $\beta\beta^2$-reduces to *some* version of $|N|$.

If the reader is interested in the proof details for the following results, please see [23].

Proposition 4. *Given $\lambda 2$-term M of type A and N of type B, we have the following.*

1. *$|M|$ and $|M|^+$ are $\lambda 2$-terms of type $|A|$.*
2. *$[|N|/x^{|B|}]|M|$ is a version of $|[N/x^B]M|$, and $[|N|/x^{|B|}]|M|^+$ is a version of $|[N/x^B]M|^+$.*
3. *$[\Delta_B/v^{|B|\to|B|}][|B|/X]|M|$ is a version of $|[B/X]M|$, and $[\Delta_B/v^{|B|\to|B|}][|B|/X]|M|^+$ is a version of $|[B/X]M|^+$.*

Proof. By a structural induction on M. □

Proposition 5. *We have the following.*

1. *For every $\lambda 2$-term $\lambda x^A.M$ of type $A \to B$, $|\lambda x^A.M| \twoheadrightarrow_\beta \lambda x^{|A|}.|M|^+$.*
2. *For every $\lambda 2$-term $\Lambda X.M$ of type $\forall X.A$, $|\Lambda X.M| \underset{\beta\beta^2}{\twoheadrightarrow} \Lambda X.\lambda v^{X\to X}.|M|^+$.*

Proof. We may assume

$$|\lambda x^A.M| = \Delta_{A\to B}^n(\lambda x^A.|M|^+) \quad\text{and}\quad |\Lambda X.M| = \Delta_{\forall X.A}^n(\Lambda X.\lambda v^{X\to X}.|M|^+)$$

for some n. The proof proceeds by induction on n.

- $n = 0$. (1) and (2) hold by definition.
- $n = k + 1$ for some $k \geq 0$. Then by induction hypothesis, we have

$$\begin{aligned}
\Delta_{A\to B}^{k+1}(|\lambda x^A.M|) &= \Delta_{A\to B}(\Delta_{A\to B}^k(|\lambda x^A.M|)) \\
&\twoheadrightarrow_\beta \Delta_{A\to B}(\lambda x^{|A|}.|M|^+), \text{ by induction hypothesis} \\
&\to_\beta \lambda y^{|A|}.\Delta_B((\lambda x^{|A|}.|M|^+)(\Delta_A(y^{|A|}))) \\
&\to_\beta \lambda y^{|A|}.\Delta_B([\Delta_A(y^{|A|})/x^{|A|}]|M|^+) \\
&= \lambda y^{|A|}.|[y^A/x^A]M|^+, \text{ by Proposition 4 (2)} \\
&= \lambda x^{|A|}.|M|^+, \alpha\text{-conversion}
\end{aligned}$$

Also, we have

$$\begin{array}{rcl} \Delta^{k+1}_{\forall X.A}(|\Lambda X.M|) & = & \Delta_{\forall X.A}(\Delta^{k}_{\forall X.A}(|\Lambda X.M|)) \\ & \underset{\beta\beta^2}{\twoheadrightarrow} & \Delta_{\forall X.A}(\Lambda X.\lambda v^{X\to X}.|M|^+), \text{ by induction hypothesis} \\ & \to_\beta & \Lambda X.\lambda v^{X\to X}.\Delta_A((\Lambda X.\lambda v^{X\to X}.|M|^+)(X)(v^{X\to X})) \\ & \underset{\beta\beta^2}{\twoheadrightarrow} & \Lambda X.\lambda v^{X\to X}.|M|^+. \end{array}$$

Hence, we are done.

□

The next lemma will be used to establish our main simulation result.

Lemma 6. *(Main Lemma) We have the followings.*

1. *Given* $R = (\lambda x^A.M)(N)$ *and* $M^* = [N/x]M$, $|R| \twoheadrightarrow^+_\beta |M^*|$.
2. *Given* $R = (\Lambda X.M)(B)$ *and* $M^* = [B/X]M$, $|R| \underset{\beta\beta^2}{\twoheadrightarrow}^+ |M^*|$.
3. *Given* $M \overset{\eta_*}{\to} \lambda x^A.M(x^A)$, $|M|^+ \twoheadrightarrow_\beta |\lambda x^{|A|}.M(x)|^+$.
4. *Given* $M \overset{\eta^2_*}{\to} \Lambda X.M(X)$, $|M|^+ \underset{\beta\beta^2}{\twoheadrightarrow} |\Lambda X.M(X)|^+$.

Proof. For (1), we have

$$\begin{array}{rcl} |R| & = & \Delta^n_{\tau(R)}(|\lambda x^A.M|(|N|^+)), \text{ by definition} \\ & \twoheadrightarrow_\beta & \Delta^n_{\tau(R)}((\lambda x^{|A|}.|M|^+)(|N|^+)), \text{ by Proposition 5 (1)} \\ & \to_\beta & \Delta^n_{\tau(R)}([|N|^+/x^{|A|}]|M|^+) \\ & = & \Delta^n_{\tau(R)}(|M^*|^+), \text{ by Proposition 4 (2)} \\ & = & |M^*|^+, \text{ since } \tau(R) = \tau(M^*) \end{array}$$

For (2), we have

$$\begin{array}{rcl} |R| & = & \Delta^n_{\tau(R)}(|\Lambda X.M|(|B|)(\Delta_B)), \text{ by definition} \\ & \underset{\beta\beta^2}{\twoheadrightarrow} & \Delta^n_{\tau(R)}((\Lambda X.\lambda v^{X\to X}.|M|^+)(|B|)(\Delta_B)), \text{ by Proposition 5 (2)} \\ & \underset{\beta\beta^2}{\twoheadrightarrow}^+ & \Delta^n_{\tau(R)}([\Delta_B/v^{|B|\to|B|}][|B|/X]|M|^+) \\ & = & \Delta^n_{\tau(R)}(|M^*|^+), \text{ by Proposition 4 (3)} \\ & = & |M^*|^+, \text{ since } \tau(R) = \tau(M^*) \end{array}$$

For (3), we have

$$\begin{array}{rcl} |M|^+ & = & \Delta_{\tau(M)}(|M|), \text{ since } M \text{ is not a } \lambda\text{-abstraction} \\ & \twoheadrightarrow_\beta & \lambda y^{|A|}.\Delta_B(|M|(\Delta_A(y^{|A|}))), \\ & & \text{by Proposition 5 (1), where } A \to B = \tau(M) \\ & = & \lambda y^{|A|}.|M(y)|^+, \text{ by definition} \\ & = & \lambda x^{|A|}.|M(x)|^+, \text{ by } \alpha\text{-conversion} \\ & = & |\lambda x^{|A|}.M(x)|^+, \text{ by definition} \end{array}$$

For (4), we have

$$\begin{aligned}
|M|^+ &= \Delta_{\tau(M)}(|M|), \text{ since } M \text{ is not a } \Lambda\text{-abstraction} \\
&\underset{\beta\beta^2}{\twoheadrightarrow} \Lambda X.\lambda v^{X\to X}.\Delta_A(|M|(X)(v^{X\to X})), \\
&\quad \text{by Proposition 5 (1), where } \forall X.A = \tau(M) \\
&= \Lambda X.\lambda v^{X\to X}.|M(X)|^+, \text{ by definition} \\
&= |\Lambda X.M(X)|^+, \text{ by definition}
\end{aligned}$$

□

Note that $\underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow}$ is defined as $\to_\beta \cup \to_{\beta^2} \cup \to_{\eta_*} \cup \to_{\eta_*^2}$ and $\underset{\beta\beta^2}{\longrightarrow}$ as $\to_\beta \cup \to_{\beta^2}$. We now state and prove our main simulation result.

Theorem 7. *(Simulation) $M \underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow} M'$ implies $|M|^+ \underset{\beta\beta^2}{\twoheadrightarrow}^+ |M'|^+$.*

Proof. We proceed by a structural induction on M to show the followings simultaneously.

1. $M \underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow} M'$ implies $|M| \underset{\beta\beta^2}{\twoheadrightarrow}^+ |M'|$ unless $M \xrightarrow{\eta_*} M'$ or $M \xrightarrow{\eta_*^2} M'$.
2. $M \underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow} M'$ implies $|M|^+ \underset{\beta\beta^2}{\twoheadrightarrow}^+ |M'|^+$.

Note $x^{a\to b} \xrightarrow{\eta_*} (\lambda y^a.x^{a\to b}(y^a))$, but $x^{a\to b}$, a version of $|x^{a\to b}|$, is in β-normal form. This explains why the restriction on (1) is necessary. We now do a case analysis on the structure of M.

– $M \xrightarrow{\beta} M'$, $M \xrightarrow{\beta^2} M'$, $M \xrightarrow{\eta_*} M'$ or $M \xrightarrow{\eta_*} M'$. This follows from Lemma 6.

– $M = M_1(N)$ and $M' = M_1'(N)$ and $M_1 \underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow} M_1'$. Then $M_1 \xrightarrow{\eta_*} M_1'$ cannot hold. By induction hypothesis on (1), $|M_1| \underset{\beta\beta^2}{\twoheadrightarrow}^+ |M_1'|$. Hence,

$$|M| = \Delta^n_{\tau(M)}(|M_1|(|N|^+)) \underset{\beta\beta^2}{\twoheadrightarrow}^+ \Delta^n_{\tau(M)}(|M_1'|(|N|^+) = |M'|.$$

Therefore (1) holds. (2) follows immediately.

– $M = N(M_1)$ and $M' = N(M_1')$ and $M_1 \underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow} M_1'$. By induction hypothesis on (2), $|M_1|^+ \underset{\beta\beta^2}{\twoheadrightarrow}^+ |M_1'|^+$. Hence,

$$|M| = \Delta^n_{\tau(M)}(|N|(|M_1|^+)) \underset{\beta\beta^2}{\twoheadrightarrow}^+ \Delta^n_{\tau(M')}(|N|(|M_2|^+) = |M'|.$$

Therefore (1) holds. (2) follows immediately.

– $M = M_1(B)$ and $M' = M_1'(B)$. Then $M_1 \xrightarrow{\eta_*^2} M_1'$ cannot hold. By induction hypothesis on (1), $|M_1| \underset{\beta\beta^2}{\twoheadrightarrow}^+ |M_1'|$. This implies that (1) and (2) hold.

– $M = \lambda x^A.M_1 \underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow} \lambda x^A.M_1' = M'$ or $M = \Lambda X.M_1 \underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow} \Lambda X.M_1' = M'$. (1) and (2) follow from induction hypothesis.

□

4 $\underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow}$ is strongly normalizing and confluent

Given some reduction $\rightarrow$, we write $\lambda 2 \models \mathcal{SN}(\rightarrow)$ meaning that $\rightarrow$ is strongly normalizing in $\lambda 2$, i.e., there exist no infinite $\rightarrow$-reduction sequences starting from $\lambda 2$-terms.

Theorem 8. $\lambda 2 \models \mathcal{SN}(\underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow})$ *if and only if* $\lambda 2 \models \mathcal{SN}(\underset{\beta\beta^2}{\longrightarrow})$.

Proof. If there exists an infinite $\underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow}$-reduction sequence

$$M_1 \underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow} M_2 \underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow} M_3 \underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow} \cdots,$$

then by Theorem 7 we have the following corresponding sequence

$$|M_1|^+ \underset{\beta\beta^2}{\twoheadrightarrow}^+ |M_2|^+ \underset{\beta\beta^2}{\twoheadrightarrow}^+ |M_3|^+ \underset{\beta\beta^2}{\twoheadrightarrow}^+ \cdots.$$

Therefore, $\lambda 2 \models \mathcal{SN}(\underset{\beta\beta^2}{\longrightarrow})$ implies $\lambda 2 \models \mathcal{SN}(\underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow})$. The other direction is trivial. □

Notice that this is a proof which can be formulated in the first-order Peano arithmetic.

Corollary 9. *Every $\lambda 2$-term is strongly $\underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow}$-normalizing and confluent.*

Proof. Since it is well-known that $\lambda 2 \models \mathcal{SN}(\underset{\beta\beta^2}{\longrightarrow})$, $\lambda 2 \models \mathcal{SN}(\underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow})$ follows immediately. It can be verified that $\underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow}$ is weakly confluent [8]. Therefore, $\underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow}$ is confluent by Newman's Lemma.

Roberto Di Cosmo and Delia Kesner have also proven Corollary 9 in [9] with a method involving *reducibility candidates* due to Tait [21] and Girard [14]. Hence, their proof is not modular. Also a modular proof for the strong normalization and confluence of a subsystem of $\lambda\beta\beta^2\eta_*\eta_*^2$ is presented in [10], but the proof — in the author's opinion — is very much involved and can hardly be scaled to other more complicated systems such as $\lambda\beta\beta^2\eta_*\eta_*^2$.

5 Conclusions and Future Work

We have demonstrated how to simulate η-expansion with β-reduction in the second-order polymorphic λ-calculus $\lambda 2$. This yields a proof of the equivalence between $\lambda 2 \models \mathcal{SN}(\underset{\beta\beta^2}{\longrightarrow})$ and $\lambda 2 \models \mathcal{SN}(\underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow})$, which can be formulated in the first-order Peano arithmetic. A clean modular proof of $\lambda 2 \models \mathcal{SN}(\underset{\beta\beta^2\eta_*\eta_*^2}{\longrightarrow})$

follows immediately. We intend to investigate the effects of augmenting first-order algebraic rewriting systems with $\xrightarrow[\beta\beta^2\eta_*\eta_*^2]{}$. Improvements on [5] seems to be immediate. Also we shall study the use of our technique in more complicated λ-calculi such as the high-order polymorphic λ-calculus $\lambda\omega$ and the construction of calculus λC. Some current work on combining algebraic term rewriting systems with $\lambda\beta\beta^2\eta_*\eta_*^2$ can be found through the pointer below.

`http://www.cs.cmu.edu/ hwxi/papers/TRS.ps`

6 Acknowledgement

I thank Frank Pfenning, Peter Andrews and Richard Statman for their support and for providing me a nice work environment. I also thank Roberto Di Cosmo for his comments on a draft of the paper.

References

1. Y. Akama (1993), On Mints' reduction for ccc-calculus. In *Typed lambda-calculi and applications*, vol. 664 of LNCS, pp 1-12.
2. H.P. Barendregt (1984), The Lambda Calculus: Its Syntax and Semantics, *North-Holland publishing company, Amsterdam.*
3. 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.
4. Val Breazu-Tannen and Jean Gallier (1991), Polymorphic rewriting conserves strong normalization, *Theoretic Computer Sicence*, vol. 83, pp 3-28.
5. Val Breazu-Tannen and Jean Gallier (1994), Polymorphic rewriting preserves algebraic confluence, *Information and Computation*, vol. 114(1), pp. 1-29.
6. Djordje Cubric (1992), On free ccc, *Manuscripts.*
7. R. Di Cosmo and D. Kesner (1994), Combining the first order algebraic rewriting systems, recursion and extensional lambda calculi. In *Serge Abiteboul and Eli Shamir, editors, International Conference on Automata, Languages and Programming*, vol. 820 of LNCS, pp. 462-472.
8. R. Di Cosmo and D. Kesner (1994), Simulating expansions without expansions. *Mathematical Structures in Computer Science*, vol. 4, pp. 1-48.
9. R. Di Cosmo and D. Kesner (1995), Rewriting with polymorphic extensional lambda-calculus. In *Proceedings of Computer Science Logic '95*, vol. 1092 of Lecture Notes in Computer Science, pages 215-232.
10. R. Di Cosmo and A. Piperno (1995), Expanding Extensional Polymorphism, In *Proceedings of Typed Lambda-Calculi and Applications*, vol. 902 of LNCS, pp. 139-153.
11. Daniel J. Dougherty (1993), Some lambda calculi with categorical sums and products. In *Proceedings of the 5th International Conference on Rewriting Techniques and Applications.*
12. N. Ghani (1995), $\beta\eta$-equality for coproducts. In *Typed lambda-calculi and applications*, vol. 902 of LNCS, pp. 171-185.
13. N. Ghani (1996), Eta Expansions in System F, *Manuscripts.*

14. J.-Y. Girard (1972), Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur, *Thèse de doctorat d'etat, Université Paris VII.*
15. C.B. Jay and N. Ghani (1996), The virtues of eta-expansion, *Journal of Functional Programming*, vol. 5(2), pp. 135-154.
16. J. Lambek and P.J. Scott (1986), *An introduction to higher order categorical logic*, Cambridge University Press.
17. G.E. Mints (1979), Theory of categories and theory of proofs (I). In *Urgent Question of Logic and the Methodology of Science* [In Russian], Kiev.
18. D. Prawitz (1971), Ideas and results of proof theory, Proceedings of the 2nd Scandinavia logic symposium, *editor J.E. Fenstad, North-Holland Publishing Company, Amsterdam.*
19. J. Reynolds (1974), Towards a theory of type structure, *Colloquium sur la Progrmmation*, vol. 19 of LNCS, pp. 408-423.
20. H. Schwichtenberg (1991), An upper bound for reduction sequences in the typed lambda-calculus, *Archive for Mathematical Logic*, 30:405-408.
21. W. Tait (1967), Intensional Interpretations of functionals of finite type I, *J. symbolic logic 32*, pp. 198-212.
22. H. Xi (1996), Upper bounds for standardizations and an application, *Research Report 96-189, Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh.*
23. H. Xi (1996), Simulating eta-expansions with beta-reductions in the second-order polymorphic lambda-Calculus, *Research Report, Department of Mathematical Sciences, Carnegie Mellon University, Pittsburgh.* Available through pointer: `http://www.cs.cmu.edu/~hwxi/papers/EtaSim.ps`