Strategy Independent Reduction Lengths in Rewriting and Binary Arithmetic

Hans Zantema
University of Technology, Eindhoven, The Netherlands
Radboud University, Nijmegen, The Netherlands
h.zantema@tue.nl

In this paper we give a criterion by which one can conclude that every reduction of a basic term to normal form has the same length. As a consequence, the number of steps to reach the normal form is independent of the chosen strategy. In particular this holds for TRSs computing addition and multiplication of natural numbers, both in unary and binary notation.

1 Introduction

For many term rewriting systems (TRSs) the number of rewrite steps to reach a normal form strongly depends on the chosen strategy. For instance, in using the standard if-then-else-rules

$$\mathsf{if}(\mathsf{true},x,y) \to x \quad \mathsf{if}(\mathsf{false},x,y) \to y$$

it is a good strategy to first rewrite the leftmost (boolean) argument of the symbol if until this argument is rewritten to false or true and then apply the corresponding rewrite rule for if. In this way redundant reductions in the third argument are avoided in case the condition rewrites to true, and redundant reductions in the second argument are avoided in case the condition rewrites to false. More general, choosing a good reduction strategy is essential for doing efficient computation by rewriting. Roughly speaking, for erasing rules, that is, some variable in the left-hand side does not appear in the right-hand side, it seems a good strategy to postpone rewriting this possibly erasing argument, as is the case in the above if-then-else example. Conversely, in case of duplicating rules, that is, a variable occurs more often in the right-hand side than in the left-hand side, it seems a good strategy to first rewrite the corresponding argument before duplicating it.

Surprisingly however, there are practical examples of TRSs including both such erasing and duplicating rules, where the strategy has no influence at all on the number steps required to reach a normal form. In this paper we investigate criteria for this phenomenon. As an example, using the results of this paper, we will show that for multiplying two natural numbers by the standard TRS

$$\begin{array}{rclrcl} \mathsf{plus}(0,x) & \to & x & \mathsf{mult}(0,x) & \to & 0 \\ \mathsf{plus}(\mathsf{s}(x),y) & \to & \mathsf{s}(\mathsf{plus}(x,y)) & \mathsf{mult}(\mathsf{s}(x),y) & \to & \mathsf{plus}(\mathsf{mult}(x,y),y) \end{array}$$

the number of steps to reach the resulting normal form is independent of the chosen strategy. Note that the third rule of this TRS is erasing, and the last rule is duplicating.

In this TRS ground terms reduce to ground normal forms of the shape $\mathsf{s}^n(0)$ for any natural number n, representing the corresponding number n in unary notation. For large numbers n this is a quite inefficient representation. Much more efficient is the binary representation. In this paper we also give a TRS for doing addition and multiplication in binary representation. Here for every addition or multiplication of two positive integer numbers we show by the main theorem of this paper that every reduction to normal form has the same number of steps. Moreover, this number has the same complexity as the standard binary algorithms: linear for addition and quadratic for multiplication.

Throughout the paper we assume familiarity with standard notions in rewriting like orthogonality and normal forms as they are introduced in e.g. [4]. For instance, a *reduction* is a sequence of rewrite steps, and a *reduct* of a term t is a term u for which there is a reduction from t to u.

In Section 2 we investigate the diamond property and present our main result in Theorem 3: a criterion by which for every basic term all reductions to normal form have the same length. Here a term is called basic if only its root is a defined symbol. We apply this to unary arithmetic. In Section 3 it is shown how our theorem applies to binary arithmetic. We conclude in Section 4.

2 The diamond property and the main result

We say that a binary relation $\to$ satisfies the *diamond property*, if for every three elements s,t,u satisfying $s \to t$ and $s \to u$ and $t \neq u$, there exists an element w such that $t \to w$ and $u \to w$.

This notion is slightly different from other variants of the diamond property, for instance introduced in [4] where it is typically used for reflexive relations. However, as we are interested in the exact number of steps to reach a normal form, our present version is the most natural in our context. In earlier texts it was sometimes called WCR[1]. It is an instance of the *balanced weak Church-Rosser property* from [5]. The following lemma is a direct consequence of both Lemma 1 from [5] and the random descent lemma from [3]; to make the paper self-contained here we give a proof too.

Lemma 1 *Let $\to$ be a relation satisfying the diamond property. Then every element has at most one normal form, and for every element having a normal form it holds that every reduction of this element to its normal form has the same number of steps.*

Proof. This follows from the following claim:

> Let t be an element having normal form n. Then every reduct of t has normal form n, and every $\to$-reduction from t to n has the same number of steps.

We prove this by induction on k, where k is the number of steps of a reduction from t to n.

If $k = 0$ then t is a normal form, and the lemma holds.

If $k > 0$ assume $t \to t_1 \to^{k-1} n$ and $t \to^p u$; we will prove that $u \to^q n$ for q satisfying $p+q = k$. Then indeed the lemma follows by applying this claim both for u being the arbitrary reduct of t and for $u = n$.

For $p = 0$ the claim trivially holds, so assume $p > 0$ and $t \to t_2 \to^{p-1} u$. In case $t_1 = t_2$ we apply the induction hypothesis to $t_1 = t_2$ and we are done. In the remaining case by the diamond property there exists v such that $t_1 \to v$ and $t_2 \to v$. By applying the induction hypothesis to t_1 we conclude $v \to^{k-2} n$, yielding $t_2 \to^{k-1} n$. Next we apply the induction hypothesis to t_2, so $u \to^* n$ and every reduction from t_2 to n has length $k-1$, $p+q = k$, concluding the proof. $\square$

An orthogonal TRS is said to be *variable preserving*[1] if for every rule $\ell \to r$ every variable occurring in ℓ occurs exactly once in r. For instance, the TRS consisting of the two rules for plus as given in the introduction, is variable preserving.

Lemma 2 *Let R be a variable preserving orthogonal TRS. Then the relation $\to_R$ on the set of all terms satisfies the diamond property.*

[1] Some texts have a weaker notion of *variable preserving*, but our version is more suitable for investigating reduction lengths.

Proof. Similar to the proof of the well-known critical pair lemma (see e.g., [4], Lemma 2.7.15) we do a case analysis on the redex patterns. If a term t has one-step reductions to two distinct terms u and v, then by orthogonality one of the following cases holds:

1. (the redexes are parallel) $t = C[\ell_1^\sigma, \ell_2^\tau]$ and $u = C[r_1^\sigma, \ell_2^\tau]$ and $v = C[\ell_1^\sigma, r_2^\tau]$, for a two-hole context C, substitutions, σ, τ and rules $\ell_i \to r_i$ in R for $i = 1,2$,
2. (one redex is above the other) $t = C[\ell^\sigma]$ and $u = C[r^\sigma]$ and $v = C[\ell^\tau]$, for a rule $\ell \to r$ in R and a context C, and substitutions σ, τ such that $x^\sigma \to_R x^\tau$ for some variable x occurring in ℓ, and $y^\sigma = y^\tau$ for all variables y distinct from x,

or the same in which u and v are swapped. For both cases we have to find w such that $u \to_R w$ and $v \to_R w$. For case (1) this holds by choosing $w = C[r_1^\sigma, r_2^\tau]$.

For case (2) write $\ell = D[x]$. Since R is left-linear, x does not occur in D. Since R is variable preserving, we can write $r = E[x]$ for a context E not containing x. Now we obtain

$$u = C[r^\sigma] = C[E^\sigma[x^\sigma]] \to_R C[E^\sigma[x^\tau]] \text{ and } v = C[\ell^\tau] \to_R C[r^\tau] = C[E^\tau[x^\tau]].$$

Since x does not occur in E we have $E^\tau = E^\sigma$, by which $w = C[E^\tau[x^\tau]] = C[E^\sigma[x^\tau]]$ satisfies the requirements for w. □

As a direct consequence of Lemmas 1 and 2 we conclude that for all terms with respect to the two plus rules from the introduction the number of steps to reach the normal form is independent of the strategy. This does not hold any more for the full system also containing the rules for mult. For instance, the term mult(0, plus(0, 0)) admits the following two reductions to normal form having lengths one and two, respectively:

$$\mathsf{mult}(0,\mathsf{plus}(0,0)) \to_R 0, \;\; \mathsf{mult}(0,\mathsf{plus}(0,0)) \to_R \mathsf{mult}(0,0) \to_R 0.$$

However, here the starting term mult(0, plus(0, 0)) is not *basic*: it contains more than one *defined symbol*. A symbol is called a *defined symbol* if it occurs as the root of the left-hand side of a rule. Similar as in texts on run-time complexity like [2] we define:

A term is defined to be *basic* if the root is a defined symbol, and it is the only defined symbol occurring in the term.

We will prove that for basic terms like terms of the shape $\mathsf{mult}(\mathsf{s}^m(0), \mathsf{s}^n(0))$ every reduction to its normal form $\mathsf{s}^{mn}(0)$ has the same length.

Theorem 3 *Let R be an orthogonal TRS over Σ, and $\Sigma' \subseteq \Sigma$, such that every rule $\ell \to r$ of R is of one of the following shapes:*

- *$\ell \to r$ is variable preserving, and neither ℓ nor r contain symbols from Σ',*
- *the root of ℓ is in Σ', and for every symbol in r from Σ' the arguments of this symbol do not contain defined symbols.*

Then any two reductions of a basic term to normal form have the same length.

Proof. First we prove the following claim.

Claim: If t is a basic term and $t \to_R^* u$, then for every symbol from Σ' in u the arguments of this symbol do not contain defined symbols.

We prove this claim by induction on the number of steps in the reduction $t \to_R^* u$. If this number is 0, then $u = t$ and the claim follows from the definition of basic term.

If this number is > 0, then $t \to_R^r u' \to_R u$, where by induction hypothesis we assume that the claim holds for u'. For every subterm of u having its root in Σ', we have to prove that this subterm does not contain other defined symbols. For every such subterm that also occurs in u' this holds by the induction hypothesis. The only way such a subterm may occur in u but not in u' is that it is created by replacing ℓ^σ by r^σ for some substitution σ and a rule $\ell \to r$ of the second type: the root of ℓ is in Σ' and arguments of symbols from Σ' in r do not contain defined symbols. Since ℓ^σ is a subterm of u' and the property holds for u' by induction hypothesis, for all variables x in ℓ the term x^σ does not contain defined symbols. Hence every subterm of r^σ having its root in Σ' does not contain other defined symbols, concluding the proof of the claim.

Next we show that the relation $\to_R$ restricted to terms u obtained by $t \to_R^* u$ for some basic term t, satisfies the diamond property. To prove this, let such a term u both rewrite to v and to w, $v \neq w$. In case the redexes are parallel, then the diamond property is easily concluded as in the proof of Lemma 2, case 1.

In the remaining case one redex is above the other. Due to the claim and the shape of the rules, one of the following cases holds:

- The roots of both redexes are not in Σ'. Then both reduction steps are with respect to variable preserving rules, and the diamond property follows from Lemma 2.
- The root of the innermost (of the two) redexes is in Σ', but the root of the outermost redex is not. Then the reduction with respect to the outermost redex is variable preserving, by which the diamond property can be concluded.

In all cases the diamond property of $\to_R$ restricted reducts u of a basic term t can be concluded. Now the theorem follows from Lemma 1. □

Indeed now by Theorem 3 we can conclude that with respect to the TRS with plus and mult as given in the introduction for every basic term of the shape $\mathsf{mult}(\mathsf{s}^m(0), \mathsf{s}^n(0))$ every reduction to its normal form $\mathsf{s}^{mn}(0)$ has the same length: let Σ' consist of the single symbol mult. Then the two rules for plus satisfy the condition for rules of the first type, and the two rules for mult satisfy the condition for rules of the second type, by which the claim follows from Theorem 3.

As another example consider the Fibonacci function fib defined by the rules

$$\begin{array}{rcl@{\qquad}rcl}
\mathsf{plus}(0,x) & \to & x & \mathsf{fib}(0) & \to & 0 \\
\mathsf{plus}(\mathsf{s}(x),y) & \to & \mathsf{s}(\mathsf{plus}(x,y)) & \mathsf{fib}(\mathsf{s}(0)) & \to & \mathsf{s}(0) \\
 & & & \mathsf{fib}(\mathsf{s}(\mathsf{s}(x))) & \to & \mathsf{plus}(\mathsf{fib}(x),\mathsf{fib}(\mathsf{s}(x))).
\end{array}$$

Choosing Σ' to consist only of the symbol fib, all requirements of Theorem 3 hold, so we conclude that for every k every reduction of $\mathsf{fib}(\mathsf{s}^k(0))$ to normal form has the same length.

For another standard recursive function, the factorial, it does not hold that distinct reductions of basic terms to normal forms have the same length. By extending our basic rules for plus and mult from the introduction by the rules for factorial

$$\begin{array}{rcl}
\mathsf{fact}(0) & \to & \mathsf{s}(0) \\
\mathsf{fact}(\mathsf{s}(x)) & \to & \mathsf{mult}(\mathsf{s}(x),\mathsf{fact}(x))
\end{array}$$

the basic term $\mathsf{fact}(\mathsf{s}(0))$ admits reductions of lengths both 5 and 6 to its normal form $\mathsf{s}(0))$, caused by the fact that after

$$\mathsf{fact}(\mathsf{s}(0)) \to \mathsf{mult}(\mathsf{s}(0),\mathsf{fact}(0)) \to \mathsf{plus}(\mathsf{mult}(0,\mathsf{fact}(0)),\mathsf{fact}(0))$$

the subterm $\mathsf{mult}(0,\mathsf{fact}(0))$ can be rewritten to 0 in one step, but also in two steps via $\mathsf{mult}(0,\mathsf{s}(0))$. Indeed Theorem 3 does not apply here since mult should be in Σ' as before, and also $\mathsf{fact} \in \Sigma'$ since mult occurs in the right-hand side of a rule for fact. But then this rules violates the conditions: $\mathsf{fact}(x)$ occurs as an argument of the symbol mult in this right-hand side.

The converse of Theorem 3 does not hold, not even for terminating orthogonal exhaustive constructor systems. Here exhaustive (in case of termination equivalent to *sufficiently complete*) means that for every defined symbol applied to ground constructor forms at least one rule is applicable, by which all ground normal forms are constructor terms.

For instance, consider the TRS consisting of the three rules

$$\mathsf{f}(0) \to 0, \quad \mathsf{f}(\mathsf{s}(x)) \to \mathsf{s}(0), \quad \mathsf{g}(x) \to \mathsf{f}(\mathsf{f}(x))).$$

As f and g are defined symbols, the only basic terms are of the shape $\mathsf{f}(\mathsf{s}^n(x))$, $\mathsf{f}(\mathsf{s}^n(0))$, $\mathsf{g}(\mathsf{s}^n(x))$ or $\mathsf{g}(\mathsf{s}^n(0))$, for $n \geq 0$. All of these terms only admit one single reduction to normal form of at most three steps, as is easily checked by applying the rules. So any two reductions of a basic term to normal form have the same length. However, this can not be concluded from Theorem 3 since this TRS does not satisfy its conditions: the rule $\mathsf{f}(\mathsf{s}(x)) \to \mathsf{s}(0)$ is not variable preserving, so for satisfying the conditions it should hold $\mathsf{f} \in \Sigma'$. But then the third rule does not satisfy the conditions as there is a nested occurrence of f in the right-hand side.

3 Binary arithmetic

For more efficient computation of numbers it is natural to exploit binary notation, in which the size of the representation is logarithmic rather than linear in the value of the number, and in which basic arithmetic can be executed by rewriting in a complexity that is polynomial in the size of this representation, so logarithmic in the values of the numbers. Although the general idea is folklore, there is not a single fixed standard. Here we present a straightforward way to proceed for implementing binary arithmetic by rewriting. It is related to the non-orthogonal system from [6] for n-ary arithmetic for arbitrary n for all integers. By restricting to $n = 2$ and positive integers, here we succeed in presenting an orthogonal system.

In standard binary notation positive integers can be seen to be uniquely composed from a constant 1 representing value 1, and two unary operators .0 and .1, where .0 means putting a 0 behind the number, by which its value is duplicated, and .1 means putting a 1 behind the number, by which its value x is replaced by $2x+1$.

Every positive integer has a unique representation as a ground term over these three symbols 1, .0 and .1, corresponding to the usual binary notation in which a postfix notation for .0 and .1 is used. For instance, the number 29 is 11101 in binary notation, and is written as 1.1.1.0.1 as a postfix ground term. Since in term rewriting it is more standard to use prefix notation rather than postfix notation, here we choose to switch to prefix notation. So instead for 29 we write $.1(.0(.1(.1(1))))$. Introducing a constant 0 would violate unicity, that is why we restrict to positive integers.

In order to express addition and multiplication in this notation the successor succ is needed as an additional operator, having rewrite rules

$$\begin{array}{rcl} \mathsf{succ}(1) & \to & .0(1) \\ \mathsf{succ}(.0(x)) & \to & .1(x) \\ \mathsf{succ}(.1(x)) & \to & .0(\mathsf{succ}(x)) \end{array}$$

Now we can express addition:

$$\begin{array}{rcl} \mathsf{plus}(1,x) & \to & \mathsf{succ}(x) \\ \mathsf{plus}(.0(x),1) & \to & .1(x) \\ \mathsf{plus}(.1(x),1) & \to & .0(\mathsf{succ}(x)) \end{array} \qquad \begin{array}{rcl} \mathsf{plus}(.0(x),.0(y)) & \to & .0(\mathsf{plus}(x,y)) \\ \mathsf{plus}(.0(x),.1(y)) & \to & .1(\mathsf{plus}(x,y)) \\ \mathsf{plus}(.1(x),.0(y)) & \to & .1(\mathsf{plus}(x,y)) \\ \mathsf{plus}(.1(x),.1(y)) & \to & .0(\mathsf{succ}(\mathsf{plus}(x,y))) \end{array}$$

Indeed now for every ground term composed from $1, .0, .1, \mathsf{succ}, \mathsf{plus}$ containing at least one symbol succ or plus a rule is applicable. So the ground normal forms are the ground terms composed from $1, .0, .1$, exactly being the binary representations of positive integers. Since the TRS is easily proved to be terminating, e.g., by recursive path order, every ground term will reduce to such a ground normal form, being the binary representations of a positive integer. As by every rule the numeric value of the term is preserved, this TRS serves for computing the binary value of any ground term considered so far.

Surprisingly, it is very simple to extend this system to multiplication, by using the fresh symbol mult for multiplication and introducing the following rules

$$\begin{array}{rcl} \mathsf{mult}(1,x) & \to & x \\ \mathsf{mult}(.0(x),y) & \to & .0(\mathsf{mult}(x,y)) \\ \mathsf{mult}(.1(x),y) & \to & \mathsf{plus}(.0(\mathsf{mult}(x,y)),y) \end{array}$$

The above observations are also easily checked for the extended TRS consisting of all rules presented so far: all rules preserve values, and for every ground term composed from $1, .0, .1, \mathsf{succ}, \mathsf{plus}, \mathsf{mult}$ containing at least one symbol succ or plus or mult a rule is applicable. So the ground normal forms are the ground terms composed from $1, .0, .1$, exactly being the binary representations of positive integers. Also the extended TRS is easily proved to be terminating, e.g., by recursive path order, so every ground term will reduce to such a ground normal form, being the binary representations of a positive integer. So the extended TRS also serves for executing multiplication.

Choosing Σ' to consist of the single symbol mult it is easily checked that all conditions of Theorem 3 are satisfied. So by Theorem 3 we conclude that for using this TRS for computing the addition or multiplication of two binary numbers, every reduction to normal form has the same length. This observation is of great value: a lot of effort is done in choosing the right rewriting strategy; this observation shows that when binary arithmetic is only used for addition and multiplication (which is often the case), strategy optimization is useless since all reductions to normal form are of the same complexity as they all have the same length.

In [7] it is proved that for addition this reduction length is linear in the size of the arguments, and for multiplication it is quadratic in the size of the arguments, so having the same complexity as the standard algorithms for binary addition and multiplication. The TRSs as presented in [7] have been the basis of the implementation of integer number computation in the mCRL2 tool set [1].

In our rules for multiplication the recursion is only in the first argument. For computing $2^n * m$ for any number m this is very efficient: the result is obtained after $O(n)$ steps, independent of the value of m. However, for computing $m * 2^n$ this is not efficient at all: first m will be decomposed and many copies of 2^n will be created, finally giving the same result as the computation of $2^n * m$. To bring more symmetry and to make the computations of $m * 2^n$ and $2^n * m$ both be linear in n, we can choose the rules for mult

to be

$$\begin{array}{rcl}
\mathsf{mult}(1,1) & \to & 1 \\
\mathsf{mult}(.0(x),1) & \to & .0(x)) \\
\mathsf{mult}(.1(x),1) & \to & .1(x)) \\
\mathsf{mult}(1,.0(x)) & \to & .0(x)) \\
\mathsf{mult}(1,.1(x)) & \to & .1(x)) \\
\mathsf{mult}(.0(x),.0(y)) & \to & .0(.0(\mathsf{mult}(x,y))) \\
\mathsf{mult}(.0(x),.1(y)) & \to & .0(\mathsf{mult}(x,.1(y))) \\
\mathsf{mult}(.1(x),.0(y)) & \to & .0(\mathsf{mult}(.1(x),y)) \\
\mathsf{mult}(.1(x),.1(y)) & \to & \mathsf{plus}(.0(\mathsf{mult}(x,.1(y))),.1(y)).
\end{array}$$

In this way duplication of arguments and introducing plus is only done if both arguments of mult are of the shape $.1(\cdots)$. Although slightly more complicated, it is easy to check that the full system consisting of the rules for succ, plus and these new rules for mult

- only rewrites terms to terms having the same value,
- is terminating (again by recursive path order),
- satisfies the conditions of Theorem 3 for $\Sigma' = \{\mathsf{mult}\}$, and
- computes both $m * 2^n$ and $2^n * m$ in $O(n)$ rewrite steps, independent of the value of m.

Hence by Theorem 3, we conclude that for this improved system for binary arithmetic for every basic term any two reductions to normal form have the same length.

4 Conclusion

Basic terms are typical terms to be rewritten: a defined symbol on top and constructor terms as arguments. We gave a criterion for orthogonal TRSs by which all reductions of a basic term to normal form have the same length, showing that the reduction length is independent of the chosen strategy. This applies to both addition and multiplication, both in unary and binary notation. This result is surprising: it shows that for these basic computations as they often occur in practice, all possible reduction strategies have the same complexity.

In unary notation the Fibonacci function still satisfies our criteria. However, for more complicated user defined functions experiments based on a simple implementation show that reduction lengths of basic terms are typically not strategy independent any more.

Acknowledgment. We want to thank Evans Kaijage for fruitful discussions on this topic and for doing some experiments.

References

[1] J. F. Groote et al: *The mCRL2 tool set*. `http://mcrl2.org/mcrl2/wiki/index.php/Home`.

[2] Martin Avanzini & Georg Moser (2010): *Closing the Gap Between Runtime Complexity and Polytime Computability*. In Christopher Lynch, editor: *Proceedings of the 21st International Conference on Rewriting Techniques and Applications*, *Leibniz International Proceedings in Informatics (LIPIcs)* 6, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, Germany, pp. 33–48.

[3] V. van Oostrom (2007): *Random Descent*. In F. Baader, editor: *Proceedings of the 18th Conference on Rewriting Techniques and Applications (RTA)*, *Lecture Notes in Computer Science* 4533, Springer, pp. 314–328, doi:10.1007/978-3-540-73449-9_24.

[4] Terese (2003): *Term Rewriting Systems*. Cambridge University Press.

[5] Y. Toyama (2005): *Reduction Strategies for Left-Linear Term Rewriting Systems*. In A. Middeldorp, V. van Oostrom, F. van Raamsdonk & R. de Vrijer, editors: *Processes, Terms and Cycles: Steps on the Road to Infinity:Essays Dedicated to Jan Willem Klop on the Occasion of His 60th Birthday*, *Lecture Notes in Computer Science* 3838, Springer, pp. 198–223, doi:10.1007/11601548_13.

[6] H. R. Walters & H. Zantema (1995): *Rewrite Systems for Integer Arithmetic*. In J. Hsiang, editor: *Proceedings of the 6th Conference on Rewriting Techniques and Applications*, *Lecture Notes in Computer Science* 914, Springer, pp. 324 – 338, doi:10.1007/3-540-59200-8_67.

[7] H. Zantema (2003): *Basic arithmetic by rewriting and its complexity*. Available at `http://www.win.tue.nl/~hzantema/aritm.pdf`.