Simply adding lambda won't make your rewrites diverge

Ezra Cooper

February 28, 2010

Mitsuhiro Okada showed in 1989 that the addition of simply-typed lambda calculus to a convergent (confluent and strongly normalizing) term-rewriting system is still convergent.[1] I had trouble understanding this paper because of its unfamiliar notations, so this document tries to restate the result in terms more similar to the other papers I've found in the literature. Here I prove only the result for simply-typed λ-calculus, but Okada also shows it for System F.

With this theorem in hand, we can safely add functional abstraction (provided it keeps a simply-typed discipline) to any first-order rewrite system and be confident that rewriting the combined calculus still terminates.

[1] Mitsuhiro Okada. Strong normalizability for the combined system of the typed lambda calculus and an arbitrary convergent term rewrite system. In *Proceedings of the ACM SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation*, pages 357–363. ACM, 1989

The setting is as follows. We have two classes of terms: one called *rewrite terms* which are composed only of function symbols and variables:

$$\begin{array}{rcll} u, v, w & ::= & x \mid f(u_1, ..., u_n) & \text{rewrite terms} \\ f & & & \text{function symbols} \end{array}$$

And another class of lambda terms, which includes the above, formed as follows:

$$L, M, N, P \quad ::= \quad x \mid LM \mid \lambda x.N \mid f(M_1, \ldots M_n) \qquad \text{lambda terms}$$

These are typed in a simple type system consisting of base types B and arrow types:

$$S, T, U \quad ::= \quad B \mid S \to T \qquad \text{types}$$

The typing rules are as follows:

$$\Gamma x : T \vdash xT \qquad \frac{\Gamma, x : S \vdash N : T}{\Gamma \vdash \lambda x.N : S \to T}$$

$$\frac{\Gamma \vdash L : S \to T \qquad \Gamma \vdash M : S}{\Gamma \vdash LM : T} \qquad \frac{\Gamma \vdash M_i : B_i \text{ for each } i \qquad \vdash f : B_1 \times \cdots B_n \to B_{n+1}}{\Gamma \vdash f(M_1, \ldots, M_n) : B_{n+1}}$$

The rule for function symbols indicates that they always take arguments of base type and form a term of base type.

We require the existence of at least one term of each base type—for example, a constant.

We write $M[N/x]$ for the capture-avoiding substitution of lambda-term N for x in the lambda-term M; we can also substitute lambda terms into rewrite terms to produce lambda terms ($u[N/x]$) and rewrite terms into rewrite terms to produce rewrite terms ($u[v/x]$). The letters σ and ρ will range over substitutions, that is, compositions of simple substitutions like $[N/x]$. The composition is written $\sigma \cdot \rho$ Terms are considered equal when they are α-equivalent.

Substitutions can be typed by typing all their terms. We will write $\rho : \Gamma$ if each $[M/x]$ in ρ is $\Gamma \vdash x : T$ and $\vdash M : T$. This requires the terms M to be closed, as they are typed in an empty context.

We are given a family of rewrite rules on the rewrite terms, which must be type-preserving:

$$u : B \rightsquigarrow v : B$$

The rewrite rules are understood to define a rewrite system over the lambda terms by compatibility with substitution: $u \rightsquigarrow v$ implies $u\rho \rightsquigarrow v\rho$.

One further reduction applies to the class of lambda terms:

$$(\lambda x.N)M \longrightarrow_\beta N[M/x]$$

This rule and the given rewrite rules apply in any term context.

WE WANT TO SHOW that all terms in this language are strongly normalizing, a property we can hope for since the typed lambda calculus, on its own, is s.n., as is the rewrite system on its own.

The proof uses the Tait-Girard method of reducibility predicates.[2]

Define a (type-indexed) family of predicates red_T on closed terms as follows:

- $M \in \mathsf{red}_B$ iff $M : B$ strongly normalizes.
- $L \in \mathsf{red}_{S \to T}$ iff for all $P \in \mathsf{red}_S$ we have $LP \in \mathsf{red}_T$

We can pronounce $M \in \mathsf{red}_T$ as "M is reducible at T."

For the purpose of inducting on reduction trees, we define the notion of maximum reduction length, defined for terms that strongly normalize:

$$\operatorname{maxred} M = \begin{cases} 1 + \max_{M', M \rightsquigarrow M'} (\operatorname{maxred} M') & \text{if } M \text{ has reducts} \\ 0 & \text{if } M \text{ is normal} \end{cases}$$

We will also write $\operatorname{maxred} \sigma$ for the sum of $\operatorname{maxred} M_i$ where $[M_i/x_i] \in \sigma$.

We assume bound variables in a term and those in its typing context are all distinct.

[2] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Cambridge University Press, New York, 1989

Lemma 1. *If $M \in \mathsf{red}_S$ and $M \longrightarrow M'$ then $M' \in \mathsf{red}_S$.*

Proof. By induction on the type S.

At base type, note that a reduct of a strongly normalizing term must also be s.n.

At arrow type, $T \to U$, we have that for any $P \in \mathsf{red}_T$, the application $MP \in \mathsf{red}_U$, and we need to show that for any $P \in \mathsf{red}_T$, the counterpart $M'P \in \mathsf{red}_U$. This is immediate from the induction hypothesis and the fact that $MP \longrightarrow M'P$. □

Lemma 2. *For any type S,*

(i) there is a reducible term of type S,

(ii) for any $M \in \mathsf{red}_S$ we have that M strongly normalizes, and

(iii) if M is an application and every M' for which $M \longrightarrow M'$ has $M' \in \mathsf{red}_S$, then $M \in \mathsf{red}_S$.

Proof. By induction on the type.

- Case B.

  (i) We have assumed the existence of at least one term of each base type. Since the rewrite system itself is strongly normalizing, some such term at each type must be strongly normalizing, hence reducible at B.

  (ii) The fact that a reducible term at B strongly normalizes is immediate from the definition of reducibility.

  (iii) The fact that M strong normalizes is immediate from the fact that all reducts are strongly normalizing.

- Case $S \to T$.

  (i) The IH (part (i)) produces a reducible term $N : T$. Construct $\lambda x.N : S \to T$. (Since N is closed, x is a dummy variable.) Now to show that $\lambda x.N$ is reducible, we show that $(\lambda x.N)M$ is reducible for each $M \in \mathsf{red}_S$. We show this by induction on maxred N + maxred M. (These values are well-defined because reducible terms are strongly normalizing, by the IH, part (ii)). In particular, we show that every reduct of $(\lambda x.N)M$ is reducible, and since it is an application, this entails (by the IH, part (iii)) that $(\lambda x.N)M$ itself is reducible. The reducts are as follows:

  - $(\lambda x.N')M$ with $N \longrightarrow N'$; here the (inner) IH applies;
  - $(\lambda x.N)M'$ with $M \longrightarrow M'$; here the (inner) IH applies;
  - $(\lambda x.N)M \longrightarrow_\beta N[M/x] = N$; here N was constructed to be reducible.

(ii) Given $M \in \mathsf{red}_{S\to T}$, we can construct $MP \in \mathsf{red}_T$ using an arbitrary $P \in \mathsf{red}_S$ as promised by the IH (part (i)). Then by the IH (part (ii)), MP strongly normalizes. And as a subterm thereof, M also strongly normalizes.

(iii) We need to show that for any $P \in \mathsf{red}_S$, the application $MP \in \mathsf{red}_T$. We show this by induction on maxred P, which is well-defined because P strongly normalizes according to the IH (part (ii)). Since M is an application, it does not react directly with P; any reductions are within M or within P. Now we see that all reducts of MP are in red_T: a reduct $MP \longrightarrow M'P \in \mathsf{red}_T$ by assumption, while a reduct $MP \longrightarrow MP'$ reduces the (inner) induction metric and hence the (inner) induction hypothesis gives us $MP' \in red_T$. And so, since all reducts of MP are in red_T, the outer induction hypothesis applies to give $MP \in \mathsf{red}_T$ and hence $M \in \mathsf{red}_{S\to T}$. □

Lemma 3. *Any base-type term M can be decomposed as $M = u\rho$ where u is a rewrite term, $\rho = [M_1/x_1, \ldots, M_n/x_n]$, each M_i has a non-function symbol at its root, and each M_i has base type.*

Proof. By induction on the structure of M.

For the base case, If M does not have a function symbol at its root, then let $u = x$, where x is a fresh variable, and $\rho = [M/x]$.

For the inductive case where M has a function symbol at its root, as $M = f(N_1, \ldots, N_n)$, we know that each of the arguments $N_1, \ldots, N_n$ has base type and so we can apply the inductive hypothesis to form terms $u_1\rho_1, \ldots, u_n\rho_n$ and collect these to form our goal, $M = u\rho = f(u_1, \ldots, u_n)\rho_1 \cdots \rho_n$. □

Lemma 4. *Given a rewrite term u typed $\Gamma \vdash u : B$ and a substitution $\rho : \Gamma$ having $\rho \in \mathsf{red}_\Gamma$ where the roots of the mapped elements of ρ are not function symbols, we have $u\rho \in \mathsf{red}_B$.*

Proof. By induction on maxred u + maxred ρ. The value maxred u is well-defined because the system R is strongly-normalizing, and so is maxred ρ because the terms of ρ are reducible. Since $u\rho$ has base type, we need only show that it strongly normalizes. Since the roots of the terms in ρ are not function symbols, and ρ has no application nodes, each reduction on $u\rho$ is either within ρ or has the form $u\rho \rightsquigarrow v\rho$. If the reduction is within ρ then we have reduced maxred ρ and the goal follows by the IH. If the reduction is $u\rho \rightsquigarrow v\rho$ then we have reduced maxred u and the goal follows by the IH. □

Theorem 1. *For every term M typed as $\Gamma \vdash M : T$ and substitution $\sigma : \Gamma$ with $\sigma \in \mathsf{red}_\Gamma$, we have $M\sigma \in \mathsf{red}_T$.*

Proof. By *strong* induction on the structure of M. (The strong induction principle allows us to derive the property for a term from the assumption that all of its subterms have the property.)

- Case x.

  This variable must appear in the context and is mapped by σ, hence its reducibility is immediate from hypotheses.

- Case $LM : T$.

  By IH, $L\sigma \in \mathsf{red}_{S\to T}$ and $M\sigma \in \mathsf{red}_S$. The definition of $\mathsf{red}_{S\to T}$ gives us $L\sigma M\sigma = (LM)\sigma \in \mathsf{red}_T$.

- Case $\lambda x.N : S \to T$.

  We must show that $((\lambda x.N)\sigma)M \in \mathsf{red}_T$ for each $M \in \mathsf{red}_S$. We show this by induction on $\mathrm{maxred}\,(N\sigma) + \mathrm{maxred}\,M$. (We can assume safely that σ does not substitute x for if it does it has no effect and we can work with the substitution where it is elided.)

  Consider all the reductions of $((\lambda x.N)\sigma)M$. We have the β-reduction $((\lambda x.N)\sigma)M \longrightarrow_\beta N\sigma[M/x]$. This reduct is in red_T by the IH, noting that $\sigma \cdot [M/x]$ is a closing, reducible subsitution.

  All other reductions are within N or within M, and preserve reducibility of those terms (by Lemma 1). That $((\lambda x.N)\sigma)M$ in reducible at T then follows from the (inner) IH.

  Since $((\lambda x.N)\sigma)M$ is an application, Lemma 2 lets us retract the property of reducibility from that of its set of reducts; hence the term itself is reducible.

  This satisfies the deifinition of $\mathsf{red}_{S\to T}$ so now $(\lambda x.N)\sigma$ is also reducible.

- CASE $f(\vec{M})$.

  The term $f(\vec{M})\sigma$ can be decomposed as $u\rho$ for some rewrite term u and substitution $\rho : \Gamma'$ having $\rho \in \mathsf{red}_{\Gamma'}$ with the terms of ρ having no function symbols at their roots. Now the terms of ρ are all reducible by virtue of being subterms of the $\vec{M}$ terms, which are all base type. So Lemma 4 applies and $u\rho$ is reducible. □

SINCE all closed terms in the language are reducible, they are strongly normalizing, and so too must be their open cousins.

Conclusions & Next steps

We've seen that first-order term-rewriting systems can be extended with functional abstraction and its counterpart, application, without

disturbing the strong normalization of the combined system, using a very explicit and conventional set of notations.

The next steps are to extend this proof to System F and to investigate how well it extends to higher-order rewrite systems.

References

[1] Jean-Yves Girard, Yves Lafont, and Paul Taylor. *Proofs and Types*. Cambridge University Press, New York, 1989.

[2] Mitsuhiro Okada. Strong normalizability for the combined system of the typed lambda calculus and an arbitrary convergent term rewrite system. In *Proceedings of the ACM SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation*, pages 357–363. ACM, 1989.