STRONG NORMALIZABILITY FOR THE COMBINED SYSTEM OF THE TYPED LAMBDA CALCULUS AND AN ARBITRARY CONVERGENT TERM REWRITE SYSTEM *)

Mitsuhiro Okada
Department of Computer Science
Concordia University
Montreal, Quebec Canada

ABSTRACT: We give a proof of strong normalizability of the typed λ-calculas extended by an arbitrary convergent term rewriting system, which provides the affirmative answer to the open problem proposed in Breazu-Tannen [1]. Klop [6] showed that a combined system of the untyped λ-calculus and convergent term rewriting system is not Church-Rosser in general, though both are Church-Rosser. It is well-known that the typed λ-calculus is convergent (Church-Rosser and terminating). Breazu-Tannen [1] showed that a combined system of the typed λ-calculus and an arbitrary Church-Rosser term rewriting system is again Church-Rosser. Our strong normalization result in this paper shows that the combined system of the typed λ-calculus and an arbitrary convergent term rewriting system is again convergent. Our strong normalizability proof is easily extended to the case of the second order (polymorphically) typed lambda calculus and the case in which μ-reduction rule is added.

*) This paper is based on one of the author's lectures at Laboratoire de Recherche en Informatique, Université de Paris-Sud, in Fall 1988, (cf. § 1.4 of the author's lecture notes "Introduction to Proof Theory to Computer Science", LRI, Université de Paris-Sud, Orsay, France, 1988).

After he finished the final version of this paper, the author was informed from Jean Gallier that they got the same results as those in this paper. cf. V. Breazu-Tanner and Gallier, Polymorphic Rewriting Conserves Algebraic Strong Normalization and Confluence, preprint, 1989.

A convergent term rewrite system gives a mathematical foundation of effective execution for equational programming languages and equational algebraic specification languages such as OBJ, and a system of lambda calculus gives a mathematical foundation of execution model for functional programming languages such as LISP, ML. A combination of a term rewrite system and a system of lambda calculus is very attractive from the point of view of binding the two paradigms, paradigm of equational language and paradigm of functional language. A natural and the most basic question was under which condition a combined system of a lambda calculus system and a term rewrite system has an appropriate property for effective execution or computation. In this note we give an answer by showing a proof of strong normalizability of the typed λ-calculus extended by an arbitrary convergent term rewriting system, which provides the affirmative answer to the open problem proposed in Breauzu-Tannen [1]. Klop [6] showed that a combined system of the untyped λ-calculus and convergent term rewriting system is not Church-Rosser in general, though both are Church-Rosser. It is well-known that the typed λ-calculus is convergent (Church-Rosser and terminating). Breazu-Tannen [1] showed that a combined system of the typed λ-calculus and an arbitrary Church-Rosser term rewriting system is again Church-Rosser. Our strong normalization result in this paper shows that the combined system of the typed λ-calculus and an arbitrary convergent term rewriting system is again convergent.

We first recall the basic notions for a term rewriting system, (cf. eg. Huet-Oppen [5], Dershowitz-Okada [2]). Let L_R be a set of (first order) function symbols and constant symbols, T_R be the set of first order terms composed of L_R and the first order variables. A rewrite system R is defined by a (finite or infinite) set of rules of the form $s \to t$ ("term s is reduced to term t"), where s and t are terms in T_R. The set L_R is called the signature for the rewrite system R. We also use the expression $u \to v$ for one step reduction in R, i.e. one application of a rewrite rule of R for term u; more precisely, if u is of the form $w[s]\sigma$, where w[s] means s occurs in a term w, $t\sigma$ means a substitution instance of term w by substitution operator σ, and if there is a rewrite rule of the form $s \to t$ in R, then we denote $u \to v$, where v is of the form $w[t]\sigma$, where $w[t/s]\sigma$, where [t/s] means the term obtained from w[s] by replacing the indicated occurrence of s by t. $\to*$ is the reflexive-transitive closure of $\to$. $s{\downarrow}t$ means there exists a term u such that $s \to* u *\!\leftarrow t$. $s =\!* t$ means there exist u, v, ..., w such that $s{\downarrow}u{\downarrow}v{\downarrow}...{\downarrow}w{\downarrow}t$. A rewrite system R is called Church-Rosser if for any terms s and t, $s =\!* t$ implies $s{\downarrow}t$. R is terminating if for any term s, any reduction path from s terminates. An irreducible term is called a normal form. A rewrite system is called convergent if it is Church-Rosser and terminating. A rewrite system R is convergent if and only if each term in R has a unique normal form. If a rewrite system R is Church-Rosser then

$$R \vdash s{\downarrow}t \text{ if and only if } E \vdash s = t,$$

where $E\vdash s = t$ means that $s = t$ is provable in the equational logic from the set of axioms of the form $u = v$ where each axiom $u = v$ is obtained from a rewrite rule of the form $u \to v$ in R; in other words, the joinability in R is equivalent to provability of equality in the corresponding equational theory.

We consider a many sorted equational theory and the corresponding many sorted rewrite theory; each function symbol and constant symbol in the signature of the theory have indicated sorts as follows:

f: $P_1 \times P_2 \times \dots\ P_n \to Q$ means that function f has the type $P_1 \times \dots\ P_n \to Q$, where $P_1, \dots, P_n$, Q are sorts, in other words, f is an n-ary function whose i-th domain is sort Pi and whose range is sort Q.

Now we define the language of the pure typed lambda calculus.

Definition of type. Given set $\{P_1, P_2, \dots, P_n\}$ of base types we define the notion of (higher) types, inductively as follows.

(1) If P is a base type then P is a type.
(2) If A and B are types then $A \Rightarrow B$ is a type.

We abbreviate $A_1 \Rightarrow (A_2 \Rightarrow \dots\ (A_n \Rightarrow B) \dots))$ as $A_1 \times A_2 \times \dots\ A_n \Rightarrow B$.

Definition of λ-terms. The language have countably many (free and bound) variables, say $x^A1, x^A2, \dots\ x^An, \dots$, for each type A. It also may contain constants, say c of type A.

(1) If X^A is a variable of type A then it is a λ-term of type A. If c^A is a constant of type A then it is an λ-term of type A.

(2) If t is a λ-term of type B and if x^A is a variable of type A then $\lambda x^A t$ is a λ-term of type $A \Rightarrow B$.

(3) If t is a λ-term of type $A \Rightarrow B$ and if s is a λ-term of type A then t(s) is a λ-term of type B.

By the simple typed lambda calculus we mean the reduction system with the following reduction rule.

β-reduction rule:

$(\lambda x^A t[x^A])(s) \Rightarrow t[s/x^A]$, where s is a λ-term of type A.

We consider the combined system of an arbitrary convergent term rewriting system and the typed λ-calculus. The language of λ-calculus is naturally extended by adding the signature of the rewrite system, where sorts $P_1, \dots, P_n$ of the rewrite system are interpreted as base types.

The following is the main result of this note:

Theorem 1. If a rewrite system is convergent then the combined system of the rewrite system and the typed λ-calculus is also convergent.

The polymorphically typed λ-calculus (or 2nd order typed λ-calculus) is obtained from the typed λ-calculus by extending as follows:

If A[B] is a type, where B is a subexpression of A, then ∀XA[X] is also a type.

If t[B] is a term of type A[B] where t[B] means that B appears in t as a type name, then △Xt[X] is a term of type ∀XA[X].

If △Xt[X] is a term of type ∀XA[X] and if B is a type, then t(B) is a term of type A[B].

Then we have the reduction corresponding to the β-reduction above.

Second order β-reduction rule

$$(\triangle X t[X])\ (B) \Rightarrow t[B/X]$$

Theorem 2. If a rewrite system is convergent then the combined system of the rewrite system and the polymorphically typed λ-calculus is also convergent.

For the proof of Theorems 1 and 2, we actually prove the strong normalizability of the combined system. For this purpose we modify Girard-Martin-Lof-Prawitz's technique for the strong normalization in [4], [7], [9]. We first prove Theorem 1.

Definition. For a term t we define the cap, the estimated cap, and the upper part of t. Assume that t is of the form $s(u_1, \ldots, u_n)$, where s is the maximum subterm of t such that s contains the top operator (the root) and that s is a term of the original language of R. Then s is the cap of t. If pure λ-terms except for variables of base types are attached to the cap, then all such λ-terms of each base type are regarded as occurrences of a new and the same variable symbol of that base type when the cap is defined. A variable of a base type attached to the cap remains. More precisely, we define the cap of a term as follows:

(1) If a root f of t is a function symbol which belongs to the signature of R, then f belongs to the cap.

(2) If t is of the form $s[u_1/x_1, \cdots u_n/x_n, f(v_1, \cdots, v_m)/x_{n+1}]$ and a part $s[*/x_1, \cdots, */x_n, */x_{n+1}]$ of t belongs to the cap, where f is a function symbol which belongs to the signature of R, then the part $s[*/x_1, \cdots, */x_n, f(*, \cdots, *)/x_{n+1}]$ also belongs to the cap.

(3) If t is of the form $s[u_1/x_1, \cdots, u_n/x_n, c/x_{n+1}]$ and a part $s[*/x_1, \cdots, */x_n, */x_{n+1}]$ of t belongs to the cap, where c is a constant symbol which belongs to the signature of R, then the part $s[*/x_1, \ldots, */x_n, c/x_{n+1}]$ also belongs to the cap.

(4) If t is of the form $s[u_1/x_1, \ldots, u_n/x_n, y]$, and a part $s[*/x_1, \cdots, */x_n, */y]$ of t belongs to the cap, where x is a variable of a sort (a base type) in R, then the part $s[*/x_1, \cdots, */x_n, */y]$ also belongs to the cap.

(5) If t is of the form $s[v_1/x_1, \cdots, v_k/x_k, u_1/y_1, \cdots, u_n/y_n, w_1/z_1, \cdots, w_m/z_m]$ and the expression $s[*/x_1, \cdots, */x_k, */y_1, \cdots, */y_n, z'/z_1, \cdots, z'_m/z_m]$ belongs to the cap of t for suitable variables $z'_1, \ldots, z'_m$, where u_1 is all the occurrences of those terms of sort P whose outermost operators do not belong to the signature of R and v_i and w_i are all the occurrences of those terms of other sorts whose outermost operators do not belong to the language of R, then the expression $s[*/x_1, \cdots, */x_k, y/y_1, \cdots, y/y_n, z'_1/z_1, \cdots, z'_m/z_m]$ also belongs to the cap of t, where y is a new variable of sort P.

It is noted that the cap of a term t is a term of R but not a subterm of t in general because of the condition (5), and that the cap is determined uniquely up to the choice of new variable symbols introduced in (5).

For example, if $s[z^{(A\Rightarrow(B\Rightarrow A))\Rightarrow C}(\lambda y^{A},\lambda x^{B}y^{A}),$ $x'^{(C\Rightarrow C)\Rightarrow B}(\lambda y'^{C}(y'^{C})),$ $w^{B\Rightarrow C}(v^{B}),$ $u^{C}]$ then the cap of the end part is $s[x_0^{C}, y_0^{B}, x_0^{C}, u^{C}]$.

Definition. The *estimated cap* (ec) s of a term t is the cap for the β-normal form βNF(t) of t, where the β-normal form βNF(t) of t is the irreducible form of t with respect to the β-reduction. Note that, since the system is convergent with respect to the β-reduction, the β-normal form is always determined uniquely, hence the estimated cap is uniquely defined for a given term t.

We introduce a notion of reducibility predicate R_A of each type A, following Girard-Martin-Lof-Prawitz (cf. [7]).

Definition.

(1) When t is of the form $\lambda x A_s[x^A]$,

$R_{A\Rightarrow B}$ (t) if $\forall u \in A$ (if $R_A(u)$ then $R_B(s(u))$).

(2) When t is not of the form λxs, $R_A(t)$ if

(2.1) t is a normal form (i.e. irreducible) with respect to both β-reduction and the rewriting of R, or

(2.2) for any term u obtained by one step reduction from t, $R_A(u)$

Lemma 1. If $R_A(t)$ then t is strongly normalizable.

Proof. By induction on the construction of the reducibility predicate R.

(Case 1) If t is of the form $\lambda x^A s[x^A]$, then by the definition of R, $\forall u \in A$ (if $R_A(u)$ then $R_B(s(u))$). Take x^A for u. Then $R_B(s[x^A])$. By the induction hypothesis, $s[x^A]$ is strongly normalizable. Hence, $\lambda x^A s[x^A]$ is also strongly normalizable.

(Case 2) Otherwise, for any s which is obtained from t by one step reduction, s is strongly normalizable, hence t is also strongly normalizable.

Lemma 2. For any term s of type A, if $R_A(s)$ and $<_{red}$, then $R_A(t)$, where $<_{red}$ means the transitive-reflexive closure of the reductions (i.e. β-reduction and term rewrite reductions of R).

Proof is carried out by the induction argument on the construction of the predicate R, which is similar to the argument above.

Lemma 3. If $R_{A_1}(u_1)$, ... $R_{A_n}(u_n)$, then $R_B(t^*[u_1, ..., u_n])$ for any $t^*[x^{A}1, ..., x^{A}n]$, where $x^{A}1, ..., x^{A}1$ are all the free variables occurring in t^*.

Proof. By induction on the length of t^*. We denote t for $t^*[u_1, ..., u_n]$ in our proof.

(Case 1) If t^* has length 0, i.e. t^* is a variable or a constant, say a variable of the form $x^{A}i$. Then t is of the form u_i, hence by our assumption, $R_{A_i}(u_i)$ holds, therefore $R_A(t)$ holds.

(Case 2) t^* is of the form $\lambda x^A t_0$ of type $A \Rightarrow B$, hence t is of the form $\lambda x^A t_0[x^A, u_1, ..., u_n]$. By the induction hypothesis, $R_B(t_0[s, u_1, ..., u_n])$ for any s of type A such that $R_A(s)$. Hence $R_{A\Rightarrow B}(t)$.

(Case 3) t^* is of the form s(v), and t is of the form $s[u_1, ..., u_n](v[u_1, ..., u_n])$. By the induction hypothesis, $R_{A\Rightarrow B}(s[u_1, ..., u_n])$, and $R_A(v[u_1, ..., u_n])$.

(3.1) Take an arbitrary reduction path. Consider the case in which by successive reductions for $s[u_1, ..., u_n]$, one reaches $\lambda x^A s_0[x^A]$, and assume that at that time $v[u_1, ..., u_n]$ is reduced to v_0, and that till one reaches this form the outermost symbol is not λ. Hence by the definition of R,

$R_{A\Rightarrow B}(\lambda x^A s_0[x^A, u_1, ..., u_n])$ holds. Then by the definition of R and by the fact that $R_A(v_0)$ holds (Lemma 2), $R_B(s_0[v_0])$ holds. Since $s_0[v_0]$ is obtained from $s[u_1, ..., u_n](v[u_1, ..., u_n])$ by finite reductions, $R_B(t)$ holds.

(3.2) Consider the case in which one reaches a normal form whose outermost operator is the application operator. Then by the definition of R, $R_B(t)$.

(Case 4) The outermost operator is a function symbol, say f, in the signature of the rewrite system R. Then t and any reduced form of t have a base type, say P. In particular, the λ-symbol will not appear as the outermost symbol after any reductions for t. Hence, by the definition of R, to show $R_P(t)$, it suffices to see the strong normalizability of t. We show the strong normalizability by the transfinite induction on (ec(t), up(t)), where ec(t) is the estimated cap and up(t) is the upper part of t, i.e., the multi-set $\{s_1, ..., s_n\}$, the pair (ec(t), up(t)) is ordered lexicographically. Let $t = f(s_1, ..., s_n)$. By our induction hypothesis, $R_{P_1}(s_1), ..., R_{P_n}(s_n)$ for suitable base types $P_1, ..., P_n$, hence by Lemma 1, $s_1, ..., s_n$ are strongly normalizable. Hence up(t) is well-founded with respect to the reduction ordering. Since our rewrite system R is convergent, hence terminating, ec(t) is well-founded with respect to the R-reductions.

We consider the three subcases as follows:

(4.1) If $\hat{t}$ is obtained from t by one step β-reduction. Then obviously $up(t^\wedge) < up(t)$ and $ec(t^\wedge) = ec(t)$.

(4.2) If $\hat{t}$ is obtained from t by one-step R-reduction, and the R-reduction reduces only the part of ec(t). Then $ec(\hat{t}) < ec(t)$.

(4.3) If $\hat{t}$ is obtained from t by one-step R-reduction, and the R-reduction does not reduce any part of ec(t). Obviously, $up(\hat{t}) < up(t)$. On the other hand, as we see from Sublemmas 2 and 3 below, $ec(\hat{t}) \equiv ec(t)$.

Sublemma 1. If λ-terms $s_1, ..., s_n$ are of base types $P_1, ..., P_n$, and if $s_1, ..., s_n$ and $t[x^{P}1, ..., x^{P}n]$ are β-normal forms, then $t[s_1, ..., s_n]$ is also a β-normal form.

Proof. Obvious from the definition of the β-normal form.

Sublemma 2. If t is a β-normal form, and if s is obtained from t by an R-reduction which does not reduce any part of the cap of t, then $ec(t) \equiv ec(s)$.

Proof. Even if some R-term part collapses by the R-reduction, there is no further possility of β-reduction because of Sublemma 1 above, hence there is not further possiblity of collapsing any λ-term part. Hence $ec(t) \equiv ec(s)$.

Sublemma 3. (Breazu-Tannen [1]). If $t \rightarrow_R \hat{t}$, then the β-normal form of t can be R-reduced to the β-normal form of $t^\wedge$, by (possibly repeated use of) the same R-rule for $t \rightarrow_R \hat{t}$. In particular, if the reduction $t \rightarrow_R \hat{t}$ is in the Case (4.3) above, then $\beta NF(t) \rightarrow_R{}^* \beta NF(\hat{t})$ is obtained by the reductions only for the part which does not reduce the cap.

$$\begin{tikzcd}
t \arrow[r, "R"'] \arrow[d, "\beta *"'] & \hat{t} \arrow[d, "\beta *"'] \\
\beta NF(t) \arrow[r, "R *"'] & \beta NF(\hat{t})
\end{tikzcd}$$

Proof. By induction on the length of t. See [Breazu-Tannen 88].

From Sublemmas 2 and 3, one can easily see that if $\hat{t}$ is obtained from t, under the condition of (4.3), ec($\hat{t}$) $\equiv$ ec(t). End of Proof.

Since the Church-Rosser (confluence) property is preserved under the extension of typed lambda calculus by a confluent term rewrite system, our strong normalizability result provides Theorem 1 above.

Sketch of proof for Theorem 2.

The proof above of strong normalization can be extended to the combined system of the polymorphically (second order) typed lambda calculus and an arbitrary convergent rewrite system, using Girard's notion of the candidates of the reducibility predicates. For example, one can just follow the definition of [Martin-Löf 71] to define the candidates of the reducibility and to extend our reducibility predicate R_A of type A to an extended reducibility predicate $R_{A[\alpha_T/X,\alpha_S/Y,...]}$ of type A[X, Y, ...] with candidates α_T, α_S, ... of types T, S, ..., respectively, (cf. [Martin-Löf 71]). Then we can follow the same argument as before. Especially we follow essentially the same Lemmas, i.e., Lemma 1, Lemma 2 and Lemma 3 above, with the extended reducibility predicates. One can follow the traditional argument (due to Girard); induction argument on the construction of the extended reducibility predicte $R_{A(X,Y,...)}$ of type A(X,Y,...) for the proofs of Lemmas 1 and 2; induction argument on the length of t^* for the proof of Lemma 3. Again, the essential difference in our case is in (Case 4) of the proof for Lemma 3. In other words, we must add (Case 4) in the proof, since the traditional framework (of e.g. [Martin-Löf 71]) does not include this. The proof for (Case 4) can be carried out in exactly the same way as that for Lemma 3 above.

Extension to the η-reduction rule

Our strong normalizability result can also be extended to η-calculus with η-reduction as well as β-reduction. It is well-known that Girard's method (the method using the reducibility predicates) for the strong normalizability argument is not affected by inclusion of μ-reduction in the traditional framework. This is also true for our case.

Conclusion

Our strong normalization technique provides the following information:

Typed lambda calculus (with and without polymorphic types) extended by a convergent (or terminating) term rewrite system is also convergent (terminating, respectively). The termination result can also be extended by adding μ-reduction.

On the other hand, it is implicit in Gödel [Dialectica 1958] and in Girard [4] that the above convergence and termination results are also true for a higher order rewrite system if the rewrite rules are restricted to the form of higher order primitive recursion (i.e., the higher order primitive recursive definition based on given constructor terms), cf. Okada [8].

It is desirable to provide a criterion of convergence and termination of the combined system with a wider range of higher order rewrite rules, to develop a programming or specification language based on the combined paradigm of functional languages and equational languages.

ACKNOWLEDGEMENTS

This work was partly supported by the Natural Science and Engineering Research Council of Canada, grant OGP0036663 and ICR0039976, Fonds pour la Formation de Rechercheur et l'Aide à la Recherche, grant 89EQ4184, Social Science and Humanities Research Council of Canada, grant 410-88-0884, and the Committee on Aid to Research Activity of Concordia University. A part of this work was done when the author was visiting Université de Paris XI, Orsay, France. The author wishes to thank those who attended his lectures at Laboratoire de Recherche Informatique at Orsay, especially Professors Common, Gaudel, Hsiang, Jouannaud, Rusinowitch, for their comments. In particular, the discussion with Prof. Jouannaud was very essential for this work. The authors would also like to express their thanks to Professors Dershowitz, Lascanne, Remy, P. J. Scott and Toyama for discussion on related topics.

REFERENCES

1. V. Breazu-Tannen, "Combining Algebra and Higher-Order Types", Proceedings of 3rd IEEE Symp. on Logic in Computer Science, Edinburgh, 1988.

2. N. Dershowitz-M. Okada, "Proof-theoretic Techniques for Theory of Rewriting", Proceedings of 3rd IEEE Symp. on Logic in Computer Science, Edinburgh, 1988.

3. N. Dershowitz-M. Okada, "Conditional Equational Programming and Conditional Rewriting", Proc. of International Conference on Fifth Generation Computer Systems, ICOT, Tokyo, 1988.

4. J. Y. Girard, "Une extension de l'interpretatin de Godel a l'analyse, et son application a l'elimination des coupures dans l'analyse et la theorie des types", in Proceedings of the Second Scandinavian Logic Symposium, ed. Fenstad, 1971, pp. 63-92.

5. G. Huet, D. C. Oppen, Equations and Rewrite Rules, in Formal Language Theory; A Survey, ed. Book, 1980.

6. J. W. Klop, Combinatory Reduction Systems, Tract 129, Math Center, Amsterdam, 1980.

7. P. Martin-Lof, "Hauptsatz for the Theory of Species", in Proceedings of the Second Scandinavian Logic Symposium, ed. Fenstad, 1971, pp. 217-233.

8. M. Okada, Introduction to Proof Theory for Computer Science, Lecture Notes, Laboratoire de Recherche en Informatique, Universite de Paris XI, Orsay, France, 1988.

9. D. Prawitz, "Ideas and Results in Proof Theory", in Proceedings of the Second Scandinavian Logic Symposium, ed. Fenstad, 1971, pp. 235-307.

10. Y. Toyama, "On the Church-Rossor Property for the Direct Sum of Term Rewriting Systems", Journal of the ACM 34 (1987), 129-143.