How to Normalize the Jay

Dieter Probst * Thomas Studer *

Abstract

In this note we give an elementary proof of the strong normalization property of the J combinator by providing an explicit bound for the maximal length of the reduction paths of a term. This result shows nicely that in the theorem of Toyama, Klop and Barendregt on completeness of unions of left linear term rewriting systems, disjointness is essential.

Keywords: Term rewriting systems; Combinatory logic; Strong normalization

1 Introduction

The combinators I and J with their reduction rules $\mathsf{I}a \to a$ and $\mathsf{J}abcd \to ab(adc)$ were introduced by Rosser [2] in 1935. These two combinators are of particular interest since they form a basis for the $\lambda\mathsf{I}$-calculus (cf. e.g. Barendregt [1]).

In combinatory logic, it is natural to ask whether a certain system is strongly normalizing, i.e. whether there exists no term with an infinite reduction path. Many standard combinators such as $\mathsf{K}, \mathsf{B}, \mathsf{C}$ and I are strongly normalizing, with the notable exception of S. But surprisingly, it appears to be unknown whether the reduction system generated by the combinator J is strongly normalizing.

In this note, we prove the strong normalization property of the J combinator by providing an explicit bound for the maximal length of the reduction paths of a term. Or, in the words of Smullyan [3], we show that binary trees with jaybirds sitting on their leaves strongly normalize.

*Research supported by the Swiss National Science Foundation.

2 Notation

Let $\mathsf{L_J}$ denote the language containing countably many variables $x_1, x_2, \dots,$ the constant symbol J and the binary function symbol $\cdot$ (application). As usual, the constant J and every variable are $\mathsf{L_J}$-terms, and if s and t are $\mathsf{L_J}$-terms then also $(s \cdot t)$. We write st for $(s \cdot t)$ and adopt the convention of association to the left, i.e. $s_1 \dots s_n$ stands for $(\dots(s_1 s_2)\dots s_n)$. By $\mathcal{C}_\mathsf{J}$ we denote the set of all $\mathsf{L_J}$-terms, and by $\mathcal{C}^0_\mathsf{J}$ the set of all closed $\mathsf{L_J}$-terms, i.e. the $\mathsf{L_J}$-terms which contain no variables.

Definition 1. $\rightarrow\ \subseteq \mathcal{C}_\mathsf{J} \times \mathcal{C}_\mathsf{J}$ is the smallest relation satisfying

(1) $\mathsf{J}abcd \rightarrow ab(adc)$ for all $\mathsf{L_J}$-terms a, b, c, d.

(2) If s, s', t are $\mathsf{L_J}$-terms and $s \rightarrow s'$, then also $st \rightarrow s't$ and $ts \rightarrow ts'$.

If $s \rightarrow t$ holds for two terms s and t, we say that s *reduces to* t.

Definition 2. An *infinite reduction path* is an infinite sequence of $\mathsf{L_J}$-terms $(t)_{n\in\mathbb{N}}$ such that $t_n \rightarrow t_{n+1}$ for all $n \in \mathbb{N}$.

Definition 3. An $\mathsf{L_J}$-term t is *strongly normalizable*, if there is no infinite reduction path starting with t. An $\mathsf{L_J}$-term t is said to be in *normal form*, if there is no $\mathsf{L_J}$-term t' such that $t \rightarrow t'$.

Whenever $s \rightarrow t$ holds, there is a subterm s' of s of the form $\mathsf{J}abcd$ which reduces to a subterm $ab(adc)$ of t. The following definition gives us a tool to indicate the particular occurrence of the subterm s' which gets reduced.

Definition 4. Let $\mathcal{W}$ be the set of all finite words over the alphabet $\{l, r\}$. The empty word is denoted by ϵ. For every $w \in \mathcal{W}$ we define a function $f_w : \mathcal{C}^0_\mathsf{J} \rightarrow \mathcal{C}^0_\mathsf{J} \cup \{\bot\}$ where $\bot \notin \mathcal{C}^0_\mathsf{J}$ by

$$\begin{array}{lcl} f_\epsilon(t) & := & t, \\ f_{lw}(\mathsf{J}) & := & \bot, \\ f_{rw}(\mathsf{J}) & := & \bot, \\ f_{lw}(st) & := & f_w(s), \\ f_{rw}(st) & := & f_w(t). \end{array}$$

In the sequel we often write $(t)_w$ for $f_w(t)$.

Definition 5. An $\mathsf{L_J}$-term of the form $\mathsf{J}abcd$ is called a *redex with contractum* $ab(adc)$. If $w \in \mathcal{W}$, we write $t \rightarrow_w t'$ if there is a redex r with contractum r' such that $(t)_w \equiv r$ and $(t')_w \equiv r'$.

The following lemmas are trivial consequences of Definition 1.

Lemma 6. *Every $t \in \mathcal{C}_{\mathsf{J}}$ is strongly normalizable if and only if every $t \in \mathcal{C}^0_{\mathsf{J}}$ is strongly normalizable.*

Lemma 7. *For every $r \in \mathcal{C}^0_{\mathsf{J}}$ we have: if there exist $r' \in \mathcal{C}^0_{\mathsf{J}}$ and $w \in \mathcal{W}$ so that $r \rightarrow_w r'$ and $w \neq \epsilon$, then there are $s, t, s', t' \in \mathcal{C}^0_{\mathsf{J}}$ and $w' \in \mathcal{W}$ such that $r \equiv st$ and either*

(1) $w = lw'$ and $s \rightarrow_{w'} s'$ and $r' \equiv s't$, or

(2) $w = rw'$ and $t \rightarrow_{w'} t'$ and $r' \equiv st'$.

3 Normalization

The next definition is the crucial step in our normalization proof. We introduce a weighting function $|\cdot|$ which assigns to every L_{J}-term an upper bound for the maximal length of its reduction paths.

Definition 8. We define $|\cdot| : \mathcal{C}^0_{\mathsf{J}} \rightarrow \mathbb{N}$ recursively by the following clauses:

$$|r| := \begin{cases} 1, & \text{if } r \equiv J, \\ |t| + 2^{|(s)_l|} + |s|, & \text{if } r \equiv st \text{ and } s \not\equiv J, \\ |t| + |s|, & \text{if } r \equiv st \text{ and } s \equiv J. \end{cases}$$

Observe that this function does not satisfy the replacement property, meaning we find terms s, s', t in $\mathcal{C}^0_{\mathsf{J}}$ so that both $|s| > |s'|$ and $|st| < |s't|$ hold. For example, choose $s \equiv \mathsf{J}(\mathsf{J}(\mathsf{J}(\mathsf{J}(\mathsf{JJ}))))$, $s' \equiv \mathsf{JJJ}$ and $t \equiv \mathsf{J}$. Then we obviously have $|s| = 6 > 5 = |s'|$ but also $|st| = 9 < 10 = |s't|$. The reason is that $|(s)_l| = 1 < 2 = |(s')_l|$. Please note that $s \rightarrow s'$ does not hold in this example, cf. Theorem 10.

Lemma 9. *For all $a, b, c, d \in \mathcal{C}^0_{\mathsf{J}}$ we have:*

(1) $|\mathsf{J}abcd| > |ab(adc)|$.

(2) $|\mathsf{J}abc| = |(\mathsf{J}abcd)_l| > |(ab(adc))_l| = |ab|$.

Proof. Let $a \not\equiv \mathsf{J}$. A straightforward calculation yields

$$|Jabcd| = |d| + 2^{|b|+2^1+|a|+1} + |c| + 2^{|a|+1} + |b| + 2^1 + |a| + 1$$

and

$$|ab(adc)| = |c| + 2^{|a|} + |d| + 2^{|(a)_l|} + |a| + 2^{|a|} + |b| + 2^{|(a)_l|} + |a|.$$

Because of $|(a)_l| < |a|$ and $n < 2^n$ ($\forall n \in \mathbb{N}$) we have

$$2 \cdot 2^{|a|} + 2 \cdot 2^{|(a)_l|} + |a| < 2^{|a|+1} + 2^{|a|} + 2^{|a|} \leq 2^{|a|+2},$$

therefore Claim (1) is verified. (2) clearly holds since

$$|Jabc| = |c| + 2^{|a|+1} + |b| + 2^1 + |a| + 1,$$

and

$$|ab| = |b| + 2^{|(a)_l|} + |a|.$$

In the case $a \equiv \mathsf{J}$ the expressions $2^{|(a)_l|}$ do not appear in the above arguments and both claims hold as well. □

Theorem 10. *For every closed $\mathsf{L_J}$-term r we have: if there is a closed $\mathsf{L_J}$-term r' with $r \to r'$, then $|r| > |r'|$ and also $|(r)_l| \geq |(r')_l|$.*

Proof. First, we note that since r contains a redex, r must be of the form st. Therefore we have $(r)_l \neq \perp$. We prove the theorem by induction on the definition of closed $\mathsf{L_J}$-terms. Consider the closed $\mathsf{L_J}$-term $r \equiv st$ and suppose that the claim holds for s and t. If $r \to_\epsilon r'$ the claim follows by Lemma 9. Otherwise, by Lemma 7, there exist $w \in \mathcal{W}$ and $s', t' \in \mathcal{C}^0_{\mathsf{J}}$ so that either

(1) $r \to_{lw} r'$ and $s \to_w s'$ and $r' \equiv s't$, or

(2) $r \to_{rw} r'$ and $t \to_w t'$ and $r' \equiv st'$.

Assume we are in the first case. By the induction hypothesis we get $|s| > |s'|$ and $|(s)_l| \geq |(s')_l|$, so that

$$|r| = |st| = |t| + 2^{|(s)_l|} + |s| > |t| + 2^{|(s')_l|} + |s'| = |s't| = |r'|.$$

Further, we obtain

$$|(r)_l| = |s| \geq |s'| = |(r')_l|.$$

In the second case the induction hypothesis yields $|t| > |t'|$, and we proceed as in the first case. □

Corollary 11. *Every $\mathsf{L_J}$-term is strongly normalizable.*

4 Conclusion

In this note we have proved a strong normalization theorem for the combinatory system generated by the combinator J. Since I and J form a basis for the $\lambda\mathsf{I}$-calculus our work shows that the following theorem of Toyama, Klop and Barendregt [4] does not apply in the context of combinatory logic: given two left-linear term rewriting systems R_1 and R_2, then we have that R_1 and R_2 are complete (i.e. confluent and terminating) if and only if the disjoint union of R_1 and R_2 is complete. The reason is that there is a hidden application function in the two systems generated by I and J, respectively. In the combinatory logic built up from I and J, these two functions are identified, whereas in the disjoint union of the two system, these application functions are distinct. Therefore, the above theorem cannot be applied.

References

[1] Hendrik Barendregt. *The Lambda Calculus.* North-Holland, revised edition, 1985.

[2] John B. Rosser. A mathematical logic without variables I. *Annals of Mathematics*, 36(1):127–150, 1935.

[3] Raymond Smullyan. *To Mock a Mockingbird.* Alfred A. Knopf, Inc., 1985.

[4] Yoshihito Toyama, Jan W. Klop, and Hendrik P. Barendregt. Termination for direct sums of left-linear complete term rewriting systems. *Journal of the ACM*, 42(6):1275–1304, 1995.

Address
Dieter Probst, Thomas Studer
Institut für Informatik und angewandte Mathematik, Universität Bern
Neubrückstrasse 10, CH-3012 Bern, Switzerland
{probst,tstuder}@iam.unibe.ch