NOTICE WARNING CONCERNING COPYRIGHT RESTRICTIONS:
The copyright law of the United States (title 17, U.S. Code) governs the making of photocopies or other reproductions of copyrighted material. Any copying of this document without permission of its author may be prohibited by law.

On Weak and Strong Normalisations

by

Hongwei Xi

Department of Mathematics
Carnegie Mellon University
Pittsburgh, PA 15213

Research Report No. 96-187
February, 1996

510.6
C28R
96-187

University Libraries
Carnegie Mellon University
Pittsburgh P

On Weak and Strong Normalisations

Hongwei Xi

Mathematics Department
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213

Telephone: (412)268-1439 Fax: (412)268-6380

Email: hwxi@cs.cmu.edu

February 2, 1996

University Libraries
Carnegie Mellon University
Pittsburgh PA 15213-3890

Abstract

With the help of continuations, we first construct a transformation $\mathcal{T}$ which transforms every λ-term t into a λI-term $\mathcal{T}(t)$. Then we apply the conservation theorem in λ-calculus to show that t is strongly normalisable if $\mathcal{T}(t)$ has a β-normal form. In this way, we succeed in establishing the equivalence between weak and strong normalisation theorems in various typed λ-calculi. This not only enhances the understanding between weak and strong normalisations, but also presents an elegant approach to proving strong normalisation theorems via the notion of weak normalisations.

1. Introduction

In λ-calculus or some other rewriting systems, a term is said to be weakly normalisable if it can be reduced to a normal form in *some* way while a term is strongly normalisable if every reduction sequence from it terminates with a normal form. Clearly, there exist weakly normalisable terms in λ-calculus which are not strongly normalisable. For instance, $(\lambda x.\lambda y.y)\Omega$ for $\Omega = (\lambda x.xx)(\lambda x.xx)$ has a normal form $\lambda y.y$, but it is not strongly normalisable. On the hand, the conservation theorem (Church, 1941) states that every λI-term is weakly normalisable if and only if it is strongly normalisable, namely, there is no distinction between weak and strong normalisations in λI-calculus. This comparison suggests the following approach to handling strong normalisations.

Given a system in which the weak normalisation theorem holds, i.e., every term t in the system is weakly normalisable, we try to construct a transformation $t \mapsto \mathcal{T}(t)$ such that $\mathcal{T}(t)$ is a λI-term in the system for every t and any infinite reduction sequence from t would induce an infinite reduction sequence from $\mathcal{T}(t)$. By the conservation theorem, $\mathcal{T}(t)$ is strongly normalisable since it is a weakly normalisable λI-term. This shows that t is also strongly normalisable. Therefore, the system must enjoy the strongly normalisation theorem if we can succeed in constructing such a transformation $\mathcal{T}$, which will be achieved with the help of continuations.

The structure of the paper is given as follows.

- The notions and basics are explained in Section 2.
- In Section 3, we construct a transformation $\mathcal{T}$ using continuations, and prove that $\mathcal{T}$ has the properties aforementioned.
- In Section 4, we show how to adapt $\mathcal{T}$ into some typed λ-calculi, and therefore, prove the equivalence between weak and strong normalisations theorems in these systems.
- Some closely related work is mentioned in Section 5.
- A formulation of simply typed λ-calculus $\lambda^{\rightarrow}$ and system **F**, with some adjustments made to suit our purpose, is given in Appendix A.

2. Notions, Terminology and Basics

We give a brief explanation on the notions and terminology used in this paper. Most details, which could not be included here, can be found in Barendregt (1984) and Girard (1989).

Definition 1 *(Λ-terms and Λ_I-terms) 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 $t \in \Lambda$ then $(\lambda x.t) \in \Lambda$; u is a subterm of $(\lambda x.t)$ if u is $\lambda x.t$ or a subterm of t.*
- *(application) If $t_0, t_1 \in \Lambda$ then $(t_0 t_1) \in \Lambda$; u is a subterm of $(t_0 t_1)$ if u is $(t_0 t_1)$ or a subterm of t_i for some $i \in \{0, 1\}$.*

The set $\mathrm{FV}(t)$ *of free variables in* t *is defined as follows.*

$$\mathrm{FV}(t) = \begin{cases} \{x\} & \text{if } t = x \text{ for some variable } x; \\ \mathrm{FV}(t_0) - \{x\} & \text{if } t = \lambda x.t_0; \\ \mathrm{FV}(t_0) \cup \mathrm{FV}(t_1) & \text{if } t = t_0 t_1. \end{cases}$$

The set Λ_I of λI-terms is a subset of Λ such that, for every term $t \in \Lambda_I$, if $(\lambda x.t_0)$ is a subterm of t then $x \in \mathrm{FV}(t_0)$.

$[u/x]v$ stands for substituting u for all free occurrences of x in v. α-conversion or renaming bounded variables may have to be performed in order to avoid name collisions. Also certain substitution properties are assumed in the following proofs.

Definition 2 *(β-redex, β-reduction and β-normal form) A term of form $(\lambda x.u)v$ is called a β-redex, and $[v/x]u$ is called the contractum of the redex; $t \rightsquigarrow t'$ stands for a β-reduction step where t' is obtained from replacing some redex in t with its contractum; a β-normal form is a term in which there is no β-redex.*

"β-" is often omitted if this causes no confusion. Usually there are many different redexes in a term t; a redex r_1 in t is left to another redex r_2 in t if the first λ in r_1 is left to the first λ in r_2. A leftmost reduction is a reduction step in which the leftmost redex gets contracted.

$\rightsquigarrow^n$, $\rightsquigarrow^+$ and $\rightsquigarrow^*$ stand for reduction sequences consisting of n steps, a positive number of steps and a nonnegative number of steps, respectively.

Definition 3 *A λ-term t is weakly normalisable if there exists a reduction sequence $t \rightsquigarrow^* u$ such that u is in β-normal form; t is strongly normalisable if all reduction sequence from t are finite.*

By König's Lemma, there exists a longest reduction sequence $\rightsquigarrow^n$ from t if t is strongly normalisable. Let $\mu(t)$ denote the n. The following lemma or its similar version can be easily found elsewhere such as Barendregt (1984). We present a detailed proof here since this is not only crucial to the proof of our main theorem, but also motivates the construction of following transformation $\mathcal{T}$.

Lemma 4 *Given $t = rt_1 \ldots t_n$, where $r = (\lambda x.u)v$ is a redex and v is strongly normalisable; if $t^* = ([v/x]u)t_1 \ldots t_n$ is strongly normalisable then t is strongly normalisable.*

Proof Suppose that there is an infinite reduction sequence σ starting from t. Then we have two cases.

- σ contains no leftmost reductions. Since v and $t^* = ([v/x]u)t_1 \ldots t_n$ are strongly normalisable, no infinite reduction sequence can start from v or any t_i for $i = 1, \ldots n$. Therefore, there exists an infinite reduction sequence from u. This implies that there also exists an infinite reduction sequence from $[v/x]u$, contradicting t^* being strongly normalisable.
- σ contains at least one leftmost reduction. Then σ is of the following form

$$t \rightsquigarrow^* t' = (\lambda x.u')v't'_1 \ldots t'_n \rightsquigarrow ([v'/x]u')t'_1 \ldots t'_n \rightsquigarrow \cdots,$$

where $u \rightsquigarrow^* u'$, $v \rightsquigarrow^* v'$ and $t_i \rightsquigarrow^* t'_i$ for $i = 1, \ldots n$. Clearly, we can reduce $t^* = ([v/x]u)t_1 \ldots t_n$ to $([v'/x]u')t'_1 \ldots t'_n$ in an obvious way. This contradicts t^* being strongly normalisable.

Therefore, there is no infinite reduction sequence starting from t, namely, t is strongly normalisable. ∎

Now let us state the conservation theorem for λI-calculus, on which the entire work is established.

Theorem 5 *(Church, 1941) A λI-term is weakly normalisable if and only if it is strongly normalisable.*

Proof Please refer to Theorem 11.3.4 in Barendregt (1984). ∎

In the next section, we will use continuations to construct our transformation. Fisher (1993), Plotkin (1975) and Reynolds(1972) provide a detailed cover on this subject.

3. Transformation

Our goal is to construct a transformation $\mathcal{T}$ from Λ into Λ_I such that t is strongly normalisable if and only if $\mathcal{T}(t)$ is strongly normalisable. Given $t = (\lambda x.u)vt_1 \ldots t_n$, we know, by Lemma 4, t is strongly normalisable if and only if both $t' = ([v/x]u)t_1 \ldots t_n$ and v are strongly normalisable. Hence, we need some kind of reduction which not only β-reduces the leftmost redex $(\lambda x.u)v$ but also verifies if v is strongly normalisable. With the help of continuations, this kind of reduction can be easily simulated by β-reduction. To simplify our presentation, we assume that there exists a constant $\langle , \rangle$ in Λ, and $\langle t, \rangle$ and $\langle t_0, t_1 \rangle$ denote $\langle , \rangle t$ and $\langle , \rangle t_0 t_1$, respectively. Constant $\langle , \rangle$ can always be replaced with a distinct variable without affecting the strong normalisability of terms. For those who know Plotkin's call-by-name continuation passing style transformation, the following $\mathcal{T}$ is a variation of it but has an ability to verify if the arguments of functions are strongly normalisable.

$$\begin{array}{ll} \mathcal{T}(x) = x & \mathcal{T}(k; x) = xk \\ \mathcal{T}(\lambda x.t) = \lambda k_0.k_0(\lambda x.\lambda k_1.\langle \mathcal{T}(t)k_1, x \rangle) & \mathcal{T}(k; \lambda x.t) = k(\lambda x.\lambda k_1.\langle \mathcal{T}(t)k_1, x \rangle) \\ \mathcal{T}(t_0 t_1) = \lambda k_0.\mathcal{T}(t_0)(\lambda k_1.k_1 \mathcal{T}(t_1)k_0) & \mathcal{T}(k; t_0 t_1) = \mathcal{T}(t_0)(\lambda k_1.k_1 \mathcal{T}(t_1)k) \end{array}$$

If $\langle , \rangle$ is replaced with $K = \lambda x.\lambda y.x$, then $\mathcal{T}$ is β-equivalent to Plotkin's call-by-name transformation. The sole purpose of $\langle , \rangle$ is to verify the strong normalisability of v when $(\lambda x.u)v$ is contracted.

Proposition 6 *$\mathcal{T}$ has the following properties.*

1. *$\mathcal{T}(t)$ is a λI-term for any λ-term t.*

2. *For any proper subterm t_s of t, $\mathcal{T}(t_s)$ is a subterm of $\mathcal{T}(t)$.*

3. *$\mathcal{T}(t)k \rightsquigarrow^* \mathcal{T}(k;t)$; $\mu(\mathcal{T}(t)) \leq 1 + \mu(\mathcal{T}(k;t))$ if $\mathcal{T}(k;t)$ is strongly normalisable.*

4. *$\mathcal{T}([t_1/x]t) = [\mathcal{T}(t_1)/x]\mathcal{T}(t)$.*

5. *Given $r = (\lambda x.u)v$, $t = rt_1 \ldots t_n$, and $t' = ([v/x]u)t_1 \ldots t_n$; then for some $p > 1$,*

$$\mathcal{T}(t) \rightsquigarrow^p \lambda k_0.\langle \mathcal{T}(k_0;t'), \mathcal{T}(v)\rangle.$$

Proof We give proofs accordingly.

1. This simply follows from the definition.

2. A straightforward structural induction on t yields the result.

3. This simply follows from the definition.

4. A straightforward structural induction on t yields the result.

5. Let us proceed by induction on n.

   - $n = 0$.

$$\begin{array}{rcl}
\mathcal{T}(t) & = & \lambda k_0.(\lambda k_0'.k_0'(\lambda x.\lambda k_1'.\langle \mathcal{T}(u)k_1', x\rangle))(\lambda k_1.k_1\mathcal{T}(v)k_0) \\
& \rightsquigarrow & \lambda k_0.(\lambda k_1.k_1\mathcal{T}(v)k_0)(\lambda x.\lambda k_1'.\langle \mathcal{T}(u)k_1', x\rangle) \\
& \rightsquigarrow & \lambda k_0.(\lambda x.\lambda k_1'.\langle \mathcal{T}(u)k_1', x\rangle)\mathcal{T}(v)k_0 \\
& \rightsquigarrow & \lambda k_0.(\lambda k_1'.\langle([\mathcal{T}(v)/x]\mathcal{T}(u))k_1', \mathcal{T}(v)\rangle)k_0 \\
& \rightsquigarrow & \lambda k_0.\langle([\mathcal{T}(v)/x]\mathcal{T}(u))k_0, \mathcal{T}(v)\rangle \\
& \rightsquigarrow & \lambda k_0.\langle \mathcal{T}([v/x]u)k_0, \mathcal{T}(v)\rangle, \text{ by (4)} \\
& = & \lambda k_0.\langle \mathcal{T}(t')k_0, \mathcal{T}(v)\rangle \rightsquigarrow^* \lambda k_0.\langle \mathcal{T}(k_0;t'), \mathcal{T}(v)\rangle, \text{ by (2)}
\end{array}$$

   - $n > 0$. Let $t_h = rt_1 \ldots t_{n-1}$ and $t_h' = ([v/x]u)t_1 \ldots t_{n-1}$. Hence,

$$\mathcal{T}(t_h) \rightsquigarrow^+ \lambda k.\langle \mathcal{T}(t_h')k, \mathcal{T}(v)\rangle,$$

     by induction hypothesis. Note

$$\begin{array}{rcl}
\mathcal{T}(t) & = & \lambda k_0.\mathcal{T}(t_h)(\lambda k_1.k_1\mathcal{T}(t_n)k_0) \\
& \rightsquigarrow^+ & \lambda k_0.(\lambda k.\langle \mathcal{T}(t_h')k, \mathcal{T}(v)\rangle)(\lambda k_1.k_1\mathcal{T}(t_n)k_0) \\
& \rightsquigarrow & \lambda k_0.\langle \mathcal{T}(t_h')(\lambda k_1.k_1\mathcal{T}(t_n)k_0), \mathcal{T}(v)\rangle \\
& = & \lambda k_0.\langle \mathcal{T}(k_0;t_h't_n), \mathcal{T}(v)\rangle = \lambda k_0.\langle(\mathcal{T}(k_0;t'), \mathcal{T}(v)\rangle
\end{array}$$

■

Theorem 7 *(Main theorem) Given any λ-term t; if $\mathcal{T}(t)$ has a normal form then t is strongly normalisable.*

Proof Assume that $\mathcal{T}(t)$ has a normal form. By the conservation theorem, $\mathcal{T}(t)$ is strongly normalisable since $\mathcal{T}(t)$ is a λI-term. Now let us proceed by lexical induction on $\mu(\mathcal{T}(t))$, the number of steps in a longest reduction sequence from $\mathcal{T}(t)$, and the structure of $\mathcal{T}(t)$.

- $t = \lambda x.t_0$. By induction hypothesis, t_0 is strongly normalisable since $\mathcal{T}(t_0)$ is a subterm of $\mathcal{T}(t)$. This implies that t is strongly normalisable.

- $t = xt_1 \ldots t_n$ for some variable x. By induction hypothesis, for $i = 1, \ldots n$, t_i are strongly normalisable since $\mathcal{T}(t_i)$ are subterms of $\mathcal{T}(t)$. This yields that t is strongly normalisable.
- $t = rt_1 \ldots t_n$, where $r = (\lambda x.u)v$. Since $\mathcal{T}(v)$ is a subterm of $\mathcal{T}(t)$ by Proposition 6 (2), v is strongly normalisable. Note $t \rightsquigarrow t' = ([v/x]u)t_1 \ldots t_n$ and $\mathcal{T}(t) \rightsquigarrow^p \lambda k.\langle \mathcal{T}(k,t'), \mathcal{T}(v)\rangle$ for some $p > 1$ by Proposition 6 (5). Hence, $1 + \mu(\mathcal{T}(k,t')) < \mu(\mathcal{T}(t))$. This implies $\mu(\mathcal{T}(t')) \leq 1 + \mu(\mathcal{T}(k,t')) < \mu(\mathcal{T}(t))$ by Proposition 6 (3). By induction hypothesis, t' is strongly normalisable. Therefore, t is strongly normalisable by Lemma 4. ■

Given a system closed under transformation $\mathcal{T}$, i.e., for every term t in the system, $\mathcal{T}(t)$ is also a term in it. If all the terms in the system are weakly normalisable, then they are strongly normalisable by the main theorem. This motivates the following applications to typed λ-calculi.

4. Applications

Now we are ready to show that some typed λ-calculi are closed under transformation $\mathcal{T}$. This amounts to showing how to type term $\mathcal{T}(t)$ when given a typed term t in these systems. Since this has been thoroughly studied by Meyer and Wand (1985), Griffin (1990), Harper and Lillibridge (1993a, 1993b), we will simply mention the results. A formulation of simply typed λ-calculus $\lambda^{\rightarrow}$ and system **F** can be found in Appendix A, where some adjustments have been made to suit our purpose.

Since type abstraction and application in **F** are treated the same as abstraction and application on terms, we have to extend the definition of $\mathcal{T}$ to handle types so that all terms in **F** can be transformed correctly.

Definition 8 *Type transformation* $|\cdot|$ *and* $\mathcal{T}$ *are defined as follows.*

$$\mathcal{T}(\alpha) = \begin{cases} \alpha & \text{if } \alpha \text{ is an atomic type or a type variable;} \\ |\alpha_0| \rightarrow |\alpha_0| & \text{if } \alpha = \alpha_0 \rightarrow \alpha_1; \\ \forall X.|\alpha_0| & \text{if } \alpha = \forall X.\alpha_0. \end{cases}$$

and

$$|\alpha| = (\mathcal{T}(\alpha) \rightarrow \mathbf{unit}) \rightarrow \mathbf{unit}$$

Notice that $|\cdot|$ is also called Kolmogorov's double negation embedding, which is used to translate classical logic into intuitionistic logic. Clearly, Theorem 7 still holds in this new setting. For a context Γ of form $x_1 : \alpha_1, \ldots, x_n : \alpha_n$, $\mathcal{T}(\Gamma)$ is of form $x_1 : |\alpha_1|, \ldots, x_n : |\alpha_n|$.

Theorem 9 *If* $\Gamma \vdash t : \alpha$ *is derivable in* $\lambda^{\rightarrow}$, *then* $\mathcal{T}(\Gamma) \vdash \mathcal{T}(t) : |\alpha|$ *is also derivable in* $\lambda^{\rightarrow}$.

Proof This is due to Meyer and Wand (1985). Under propositions-as-types interpretation, the proof corresponds exactly to Kolmogorov's double negation embedding for propositional logic. ■

Corollary 10 *If all terms in* $\Lambda^{\rightarrow}$ *are weakly normalisable, then they are strongly normalisable.*

Proof Given $t \in \Lambda^{\rightarrow}$, we know $\mathcal{T}(t) \in \Lambda^{\rightarrow}$ by Theorem 9. Hence $\mathcal{T}(t)$ is weakly normalisable. By Theorem 7, t is strongly normalisable. ∎

It can be readily shown that all $\Lambda^{\rightarrow}$ are weakly normalisable by a well-known method originally due to Turing according to Gandy (1980), which can also be found in Andrews (1971) and Girard (1989). Therefore, the simply typed λ-calculus enjoys strong normalisation theorem.

Theorem 11 *If* $\Gamma \vdash t : \alpha$ *is derivable in* $\mathbf{F}$, *then* $\mathcal{T}(\Gamma) \vdash \mathcal{T}(t) : |\alpha|$ *is also derivable in* $\mathbf{F}$.

Proof The proof corresponds exactly to Kolmogorov's double negation embedding for second-order propositional logic. ∎

Corollary 12 *If all terms in* $\mathbf{F}$ *are weakly normalisable then they are strongly normalisable.*

Proof Parallel to the proof of Corollary 10 ∎

It is first shown by Girard (1971) that all terms in $\mathbf{F}$ are weakly normalisable. Though it is proven later that all terms in $\mathbf{F}$ are strongly normalisable by Prawitz (1971), the formulation of reducibility candidates gets complicated accordingly. This comparison can also be done with a sharp weak normalisation proof for the theory of species (Martin-Löf, 1971), where the concise and perspicuous formulation of reducibility candidates certainly enhances the understanding. With Corollary 12, we can keep the merits in proofs for weak normalisation theorems while establishing the results for strong normalisation.

For those know system $\mathbf{F}_{\omega}$, it can be verified that system $\mathbf{F}_{\omega}$ is also closed under transformation $\mathcal{T}$. This simply follows from the typing properties of CPS conversion for $\mathbf{F}_{\omega}$ (Harper and Lillibridge, 1993a). Hence, the weak normalisation theorem in $\mathbf{F}_{\omega}$ implies the strong normalisation theorem in it.

5. Related work and Conclusion

Ideas of transforming strong normalisation into weak normalisation can also be found in Nederpelt (1973), Klop (1980), de Groot (1993) and Kfoury and Wells (1994). To some extent, our transformation $\mathcal{T}$ is related to the *controlling erasure* in the literature, which, in addition to contracting β_I-redex, only reduces $(\lambda x.\lambda y.u)v$ to $\lambda y.(\lambda x.u)v$ when u contain no free occurrences of y. In this way, the conservation theorem for λK-calculus can be called to establish results for strong normalisation upon the corresponding ones for weak normalisation.

The typing properties of continuation-passing-style transformation were first studied by Meyer and Wand (1985), and extended to polymorphic type systems by Harper and Lillibridge (1993a, 1993b).

The formulation of transformation $\mathcal{T}$ makes use of continuations, which not only avoids introducing other uncommon reductions such as the ones used by de Groot (1993) and Kfoury and Wells (1994), but also reveals an intimate relation between types and normalisations. With some known results on typing properties in various typed λ-calculi, we can readily claim that these systems are closed under transformation $\mathcal{T}$. Therefore, it becomes unnecessary to complicate methods for

the purpose of proving strong normalisation theorems in these systems since weak normalisation theorems are equally strong. This is particularly helpful if such theorems are to be proven with the help syntactical methods such as the one used to show all terms in $\lambda^{\rightarrow}$ are weakly normalisable.

References

Andrews, P.B., (1971), Resolution in type theory, *J. Symbolic Logic 36*, pp. 414-432.

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

Church, A., (1941), The calculi of lambda conversion, *Princeton University Press, Princeton.*

de Groote, P., (1993), The conservation theorem revisited, *Int'l conf. Typed lambda calculi and applications*, vol. 664 of LNCS, pp. 163-178.

M.J. Fisher, Lambda-calculus schemata, *LISP and Symbolic Computation*, vol. 6, 3(4), pp. 259-288.

Gandy, R.O., (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.

Girard, J.-Y., (1971) Une extension de L'interpretation de Gödel à l'analyse, et son application à l'élimination des coupures dans l'analyse et la théorie des types, Proceedings of the 2nd Scandinavian logic symposium, *editor J.E. Fenstad, North-Holland Publishing Company, Amsterdam.*

Girard, J.-Y., LaFont, Y. and Taylor, P., (1989), Proofs and types, vol 7 of Cambridge Tracts in Theoretical Computer Science, *Cambridge Press, Cambridge University*, 176 pp.

Griffin, T., (1990) A formulae-as-types notion of control, *Seventh ACM Symposium on Principles of Programming Languages.*

Harper, R. and Lillibridge M., (1993a), Explicit polymorphism and CPS conversion, *Twentieth ACM Symposium on Principles of Programming Languages*, pp. 206-209

Harper, R. and Lillibridge M., (1993b), Polymorphic type assignment and CPS conversion, *LISP and Symbolic Computation*, vol. 6, 3(4), pp. 361-380.

Klop, J.W., (1980), Combinatory reduction systems, *Ph.D. thesis, CWI, Amsterdam, Mathematical center tracts, No. 127.*

Kfoury, A.J. and Wells, J.B., (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.*

Martin-Löf, P., (1971), Hauptsatz for the theory of species, Proceedings of the 2nd Scandinavian logic symposium, *editor J.E. Fenstad, North-Holland Publishing Company, Amsterdam.*

Meyer, A. and Wand M., (1985), Continuation semantics in typed lambda calculi (summary). *editor, Rohit Parikh, Logics of Programs, Lecture Notes in Computer Science, vol. 193*, pp 219-224.

Nederpelt, R.P., (1973), Strong normalisation in a typed lambda calculus with lambda structured types, *Ph.D. thesis, Technische Hogeschool Eindhoven.*

Plotkin, G., (1975), Call-by-name, call-by-value, and the lambda calculus, *Theoretical Computer Science*, vol 1, pp. 125-159.

Prawitz, D., (1971), Ideas and results of proof theory, Proceedings of the 2nd Scandinavian logic symposium, *editor J.E. Fenstad, North-Holland Publishing Company, Amsterdam.*

Reynolds, J., (1972), Definitional interpreters for higher-order programming languages, *Conference record of the 25th national ACM conference*, page 717-740. Reynolds, J., (1974), Towards a theory of type structure, *Colloquium sur la Progrmmation*, vol. 19 of LNCS, pp. 408-423.

A Formulation of $\lambda^{\rightarrow}$ and F

A judgement of form $\vdash \alpha : \mathbf{type}$ states that α is type; a context Γ is of form $x_1 : \alpha_1, \ldots, x_n : \alpha_n$, where $x_1, \ldots, x_n$ are distinct variables; a judgement of form $\Gamma \vdash t : \alpha$ states that t is of type α under context Γ.

Definition 13 *($\lambda^{\rightarrow}$-calculus) The rules for formulating simple types are given below.* **unit** *is simply used to facilitate the presentation, and it can be replaced by any other atomic type, or by $\forall X.X$ in the polymorphic setting.*

- *(Formulation of atomic type) If α is* **unit** *or other given atomic types,*

$$\frac{}{\vdash \alpha : \mathbf{type}}$$

- *(Formulation of function type)*

$$\frac{\vdash \alpha : \mathbf{type} \quad \vdash \beta : \mathbf{type}}{\vdash \alpha \rightarrow \beta : \mathbf{type}}$$

- *(Variable)*

$$\frac{\vdash \alpha : \mathbf{type}}{\Gamma, x : \alpha \vdash x : \alpha}$$

- *(Constant)*

$$\frac{\vdash \mathbf{unit} \rightarrow (\alpha \rightarrow \mathbf{unit}) : \mathbf{type}}{\Gamma \vdash \langle , \rangle : \mathbf{unit} \rightarrow (\alpha \rightarrow \mathbf{unit})}$$

- *(Abstraction)*

$$\frac{\Gamma, x : \alpha \vdash t : \beta}{\Gamma \vdash \lambda x.t : \alpha \rightarrow \beta}$$

- *(Application)*

$$\frac{\Gamma \vdash t_0 : \alpha \rightarrow \beta \quad \Gamma \vdash t_1 : \alpha}{\Gamma \vdash t_0 t_1 : \beta}$$

Let $\Lambda^{\rightarrow}$ be the set of all terms t such that $\Gamma \vdash t : \alpha$ is derivable in $\lambda^{\rightarrow}$ for some Γ and α.

Definition 14 *(System* **F***) In addition to the rules for simple types, the following ones are needed to formulate polymorphic types.*

- *(Formulation of type variable) If X is a given type variable,*

$$\frac{}{\vdash X : \mathbf{type}}$$

- *(Formulation of polymorphic type)*

$$\frac{\vdash \alpha}{\vdash \forall X.\alpha : \mathbf{type}}$$

In addition to the rules for terms in $\lambda^{\rightarrow}$, the following ones are needed to formulate terms in **F**.

- *(Constant)*

$$\frac{\vdash \mathbf{unit} \rightarrow \forall X.\mathbf{unit} : \mathbf{type}}{\Gamma \vdash \langle , \rangle : \mathbf{unit} \rightarrow \forall X.\mathbf{unit}}$$

- *(Quantification)*

$$\frac{\Gamma \vdash t : \alpha}{\Gamma \vdash \lambda X.t : \forall X.\alpha}\ ,$$

  where no types in Γ contain a free occurrence of X.

- *(Instantiation)*

$$\frac{\Gamma \vdash t : \forall X.\alpha \qquad \vdash \beta : \mathbf{type}}{\Gamma \vdash t\beta : [\beta/X]\alpha}\ ,$$

  where $[\beta/X]\alpha$ is obtained from substituting β for every occurrence of X in α.

t is a term in **F** *if $\Gamma \vdash t : \alpha$ is derivable in* **F** *for some Γ and α.*

APR 1 5 2004

Carnegie Mellon University Libraries

3 8482 01379 3407