The complexity of normal form rewrite sequences for
Associativity.*

Michael Niv
December 22, 1993

Abstract

The complexity of a particular term-rewrite system is considered: the rule of associativity

\((x * y) * z \triangleright x * (y * z)\).

Algorithms and exact calculations are given for the longest and shortest
sequences of applications of \(\triangleright\) that result in normal form (NF). The shortest NF sequence for a
term \(x\) is always \(n - d_{\text{rm}}(x)\), where \(n\) is the number of occurrences of \(*\) in \(x\) and \(d_{\text{rm}}(x)\) is the
depth of the rightmost leaf of \(x\). The longest NF sequence for any term is of length \(n(n - 1)/2\).

1 Preliminaries

(Klop 1992) provides an overview of the theory of term rewrite systems. There is relatively
little known about the complexity of various term rewrite systems. Here, I consider a particular
rewrite system with one binary connective \(*\) and one rewrite rule \((x * y) * z \triangleright x * (y * z)\).

A rewrite system \(\rightarrow\) is Strongly Normalizing (SN) iff every sequence of applications of \(\rightarrow\)
is finite. A rewrite system is Church-Rosser (CR) just in case

\[\forall x, y. (x \leftarrow \triangleright y \supset \exists z. (x \rightarrow z \land y \rightarrow z))\]

A rewrite system is Weakly Church-Rosser (WCR) just in case

\[\forall x, y, w. (w \rightarrow x \land w \rightarrow y) \supset \exists z. (x \rightarrow z \land y \rightarrow z)\]

Let \(\longrightarrow\) be the relation between two terms such that \(x \longrightarrow y\) just in case \(x\) contains a
subterm, the \textit{redex}, which matches the left hand side of the rule \(\triangleright\), and replacing the redex by
the corresponding right hand side, the \textit{contractum}, yields the new term \(y\). A term is in normal
form (NF) if it contains no redex. Let \(\leftarrow\) be the converse of \(\longrightarrow\). Let \(\leftrightarrow\) be \(\rightarrow \cup \leftarrow\). Let
\(\rightarrow\) be the reflexive transitive closure of \(\rightarrow\) and similarly, \(\leftarrow\) the reflexive transitive closure of
\(\leftarrow\), and \(\leftrightarrow\) the reflexive transitive closure of \(\leftrightarrow\). Note that \(\leftrightarrow\) is an equivalence relation.

Given a term \(x\), define \(\lambda(x)\) (resp. \(\rho(x)\)) refers to its the left (right) child of \(x\).

---

*This paper appears as Technical Report LCL 94-6 at the Computer Science Department of the Technion –
Israel Institute of Technology. It is also electronically archived in the Computation and Language E-Print Archive
as cmp-lg/9406030.
2 Longest rewrite sequence

Given a term $x$, $\#x$ and $\sigma x$ are defined as follows:

$$
\#x = \begin{cases} 
0 & \text{if } x \text{ is a leaf node} \\
1 + \#\lambda(x) + \#\rho(x) & \text{otherwise}
\end{cases}
$$

$$
\sigma x = \begin{cases} 
0 & \text{if } x \text{ is a leaf node} \\
\sigma \lambda(x) + \sigma \rho(x) + \#\lambda(x) & \text{otherwise}
\end{cases}
$$

Note that $\#x$ is the number of internal nodes in $x$. By convention, $n$ is $\#x$. Note also that if $x \triangleright x'$ then $x' = \lambda(\lambda(x)) * (\rho(\lambda(x)) * \rho(x))$ and $\sigma x' = \sigma x - (\#\lambda(x) + 1)$.

**Lemma 1** $\rightarrow$ is SN.

**proof** Every term $x$ is assigned a positive integer measure $\sigma x$. An application of $\rightarrow$ is guaranteed to lower the measure. This follows from monotonic dependency of $\sigma x$ upon the $\sigma$’s of each of $x$’s subterm, and from the fact that locally, applying $\triangleright$ lowers $\sigma$. \hfill $\square$

**Theorem 1** For every term $x$, there exists a NF-yielding sequence of $\sigma x$ applications of $\rightarrow$, furthermore, this is the longest possible NF sequence for $x$.

**proof** The sequence of constructed by induction on $\sigma x$:

Base case: $\sigma x = 0$. For every subterm $y$ of $x$, $\#\lambda(y) = 0$, i.e. $\lambda(y)$ is a leaf. So $x$ is in NF.

Induction: I show that for every term $x$ such that $\sigma(x) > 0$ there exists another term $x'$ such that $x \rightarrow x'$ and $\sigma x' = \sigma x - 1$. Let $y$ be the deepest leftmost descendant of $x$ such that $y$ is a redex. Note that $\lambda(\lambda(y))$ is a leaf (otherwise $\lambda(y)$ would be a deeper leftmost descendant redex). Let $y'$ such that $y \triangleright y'$. So $\#y' = \#y$, $\sigma y' = \sigma y - 1$ and by the straightforward dependency of $\sigma x'$ on the $\sigma$’s of each of $x'$’s subterms, in particular $y'$, $\sigma x' = \sigma x - 1$.

The maximality of the length of the rewrite sequence follows from the fact that the applications of $\rightarrow$ decrease $\sigma$ by the minimum amount possible, 1. \hfill $\square$

**Corollary 1** For every term $x$, every sequence of applications of $\rightarrow$ is of length at most $n(n-1)/2$.

**proof** It suffices to show that for every term $x$, $\sigma x \leq n(n-1)/2$. By induction on $n$:

Base case: $n = 1$, $\sigma x = 0$.

Induction: Suppose true for all terms $x'$ such that $\#x' < n$. Let $m = \#\lambda(x)$. So $0 \leq m \leq n - 1$ and $\#\rho(x) = n - m - 1$.

$$
\sigma x - n(n-1)/2 = \sigma \lambda(x) + \sigma \rho(x) + \#\lambda(x) - n(n-1)/2 \\
\leq \frac{m(m-1)}{2} + \frac{(n - m - 1)(n - m - 2)}{2} + m - \frac{n(n-1)}{2} \\
= (m + 1)(m - (n - 1)) \\
\leq 0 \quad \text{recalling that } 0 \leq m \leq n - 1
$$

\hfill $\square$
Corollary 2 There exists a term $x$ that can be rewritten to NF by a sequence of exactly $n(n-1)/2$ applications of $\rightarrow$.

proof

An $n$-left-chain is defined as follows: A 0-left-chain is a leaf. An $n$-left-chain is an $(n-1)$-left-chain $*$ a leaf. Let $x$ be an $n$-left-chain. $\#x = n$. I show by induction on $n$ that $\sigma x = n(n-1)/2$:

Base case: $n = 1$, $\sigma x = 0$.

Induction: Suppose true for an $(n-1)$-left-chain.

\[ \sigma x = \sigma x_1 + \#x_1 \]
\[ = (n-1)(n-2)/2 + n - 1 \]
\[ = n(n-1)/2 \]

\[ \square \]

3 Shortest rewrite sequence

I now show that a NF of a term (in fact the NF) can be computed in linear time.

Lemma 2 $\rightarrow$ is WCR.

proof

Let $w$ be a term with two distinct redexes $x$ and $y$, yielding the two distinct terms $w'$ and $w''$ respectively. There are a few possibilities: (without loss of generality, suppose $x$ is not a subterm of $y$.)

case 1: $y$ is either not a subterm of $x$ or it is a subterm of $\lambda(x)$ or a subterm of $\rho(x)$ or it is $\rho(x)$. In each case is clear that the order of application of $\rightarrow$ makes no difference.

case 2: $y = \lambda(x)$. For convenience let $x = ((a*b)*c)*d$. Applying $\triangleright$ at $x$ gives $(a*b)*(c*d)$; applying $\triangleright$ at $y$ gives $(a*(b*c))*d$. The former can be rewritten to $a*((b*c)*d)$ using one application of $\triangleright$, and the latter is rewritten first to $a*((b*c)*d)$ which is then rewritten to $a*(b*(c*d))$.

\[ \square \]

Lemma 3 (Newman) WCR $\land$ SN $\supset$ CR.

Lemma 4 CR $\land$ SN $\supset$ ($\forall x, y.(x \leftrightarrow y \land x, y$ are NFs $) \supset x = y$).

Theorem 2 $\rightarrow$ NFs are unique.

proof Follows from lemmas 1, 2, 3, and 4.

Therefore any deterministic computational path of applying $\rightarrow$ will lead to the NF. I now give an algorithm $\text{ctr}$ for computing NFs. It applies $\triangleright$ as close as possible to the root of its argument.
**Lemma 5** The depth of the rightmost leaf of \( x \) is \( n \) iff \( x \) is a NF.

**proof** \( x \) must be an \( n \)-right-chain — the mirror of an \( n \)-left-chain. \( \square \)

**Lemma 6** If \( d_{\text{RM}}(x) < n \), algorithm \( \text{ctr}_1 \) increases \( d_{\text{RM}}(x) \) by 1.

**proof** algorithm \( \text{ctr}_1 \) scans down the path from the root to the rightmost leaf, stopping at a redex \( y \). By applying \( \triangleright \), it pushes everything in \( \rho(y) \) (including the rightmost leaf) one arc further away from the root. 

So iterating \( \text{ctr}_1 \) \( n - d_{\text{RM}}(x) \) times computes the NF. This process is inefficient, as it needlessly rescans the prefix of its argument. The following algorithm avoids this inefficiency.

\[
\text{ctr}(x) \\
1. \quad y := x \\
2. \quad \text{while } y \text{ is not a leaf} \\
3. \quad \quad \text{while } y \text{ is not a leaf and } \lambda(y) \text{ is a leaf} \\
4. \quad \quad \quad y := \rho(y) \\
5. \quad \quad \text{if } y \text{ is not a leaf} \\
6. \quad \quad \quad \text{then apply } \triangleright \text{ to } y
\]

**Theorem 3** Given term \( x \), \( \text{ctr}(x) \) computes a NF for \( x \) in \( n - d_{\text{RM}}(x) \) applications of \( \rightarrow \).

**proof** Clearly, \( \text{ctr} \) gives the same result as \( \text{ctr}_1 \) run \( n - d_{\text{RM}}(x) \) times, that is, the NF of \( x \). \( \square \)

### 4 Application

Hepple and Morrill (1989) proposed using normal forms for overcoming certain difficulties with the parsing of Combinatory Categorial Grammar, a formalism for natural language syntax. The results above have been incorporated into an efficient parsing algorithm (Niv 1993, 1994).

### Bibliography

[Hepple and Morrill1989] Hepple, Mark R. and Glyn V. Morrill. 1989. Parsing and Derivational Equivalence. In *Proceedings of the Annual Meeting of the European Chapter of the Association for Computational Linguistics*. 4
[Klop1992] Klop, Jan W. 1992. Term Rewrite Systems. In Samson Abramsky, Dov M. Gabbay, and T. S. E. Maibaum (Eds.), *Handbook of Logic in Computer Science*, Vol. 2, 1 – 116. Oxford: Clarendon Press.

[Niv1993] Niv, Michael. 1993. *A Computational Model of Syntactic Processing: Ambiguity Resolution from Interpretation*. PhD thesis, University of Pennsylvania. (electronically: cmp-lg/9406029 or ftp://ftp.cis.upenn.edu/pub/ircs/tr/93-27.ps).

[Niv1994] Niv, Michael. 1994. A Psycholinguistically Motivated Parser for CCG. In *Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics*, to appear. (electronically: cmp-lg/9406031 or ftp://ftp.cis.upenn.edu/pub/niv/acl94.ps).