DOI 10.1070/SM2015v206n09ABEH004493

On some slowly terminating term rewriting systems

L. D. Beklemishev and A. A. Onoprienko

Abstract. We formulate some term rewriting systems in which the number of computation steps is finite for each output, but this number cannot be bounded by a provably total computable function in Peano arithmetic PA. Thus, the termination of such systems is unprovable in PA. These systems are derived from an independent combinatorial result known as the Worm principle; they can also be viewed as versions of the well-known Hercules-Hydra game introduced by Paris and Kirby.

Bibliography: 16 titles.

Keywords: term rewriting systems, Peano arithmetic, Worm principle.

§ 1. Introduction

The first examples of term rewriting systems and important classes of these appeared at the beginning of the 20th century, in the works of Thue (Thue systems), Schönfinkel and Curry (combinatory logic), Herbrand and Gödel (Herbrand-Gödel computability), Church (λ-calculus), Post, Markov and other authors. Nowadays, term rewriting systems are widely used in informatics and, in particular, in functional programming and in computer algebra systems. Investigating their properties, such as termination (which means that each chain of transformations following the rules of the system terminates), confluence (branches of computation with the same initial state can be continued to meet), the existence and uniqueness of normal forms, is an important field in computer science (see [1] and [2]).

In this paper we look at systems which can be called *slowly terminating*. Each chain of transformations in such a system is finite, so that the system is terminating, but the number of steps in the chain cannot be bounded by a function of a reasonable order of growth in the length of the initial term.

Given a system W we define its complexity function $C_W(n)$ as the maximum possible number of steps in a chain of transformations of W starting from a term of length at most n. (If for some n there exists no upper bound for the number of steps, then the system W is nonterminating.) The complexity function of the slowly terminating systems presented in this paper is not provably total in Peano arithmetic PA, and the lower bound on the number of steps exceeds any function

L. D. Beklemishev's research was financed by a grant of the Russian Science Foundation (project no. 14-50-00005) at the Steklov Mathematical Institute of the Russian Academy of Sciences. Sections 1–5 were written by L. D. Beklemishev, and § 6 was written by L. D. Beklemishev together with A. A. Onoprienko.

AMS 2010 *Mathematics Subject Classification.* Primary 68Q42, 03D03, 03F45, 03F30.

in the so-called extended Ggrzegorczyk hierarchy up to the ordinal ε_0 (see [3]). In particular, the termination of such systems cannot be proved in PA.

The existence of slowly terminating term rewriting systems is a consequence of some classical results in mathematical logic. It is well known that term rewriting systems, and even certain more special subclasses of these such as Thue systems, are a universal model of computation. Since the set of sentences provable in PA is computably enumerable, we can consider a computable enumeration $(\varphi_n)_{n\in\mathbb{N}}$ of all computable functions from $\mathbb{N}$ to $\mathbb{N}$ which are provably total in PA. Then the function

$$\psi(n) = \max_{i,m\leqslant n} \varphi_i(m) + 1$$

satisfies the inequality $\psi(n) > \varphi_k(n)$ for all $n \geqslant k$. Since ψ is a computable function, we can easily construct a finite term rewriting system W with $C_W(n) > \psi(n)$, so that C_W exceeds any provably total computable function in PA.

Examples of systems which arise from this construction are based on the method of proof encoding used in Gödel's incompleteness theorem, so they are rather awkward and difficult to write out explicitly. One can ask if there exist natural, easy to formulate and memorable examples of such systems. One can give an answer to this question based on some independent combinatorial principles, such as the well-known Hercules-Hydra principle (see [4]).

The first example of using a term rewriting system to interpret the Hercules-Hydra battle was formulated by Dershowitz and Jouannaud [5]. Subsequently, that system was corrected in [6]; to analyze it accurately has proved to be rather complicated.

An alternative sequence of systems was proposed by Touzet [7]. In these systems the question of termination is simpler to analyse, but they only correspond to a restricted version of the Hercules-Hydra battle. In particular, the termination for each individual system is provable in PA. The strength of these systems approximates PA from below, but the arity of the signature functions involved in these systems increases ad infinitum.

The Worm principle [8], [9] is one of the simplest combinatorial principles, which looks almost like a statement of termination of a particular word rewriting system. It also leads to some natural examples of slowly terminating term rewriting systems. We present three such systems here. The first is a word rewriting system corresponding very closely to the nondeterministic version of the Worm principle. Its rules are extremely simple, but the alphabet and the system of rules are infinite (albeit recursive). The second system has a finite alphabet and is also easily memorable, but its system of rules is still infinite. The third system, obtained by adapting the second, is finite and allows for a sufficiently simple analysis of termination, but it is not as elementary as the first two.

Our results here can be summarized as follows.

Theorem 1. *Let W be any one of the three systems W_1, W_2 and W_3 formulated below. Then*

(i) *W is terminating*;

(ii) *the complexity function $C_W(n)$ is not bounded by a provably total computable function in* PA;

(iii) *the termination of W is unprovable in* PA.

The results in this paper were first announced in [10]. At the same time, Buchholz [11] gave another elegant encoding of the Hercules-Hydra game, related to the Derschowitz system, and gave a simple proof that it terminates. In publishing the results from [10] here we have been motivated by investigations by Zankl, Winkler and Middeldorp [12], who used the systems W_i as test examples for the methods they were developing for computer-aided proofs of termination for term rewriting systems. They could prove termination for the system W_3 using computer calculations. Furthermore, they found a rule that was forgotten in the formulation of the system W_3 in [10], but is actually necessary to model the Worm principle. In this paper we give our own proof of termination for the corrected formulation of W_3.

§ 2. Term rewriting systems

By a *signature* we mean a set Σ of function symbols of various arities (function symbols of arity zero are the constants). Apart from the symbols of the signature we consider an infinite alphabet of variables $\{x_0, x_1, \dots\}$ and auxiliary symbols for the brackets and the comma. The set of *terms* Tm_Σ of Σ is defined recursively as usual: variables and constants are terms, and if $f \in \Sigma$ is a function symbol of arity $n > 0$ and $t_1, \dots, t_n$ are terms, then the expression $f(t_1, \dots, t_n)$ is a term. By *closed terms* we mean those containing no variables. Terms can be conveniently represented as finite trees, whose leaves are labelled by variables or constants and whose interior vertices are labelled by the function symbols of the signature. The arity of a label must correspond to the degree of the corresponding vertex.

By a *term rewriting rule* we mean an expression of the form $l \to r$, where l and r are terms. A *term rewriting system* for a signature Σ is an arbitrary set W of such rules for terms of Σ.

To define the concept of derivation in a given term rewriting system we require the concepts of a context and a substitution. A *context* $t[x]$ is a term with a unique occurrence of the distinguished variable x. The result of the replacement of x by a term s in a context t is denoted by $t[s]$.

A *substitution* is a map $\sigma\colon \mathrm{Tm}_\Sigma \to \mathrm{Tm}_\Sigma$ which commutes with all the symbols of the signature:

$$(f(t_1, \dots, t_n))^\sigma = f(t_1^\sigma, \dots, t_n^\sigma).$$

Here, t^σ denotes the result of applying σ to t. In fact, t^σ is obtained by replacing all the variables in t by certain terms.

By an *application of the rule* $l \to r$ we mean the transformation of the term $t[l^\sigma]$ into $t[r^\sigma]$, for some substitution σ and some context $t[x]$. For terms s and t we write $s \to_W t$ if we can obtain t from s by applying some rule of the system W. If it is clear which system W is under consideration, then we drop the subscript W from the arrow.

By a *derivation of a term t_n from a term t_1 in W* we mean a sequence of applications of rules of this system of the form:

$$t_1 \to_W t_2 \to_W \cdots \to_W t_{n-1} \to_W t_n.$$

We write $s \to_W^* t$ if t can be derived from s in W.

The term rewriting system W *terminates* if for each initial term t any chain of applications of rules in W starting from t is finite.

Note that string rewriting systems, also known as *semi-Thue systems*, can be regarded as a particular kind of term rewriting systems for a signature containing only function symbols of arity 1. Each symbol corresponds to a letter in the alphabet of the Thue system in question. Then a word $A = a_1 \dots a_n$ corresponds to the term $A(x) = a_1(\cdots a_n(x)\cdots)$, where x is a fixed variable, and each rewrite rule $A \to B$ in the Thue system corresponds to the term rewrite rule $A(x) \to B(x)$. The concept of derivation also matches in the two formalisms.

To prove that a particular term rewriting system terminates, authors often use some well-founded ordering of terms, chosen so that each application of the rules results in a decrease of the terms. One standard choice is the so-called *lexicographic path ordering*; see [5].

Assume that Σ is finite and a strict linear ordering $\prec$ is defined on Σ. Then we define the associated order relation $\prec_{\mathrm{lpo}}$ on Tm_Σ (together with its reflexive closure $\preceq_{\mathrm{lpo}}$) recursively as follows.

We set $s \succ_{\mathrm{lpo}} t$ if and only if either t is a variable occurring in s and $t \neq s$ or $s = f(s_1, \dots, s_n)$, $t = g(t_1, \dots, t_m)$ and one of the following conditions is satisfied:

1) $s_i \succeq_{\mathrm{lpo}} t$ for some i, $1 \leqslant i \leqslant n$;
2) $f \succ g$ and $s \succ_{\mathrm{lpo}} t_j$ for all j, $1 \leqslant j \leqslant m$;
3) $f = g$, $s \succ_{\mathrm{lpo}} t_j$ for all j, $1 \leqslant j \leqslant m$, and for some i, $1 \leqslant i \leqslant n$, we have $s_1 = t_1, \dots, s_{i-1} = t_{i-1}$, $s_i \succ_{\mathrm{lpo}} t_i$.

It is well known that the ordering $\prec_{\mathrm{lpo}}$ is well-founded. Furthermore, if each rule $l \to r$ of the system W satisfies $l \succ_{\mathrm{lpo}} r$, then each application of this rule $t[l^\sigma] \succ_{\mathrm{lpo}} t[r^\sigma]$ also has this property, therefore the system W terminates (see [5], [13] and [14]).

§ 3. The Worm principle

Now we formulate an independent combinatorial statement known as the *Worm principle* (see [8], [9], [15]). Words in the alphabet of natural numbers $\mathbb{N}$ will be called *worms*; we denote the set of all such words by S. The length of a word A is denoted by $|A|$; the *height* of a worm is its largest letter. The *size* of the worm A is defined as the maximum of its length and height. The first element of the worm $A = n_0 n_1 \dots n_k$ will be called its *head.* Let S_n denote the set of words in the sub-alphabet $\{i \colon i \geqslant n\}$.

Informally speaking, the life of a worm is described by the following process: if the head of the worm is equal to 0, then it is cut off at the next step; otherwise the head decreases by 1, but the worm regenerates in accordance with the simple rule below. The Worm principle states that a worm cannot live forever.

More formally, for each $m \in \mathbb{N}$ we specify a *worm function* $A \mapsto A[m]$ mapping worms to worms. For each initial worm $A = n_0 n_1 \dots n_k$ this function defines a sequence $(A_i)_{i\in\mathbb{N}}$, called the *evolution* of the worm:

$$A_0 := A, \qquad A_{i+1} := A_i[i+1].$$

Thus, m counts the steps of the process. The rules defining $A[m]$ are as follows:

1) if $n_0 = 0$, then $A[m] := n_1 \dots n_k$; in this case the head of the worm is cut off;

2) if $n_0 = n+1 > 0$, then we find the maximal initial piece n_0B of A such that $B \in S_{n+1}$ (B may be empty); if $A = n_0BC$, then

$$A[m] := (nB)^{m+1}C.$$

Example 1. Consider the evolution of the worm $A = 1302$. At the first step we obtain $B = 3$ and $A_1 = A[1] = 030302$. Then the following sequence appears:

$$\begin{aligned}
&A_0 = 1302,\\
&A_1 = 030302,\\
&A_2 = 30302,\\
&A_3 = 22220302,\\
&A_4 = (1222)^5 0302,\\
&\dots\dots\dots\dots\dots\dots\dots
\end{aligned}$$

Note that for each word A we have $|A[m]| \leqslant |A| \cdot (m+1)$, so the lengths of the A_i are bounded by $|A_i| \leqslant |A| \cdot (i+1)!$.

The *Worm principle* states that for each initial word A there exists $i \geqslant 0$ such that $A_i = \Lambda$.

Let $D(n)$ denote the maximal lifespan of worms of size at most n. Since the number of worms of size n is finite, the Worm principle is equivalent to the assertion that $D(n)$ is well defined for each $n \in \mathbb{N}$. The next theorem follows from the results in [8] and [9].

Theorem 2. (i) *For each $A \in S$ there exists i such that $A_i = \Lambda$.*

(ii) *Statement* (i) *is not provable in Peano arithmetic* PA.

(iii) *The function $D(n)$ majorizes each computable function provably total in* PA.

Proof. For the convenience of the reader we give the proof of (i) here. The proofs of (ii) and (iii) are more complicated (see, for instance, [15]), and (ii) is a direct consequence of (iii).

We first define a function $o\colon S \to \varepsilon_0$ and show that in transformations of worms the corresponding ordinals decrease. Let B^+ denote the result of the replacement of each letter n with $n+1$ in the word B, and let B^- denote the result of the inverse operation (which is defined for $B \in S_1$). The ordinal $o(A)$ is defined by induction on the height of A.

If $A = 0^k$, then $o(A) := k$. Otherwise A can be uniquely represented as $A_1 0 A_2 0 \dots 0 A_n$, where the words A_i contain no zeros and some of these words are nonempty. Then we set $o(A) := \omega^{o(A_n^-)} + \dots + \omega^{o(A_2^-)} + \omega^{o(A_1^-)}$. The height of each A_i^- is less than that of A, so the inductive hypothesis applies.

Note that $o(\Lambda) = 0$ and $o(0A) = o(A) + 1$ for each word A. If $B \neq 0^k$, then

$$o(B0A) = o(A) + o(B), \tag{3.1}$$

and if $B \in S_1$ is nonempty, then $o(B) = \omega^{o(B^-)}$.

We will show that $o(A) > o(A[m])$ for each nonempty word A. We use induction on the height of A. For words of height 0 the inequality is obvious. Now we consider $A = A_1 0 A_2 0 \dots 0 A_k$, where $A_i \in S_1$ and some A_i are nonempty. Setting $C := A_2 0 \dots 0 A_k$ we obtain $A = A_1 0 C$, where A_1 and C cannot both be empty.

If $A_1 = \Lambda$, then $A = 0C$ and $A[m] = C$, so $o(A) > o(A[m])$.

Let $A_1 \neq \Lambda$. Then $A[m] = (A_1[m])0C$. The word $A_1[m]$ has the form 0^k only for $A_1 = 1$. In this case $o(A) = o(C) + \omega > o(C) + k + 1 = o(A[m])$ and the proof is complete. Otherwise, from (3.1) we obtain

$$o(A) = o(C) + o(A_1), \tag{3.2}$$

$$o(A[m]) = o(C) + o(A_1[m]). \tag{3.3}$$

Thus it is sufficient to show that $o(A_1) > o(A_1[m])$. We distinguish the following two cases.

1) $A_1 = 1B$ for some $B \in S_1$. Then $A_1[m] = (0B)^{m+1}$ and we have

$$o(A_1) = \omega^{o((1B)^-)} = \omega^{o(B^-)+1}.$$

On the other hand

$$o(A_1[m]) = \omega^{o(B^-)} \cdot (m+1) + 1 < \omega^{o(B^-)} \cdot \omega = o(A_1).$$

2) $A_1 = (n+1)B$ for some $n > 1$. Then $A_1^-[m] = (A_1[m])^-$ by the definition of the worm function. Now, by the inductive assumption

$$o(A_1) = \omega^{o(A_1^-)} > \omega^{o(A_1^-[m])} = \omega^{o((A_1[m])^-)} = o(A_1[m]),$$

as required.

§ 4. The system W_1

This string rewriting system corresponds to a nondeterministic version of the Worm principle. Let $\Sigma = \{a_0, a_1, a_2, \dots\}$ be an infinite alphabet and S_n the set of words in the sub-alphabet $\{a_i \mid i \geqslant n\}$. The system W_1 is defined by the following rules:

$$a_{n+1}A \to (a_nA)^k \quad \text{for each } A \in S_{n+1}, \quad k \geqslant 1. \tag{$*$}$$

We point out several distinctions between computations in the system W_1 and the worm sequence.

First, transformations can occur anywhere within the word, not only at its start. Second, the symbol a_0 is never erased. Thus the computation terminates if and only if the word consisting of the single letter a_0 appears. Third, k, the number of copies occurring after the application of the rule ($*$), is chosen arbitrarily at each step. Fourth, the sub-word $A \in S_{n+1}$ to which we apply ($*$) need not be the longest possible in general.

It is easy to see that the evolution of any worm B can be modelled by some computation branch in the system W_1. In fact, suppose we apply the rules ($*$) each time to the leftmost occurrence of a sub-word of the form $a_{n+1}A$, where $A \in S_{n+1}$ has the maximum possible length, and we take k to correspond with the step of the evolution of the worm B. It is easy to prove by induction that the words obtained from B in this way in the system W_1 are only different from the corresponding worms by a certain prefix of the form a_0^i. Hence if W_1 is terminating then the lifespan of each worm is finite.

Theorem 3. *The system W_1 is terminating.*

Proof. We shall use the same map $o\colon S_0 \longrightarrow \varepsilon_0$ taking words to ordinal numbers as before, that is, we set $o(a_0^k) = k$ and $o(A_1 a_0 A_2 a_0 \dots a_0 A_n) = \omega^{o(A_n^-)} + \dots + \omega^{o(A_1^-)}$, where all the A_i belong to S_1 and some of them are nonempty, and where B^- is obtained from $B \in S_1$ by replacing all the letters a_{m+1} by a_m.

Lemma 1. *Let $A \to_{W_1} B$ be an application of some rule of the system W_1. Then $o(A) \geqslant o(B)$. Moreover, if this rule is applied to the beginning of A, then $o(A) > o(B)$.*

Proof. To spare us extra calculations we use the standard interpretation of worms as modal formulas in the polymodal provability logic GLP or in its strictly positive fragment RC (see [8] and [16]). For brevity we will identify words $a_{i_1} a_{i_2} \dots a_{i_n}$ with formulas of the form $\langle i_1 \rangle \langle i_2 \rangle \dots \langle i_n \rangle \top$ in GLP. The notation $A \vdash B$ means that the implication $A \to B$ is provable in GLP.

It is known (see [8], Lemma 12) that for any words $A, B \in S_0$

$$o(A) < o(B) \quad \Longleftrightarrow \quad B \vdash a_0 A.$$

Hence, to prove the first statement it is sufficient to show that if $A \to_{W_1} B$, then $A \vdash B$. (In which case we have $B \nvdash a_0 A$ because $A \nvdash a_0 A$, so that $o(A) \not< o(B)$.)

Using induction on k we will show first that $a_{n+1} A C \vdash (a_n A)^k C$ for each $A \in S_{n+1}$. The basis of induction is an axiom of GLP. We assume that the formula holds for k and prove it for $k+1$. We observe that $a_{n+1} A C$ implies $a_{n+1} A \wedge (a_n A)^k C$. By Lemma 11, (ii) in [8] this formula is equivalent to $a_{n+1} A (a_n A)^k C$, which obviously implies $a_n A (a_n A)^k C$, as required.

It follows that $D a_{n+1} A C \vdash D (a_n A)^k C$ for each word D, which proves the first part of the lemma. The second part is a consequence of the following observation.

Lemma 2. *If B is a nonempty word, then $o(A) < o(BA)$.*

Proof. Using induction on the length of B in an obvious way we see that $BA \vdash a_0 A$.

Since $a_{n+1} A C \vdash (a_n A)^k C$ for each $k \geqslant 1$, it follows that

$$o(a_{n+1} A C) \geqslant o((a_n A)^{k+1} C) > o((a_n A)^k C).$$

Thus the second part of Lemma 1 is proved.

Suppose that W_1 is not terminating and $A_0 \to A_1 \to A_2 \to \cdots$ is an infinite sequence of transformations. Using Lemma 1 we can assume that $o(A_0) = o(A_1) = o(A_2) = \cdots$. Further, among all such sequences we can select one with the minimal ordinal $o(A_0)$. Since $o(A_0) = o(A_1) = o(A_2) = \cdots$, Lemma 1 shows that there are no transformations in this sequence at the beginning of words. Hence, there exists a nonempty B equal to the maximal common initial piece of all the A_i, so that $A_i = B A_i'$ for each $i \geqslant 0$. This means that all the transformations occur to the right of B, and hence $A_0' \to A_1' \to A_2' \to \cdots$ where $o(A_i') < o(A_i) = o(A_0)$ for each i. This contradicts the choice of the ordinal $o(A_0)$ to be minimal.

§ 5. The system W_2

The signature of the system W_2 contains a constant 0, a unary function symbol f and a binary multiplication symbol $\cdot$. The system W_2 has the following rules:

$$\begin{cases} (x\cdot y)\cdot z \to x\cdot(y\cdot z), \\ f(0\cdot x)\to (0\cdot f(x))^m, \quad m\geqslant 1, \\ f(0)\to 0^m, \quad m\geqslant 1. \end{cases}$$

Here x^m means $\underbrace{x\cdot(\cdots(x\cdot(x\cdot x))\cdots)}_{m\text{ times}}$.

Intuitively, f and $\cdot$ can be viewed as operations on words in the alphabet $\Sigma=\{a_0,a_1,a_2,\dots\}$, where 0 denotes a_0, $x\cdot y$ denotes the concatenation of the words x and y, and $f(x)$ denotes the word obtained by replacing every letter a_i in x by a_{i+1}. Thus, according to this interpretation any closed term t has a value, which is a nonempty word $A\in S_0$.

We can find A from t by calculating the f-depth of zeros. By the *f-depth* of an occurrence of a subterm s in a term t we mean the number of symbols f occurring on the path from the root of the subtree s to the root of the tree t (the starting vertex of the path is not counted). If the nth leftmost zero lies at f-depth k in t, then the nth symbol in A is equal to a_k.

We define a map of nonempty words $A\in S_0$ to closed terms $A^{\#}$ of the signature of W_2; the value of $A^{\#}$ will be equal to A. If $A=a_0^n$, then $A^{\#}:=0^n$. Otherwise $A=A_1a_0A_2a_0\dots a_0A_n$, where the A_i are words in S_1 not all of which are empty. Then

$$A^{\#}:=f((A_1^-)^{\#})\cdot 0\cdot f((A_2^-)^{\#})\cdot 0\cdots 0\cdot f((A_n^-)^{\#}),$$

where the factor $f((A_i^-)^{\#})$ is left out if the word A_i is empty. (We assume right association of brackets, so that $x\cdot y\cdot z$ is treated as $x\cdot(y\cdot z)$.)

Example 2. $(a_1a_2a_0a_1)^{\#}=f((a_0a_1)^{\#})\cdot(0\cdot f(a_0^{\#}))=f(0\cdot f(0))\cdot(0\cdot f(0))$.

As previously, $A[m]$ denotes the worm function.

Lemma 3. *If A is nonempty and does not begin with a_0, then $A^{\#}\to^*_{W_2} A[m]^{\#}$.*

Proof. Since A does not begin with a_0, $A^{\#}$ contains an occurrence of a subterm of the form $f(0\cdot t)$ or $f(0)$. Pick the leftmost of such occurrences and apply to it the second or the third rule with the given m. Then reassociate the brackets to the right using the first rule. We claim that the resulting term is $A[m]^{\#}$.

Now, if $C\in S_n$, then $C^{\#}=f^n(t)$ for some t. Thus, if B begins with a_n, then $B^{\#}$ begins with n symbols f: $B^{\#}=f(f(\cdots f(t)\cdots)\cdots)\cdots$, and the scope of the kth f corresponds to the maximal initial piece of B which belongs to S_k. Hence if $a_{n+1}B$ is the maximal initial piece of A which belongs to S_{n+1}, then there are two possibilities.

If B is empty, then $A^{\#}$ contains a subterm $f(0)$ (and this is the $(n+1)$st occurrence of the symbol f.) We replace it with 0^m in accordance with the third rule and normalize the whole term, associating the brackets to the right if necessary. These zeros are preceded by n symbols f and the f-depths of all the other zeros have remained the same. Hence the first symbol, a_{n+1}, in A has been replaced with $(a_n)^m$.

If B is nonempty, then $A^{\#}$ contains a subterm $f(0\cdot t)$ preceded by n symbols f. Here $t = C^{\#}$, where C is the result of subtracting n from the index of each letter in B. We replace $f(0\cdot t)$ by $(0\cdot f(t))^m$ following the second rule and normalize the term, applying the first rule if necessary. This corresponds to replacing the prefix $a_{n+1}B$ in A with $(a_nB)^m$.

This completes the proof.

Corollary 1. *The complexity function $C_{W_2}(n)$ is not bounded by a provably total computable function in* PA.

Proof. We will prove that for certain primitive recursive functions g and h (which are provably total in PA) we have $h(n, C_{W_2}(g(n))) \geqslant D(n)$, where $D(n)$ is the 'lifespan function' of a worm of size n. Let A be a worm of size n and let $g(n)$ be a function providing an upper bound on the length of the term $A^{\#}$. Clearly, g can be chosen to be primitive recursive. Now we consider the evolution of the worm A and extract the subsequence A_{i_k} of all the worms that do not begin with 0 from the sequence A_i.

Note that the transition from A_{i_k} to $A_{i_{k+1}}$ consists in applying one rule and then removing all the zeros at the beginning of the word. Hence the number of such steps is bounded by the length of the word A_{i_k+1}, and from the bound on the length of worms we obtain

$$i_{k+1} \leqslant i_k + 1 + n\cdot(i_k+2)!.$$

This yields a primitive recursive bound on i_k as a function of k and n (we denote this bound by $I(n,k)$). By Lemma 3 there exists a sequence of terms B_k in W_2 such that

$$B_0 \to^* B_1 \to^* B_2 \to^* \cdots \to^* B_k \to^* \cdots$$

and for each k the term B_k has the form $0^{m_k}\cdot A^{\#}_{i_k}$ for some $m_k \in \mathbb{N}$. Thus the lifespan of the worm A is bounded by

$$i_N + |A_{i_N}| \leqslant i_N + n(i_N+1)! \leqslant I(n,N) + n(I(n,N)+1)!,$$

where N is the length of the sequence of transformations B_k. Let $h(n,N)$ denote the right-hand side of this inequality. Since N is bounded by $C_{W_2}(g(n))$, we obtain the required result.

Lemma 4. *The system W_2 is terminating.*

Proof. We prove that W_2 is terminating using the lexicographic path ordering. We order the symbols of the signature as $0 \prec \cdot \prec f$ and show that all the three schemas of rules in W_2 reduce the term in the sense of the ordering $\prec_{\text{lpo}}$.

In fact, for the first rule we have $x\cdot y \succ_{\text{lpo}} y$, hence

$$(x\cdot y)\cdot z \succ_{\text{lpo}} y\cdot z.$$

Since $x\cdot y \succ_{\text{lpo}} x$, we infer that

$$(x\cdot y)\cdot z \succ_{\text{lpo}} x\cdot(y\cdot z).$$

The statement $f(0\cdot x) \succ_{\mathrm{lpo}} (0\cdot f(x))^m$ can be proved by induction on m. It holds for $m = 1$ because $f \succ \cdot$, $f(0\cdot x) \succ_{\mathrm{lpo}} 0$ and $f(0\cdot x) \succ_{\mathrm{lpo}} f(x)$. For $m+1$ we use the assumption $f \succ \cdot$, the inductive assumption for m and the fact that $f(0\cdot x) \succ_{\mathrm{lpo}} (0\cdot f(x))$, which has just been established.

In a similar way we can prove that $f(0) \succ_{\mathrm{lpo}} 0^m$ for each $m \geqslant 1$.

§ 6. The system W_3

We modify the system W_2 to obtain a slowly terminating term rewriting system with a finite set of rules. Now a term will not encode the worm alone, but also the step m of its evolution. We will arrange things so that the term encoding a pair (m, A) rewrites to the term encoding $(m+1, A[m])$.

We introduce new unary function symbols a, b, c and d playing the role of markers. A pair (m, A) will be represented by a term of the form $da^m(t)$, where $t = A^{\#}$ (as in W_2) and d always marks the beginning of the term. The rules of W_3 will enable the marker a to move to some place within t, where transformations are to be made. Each symbol a performs one act of copying the appropriate piece of the worm and is replaced by b. Further transformations follow the rules of W_2. We call them reductions, and they result in a reduction in the ordinal of the corresponding W_2-term. After one reduction a new symbol c appears. Symbols c can travel to the beginning of the term; all the symbols b encountered in the process are transformed into c, while the symbols c themselves are transformed into symbols a upon encountering d. The latter event triggers the start of a new cycle of computations of the next step of the worm evolution.

A notable feature of W_3 is that in fact the order of transformations can be somewhat different from the one described above. So, the proof that W_3 terminates requires a thorough analysis.

The rules of the system W_3 are as follows:

1) $(x\cdot y)\cdot z \longrightarrow x\cdot(y\cdot z)$;

2) (copying)

$$a(f(0\cdot x)) \longrightarrow b\bigl(f(0\cdot x)\cdot(0\cdot f(x))\bigr), \qquad a(f(0)) \longrightarrow b(f(0)\cdot 0);$$

3) (reduction)

$$f(0\cdot x) \longrightarrow c(0\cdot f(x)), \qquad f(0) \longrightarrow c(0), \qquad 0\cdot x \longrightarrow c(x);$$

4) (a moves downwards)

$$a(f(x)) \longrightarrow f(a(x)), \qquad a(x\cdot y) \longrightarrow a(x)\cdot y,$$
$$a(b(x)) \longrightarrow b(a(x)), \qquad a(c(x)) \longrightarrow c(a(x));$$

5) (c moves upwards and subsumes b)

$$f(c(x)) \longrightarrow c(f(x)), \qquad c(x)\cdot y \longrightarrow c(x\cdot y), \qquad b(c(x)) \longrightarrow c(c(x));$$

6) $d(c(x)) \longrightarrow d(a(x))$.

First we show that W_3 simulates the evolution of any worm. For brevity we shall write $ab(t)$ in place of $a(b(t))$ and $a^n(t)$ in place of $a(a(\cdots a(t)\cdots))$.

Lemma 5. *For each $n \geqslant 0$:*
(i) $a^n f(0\cdot x) \to^*_{W_3} c^{n+1}(0\cdot f(x))^{n+1}$;
(ii) $a^n f(0) \to^*_{W_3} c^{n+1}(0^{n+1})$.

Proof. We will prove (i); assertion (ii) has a similar, but simpler, proof. We consider the derivation when the symbols a move one by one to the occurrence of $f(0\cdot x)$, change to b and produce copies of the term $(0\cdot f(x))$ on the right. Then the reduction rule is applied, producing a symbol c. This symbol moves upwards, subsuming n symbols b, and then the brackets are reassociated to the right:

$$\begin{aligned} a^n f(0\cdot x) &\longrightarrow a^{n-1}b(f(0\cdot x)\cdot(0\cdot f(x))) \longrightarrow^* a^{n-2}b\big(a(f(0\cdot x))\cdot(0\cdot f(x))\big)\\ &\longrightarrow a^{n-2}b\big(b(f(0\cdot x)\cdot(0\cdot f(x)))\cdot(0\cdot f(x))\big)\\ &\longrightarrow^* b\big(\cdots b(b(f(0\cdot x)\cdot(0\cdot f(x)))\cdot(0\cdot f(x))\cdots(0\cdot f(x)))\big)\\ &\longrightarrow b\big(\cdots b(b(c(0\cdot f(x))\cdot(0\cdot f(x)))\cdot(0\cdot f(x))\cdots(0\cdot f(x)))\big)\\ &\longrightarrow^* c^{n+1}(0\cdot f(x))^{n+1}. \end{aligned}$$

The proof is complete.

The following lemma shows that W_3 simulates one step in the worm sequence.

Lemma 6. *Let A be a nonempty word distinct from a_0. Then for each $n \geqslant 1$*

$$da^n(A^{\#}) \longrightarrow^*_{W_3} da^{n+1}(A[n]^{\#}).$$

Proof. Two cases are possible here.
Case 1: A begins with a_0, so that $A^{\#} = 0\cdot t$. Then

$$da^n(0\cdot t) \longrightarrow da^n c(t) \longrightarrow^* dca^n(t) \longrightarrow^* da^{n+1}(t).$$

Case 2: A does not begin with a_0. In this case the leftmost occurrence of 0 in the term $A^{\#}$ occurs in the context $f(0\cdot t)$ or $f(0)$. Then the term $A[n]^{\#}$ is obtained from $A^{\#}$ by replacing these occurrences with $(0\cdot f(t))^{n+1}$ or 0^{n+1}, respectively. Without loss of generality we look at the first case.

We move the symbols a rightwards to the first occurrence of $f(0\cdot t)$. Using Lemma 5 we can transform the occurrence of $a^n f(0\cdot t)$ into the term $c^{n+1}(0\cdot f(t))^{n+1}$. Next we move c^{n+1} to the beginning of the term, obtaining $dc^{n+1}(A[n]^{\#})$. Finally, from $dc^{n+1}(A[n]^{\#})$ we obtain $da^{n+1}(A[n]^{\#})$.

Corollary 2. *The complexity function $C_{W_3}(n)$ is not bounded by any provable total computable function in* PA.

Proof. We consider the evolution of the worm $A = A_0$:

$$A_0 \to A_1 \to A_2 \to \cdots \to A_k \to \cdots .$$

From Lemma 6 we obtain a derivation in W_3:

$$daA_0^{\#} \longrightarrow^* da^2A_1^{\#} \longrightarrow^* da^3A_2^{\#} \longrightarrow^* \cdots \longrightarrow^* da^{k+1}A_k^{\#} \longrightarrow^* \cdots .$$

This can be continued until the last step but one in the evolution of A (when the word a_0 appears). Each step of this evolution can be simulated by at least one application of the rules of W_3. Hence, the length of the derivation in W_3 is no smaller than the length of the worm evolution (minus one step).

Theorem 4. *The system W_3 terminates.*

Proof. It is sufficient to show that there can be no infinite chains of transformations of closed terms in W_3. In fact, from each derivation in W_3 we obtain a derivation of the same length by replacing all the occurrences of variables with 0.

Now we prove several auxiliary results.

Lemma 7. *If a term t does not contain a symbol d, then there exists no infinite chain of transformations starting with t.*

Proof. Note first that rule 6) is not used in derivations from the term t because the symbol d cannot occur there. Now we order the symbols in the original alphabet: $0 \prec c \prec \cdot \prec f \prec b \prec a$. A direct verification shows that all the rules of W_3, with the exception of rule 6), reduce the term in the sense of the corresponding lexicographic path ordering. Hence each chain of transformations following these rules terminates.

Next we consider the special case when the initial term has the form $d(t)$, where t does not contain a symbol d or variables. In this case the other terms in the chain of transformations have the same form. The first (leftmost) occurrence of zero in such a term will be called *critical.* We will represent terms as upwards growing trees. By the *trunk* of a tree we will mean the path from the root to the critical leaf of the tree, that is, the leftmost path in the tree. By *critical reductions* (*copyings*) we shall mean reduction (copying) transformations applied to the critical occurrence of 0. The following lemma plays a key part.

Lemma 8. *Each infinite sequence of transformations of the form $dt_0 \to dt_1 \to dt_2 \to \cdots$ in the system W_3, where the t_i are closed terms not containing d, involves a critical reduction.*

Proof. Suppose that an infinite chain of transformations involves no critical reduction. Then we can make the following observations.

1. Let n_a denote the number of letters a in the trunk of the tree (we also introduce similar notation for b and c). Then the sum $n_a + n_b + n_c$ is the same for all the terms in the derivation.

Note that transformations in groups 2) and 4)–6) have the following property: no letters a, b or c positioned to the right of the trunk of the term will end on the trunk (a can only move upwards, while c can move downwards, but only along the left-hand branch). The occurrences of a, b and c already positioned on the trunk remain on it or are renamed in the order $a \to b \to c \to a$, so that the sum $n_a+n_b+n_c$ does not change. Transformation 1) can only cancel out one multiplication sign on the trunk. Reduction transformations 3) can change labels on the trunk only when we apply them to the critical occurrence; otherwise the symbol c that appears can never reach the trunk.

2. The number of critical copyings in the chain of transformations in question is finite.

Let m_b denote the number of symbols b on the trunk positioned under some symbol c there. Then the quantity $n_a + m_b + n_c$ is nonincreasing and, moreover, decreases by 1 after each critical copying: the symbols b arising after critical copyings occur above all the symbols c on the trunk, and should be subtracted from the sum.

3. If a chain of transformations contains neither critical copyings nor reductions then the quantity n_a is nondecreasing. Since the sum $n_a + n_b + n_c$ is constant, starting from some step n_a stabilizes. This means that rule 6) is not used in the subsequent transformations. But in this case we known already from Lemma 7 that the chain of transformations terminates.

Now we define a map $\overline{v}$ from the set of closed terms which do not contain d in W_3 to words in the alphabet $\Sigma = \{a_0, a_1, \dots\}$. This map is similar to the interpretation of closed terms in W_2 as words, but is a modification of the latter, which takes into account the position of some symbols c in the terms under consideration.

We extend the alphabet Σ by an infinite series of (new) symbols c_i, so that we set $\Sigma' = \Sigma \cup \{c_0, c_1, c_2, \dots\}$. For words A in the alphabet Σ' we have the map $A \mapsto A^+$ increasing the index of each letter by 1. For each $n \geqslant 0$, let S'_n be the set of words in the alphabet $\{a_i, c_i : i \geqslant n\}$. As previously, S_n denotes the set of words in the alphabet $\{a_i : i \geqslant n\}$.

Let t be a closed term in W_3 which does not contain d. We assign to t a word $v(t)$ in the alphabet Σ' recursively as follows:

$$\begin{gathered} v(0) = a_0, \qquad v(f(t)) = v(t)^+, \qquad v(t \cdot s) = v(t)v(s), \\ v(c(t)) = c_0 v(t), \qquad v(at) = v(bt) = v(t). \end{gathered} \tag{6.1}$$

Now let $A = A'a_i \in S_0$. We set $A \circ c_j = A$ if $i \leqslant j$ and $A \circ c_j = Aa_j$ if $i > j$. We also set $A \circ c_j = \Lambda$ if A is empty. We define a map $w \colon S'_0 \to S_0$ as follows:

$$w(\Lambda) = \Lambda, \qquad w(Aa_i) = w(A)a_i, \qquad w(Ac_i) = w(A) \circ c_i.$$

Finally, we set $\overline{v}(t) = w(v(t))$ and $\overline{o}(t) = o(\overline{v}(t))$, where o is the ordinal of a word in the system W_1.

Note that for any words $A, B \in S'_0$ we have $w(Aa_iB) = w(A)w(a_iB)$. This can easily be proved by induction on the length of B. We shall often use this observation without special mention in the proof of the next lemma.

Lemma 9. *Let t_1 and t_2 be closed terms not containing d. If $dt_1 \to_{W_3} dt_2$, then $\overline{o}(dt_2) \leqslant \overline{o}(dt_1)$. If $t_1 \to_{W_3} t_2$ is a critical reduction, then*

$$\overline{o}(dt_1) < \overline{o}(dt_2).$$

Proof. First we note that for any term t the map v assigns a nonempty subword A of the word $v(t) = CAD$ to each subterm s of t. If the occurrence of s in t has f-depth n, then A is obtained from $v(s)$ by n applications of the function $(\,\cdot\,)^+$. The rightmost symbol in A corresponds to the rightmost occurrence of 0 in s, so that it is some a_i. Hence the word $\overline{v}(t) = w(CAD)$ has the form $w(C)A'D'$ for some words A' and D' which can be obtained by cancelling out and renaming some of the symbols c_j occurring in A and D, respectively. (Generally speaking, A' and D' do not necessarily coincide with $w(A)$ and $w(D)$.)

To prove the first part of the lemma, as in Lemma 1, it suffices to show that $\overline{v}(t_1) \vdash \overline{v}(t_2)$. We examine the applications of all the rules of W_3.

1. Using rules 1), 4) or 6) does not change the word $v(t_1)$, and therefore does not change $\overline{v}(t_1)$.

2. *The rule* $af(0) \to b(f(0) \cdot 0)$. Let $v(t_1) = Ca_{n+1}D$; then $v(t_2) = Ca_{n+1}a_nD$. Using induction on the length of D we shall show that

$$w(a_{n+1}D) \vdash w(a_{n+1}a_nD),$$

which yields $w(C)w(a_{n+1}D) \vdash w(C)w(a_{n+1}a_nD)$ and proves the required result. The basis of induction reduces to the obvious $a_{n+1} \vdash a_{n+1}a_n$. Now we discuss the induction step.

If $D = a_iD_1$, then $w(a_{n+1}D) = a_{n+1}w(D)$, and therefore

$$w(a_{n+1}D) \vdash a_{n+1} \wedge a_nw(D) \vdash a_{n+1}a_nw(D) = w(a_{n+1}a_nD).$$

If $D = c_iD_1$, then we distinguish the following cases.

If $i > n$, then by the inductive assumption

$$w(a_{n+1}D) = w(a_{n+1}D_1) \vdash w(a_{n+1}a_nD_1) = w(a_{n+1}a_nD).$$

If $i = n$, then

$$w(a_{n+1}D) = a_{n+1}w(a_nD_1) = a_{n+1}w(a_nc_nD_1) = w(a_{n+1}a_nD).$$

If $i < n$, then

$$w(a_{n+1}D) = a_{n+1}w(a_iD_1) \vdash a_{n+1} \wedge a_nw(a_iD_1) \vdash a_{n+1}a_nw(a_iD_1) = w(a_{n+1}a_nD).$$

3. *The rule* $a(f(0 \cdot x)) \to b(f(0 \cdot x) \cdot (0 \cdot f(x)))$. In this case $v(t_1)$ has the form $Ca_{n+1}AD$, where $A \in S'_{n+1}$ ends with some a_i, and $v(t_2) = Ca_{n+1}Aa_nAD$. We have $w(a_{n+1}AD) = a_{n+1}A'D'$ for some $A' \in S_{n+1}$ and D' and $w(a_{n+1}Aa_nAD) = a_{n+1}A'a_nA'D'$ for the same A' and D'. Here A' in the second equality is the same as in the first because A contains no symbols c_j with $j \leqslant n$. In a similar way, D' in the second word is the same as in the first because A' ends with the same a_i as A.

As in Lemma 1 above, we obtain

$$a_{n+1}A'D' \vdash a_{n+1}A' \wedge a_nA'D' \vdash a_{n+1}A'a_nA'D',$$

which proves the required result.

4. *The rule* $f(0 \cdot x) \to c(0 \cdot f(x))$. In this case $v(t_1)$ has the form $Ca_{n+1}AD$, where $A \in S'_{n+1}$, and $v(t_2) = Cc_na_nAD$. Then

$$w(Ca_{n+1}AD) = w(C)a_{n+1}A'D'$$

for some $A' \in S_{n+1}$ and D'. In turn, $w(Cc_na_nAD)$ coincides with $w(C)a_na_nA'D'$ (if c_n is not cancelled out) and with $w(C)a_nA'D'$ (otherwise).

We obtain $a_{n+1}A'D' \vdash a_na_nA'D' \vdash a_nA'D'$, which proves the required result in either case.

Note also that when the application of this rule is critical, the word C is empty and c_n is cancelled out, so $\overline{v}(t_1) \vdash a_n\overline{v}(t_2)$, and therefore $\overline{o}(t_1) > \overline{o}(t_2)$.

5. *The rule* $f(0) \to c(0)$. In this case $v(t_1)$ has the form $Ca_{n+1}D$ and $v(t_2) = Cc_n a_n D$. This case is similar to the previous one.

6. *The rule* $0 \cdot x \to c(x)$. In this case $v(t_1)$ has the form $Ca_n AD$ and $v(t_2) = Cc_n AD$. Let $w(C) = C'a_i$. If $n < i$, then $w(Cc_n AD) = w(Ca_n AD)$, and there is nothing to prove. If $n \geqslant i$, then $w(Cc_n AD) = C'w(a_i A)D'$ and $w(Ca_n AD) = C'w(a_n A)D'$. Here D' is the same in both cases because the word A ends with some letter a_k, which is not cancelled out in $w(a_n A)$. Now the required result follows from the lemma below.

Lemma 10. *For each $n \geqslant i$ and any words A and D', $w(a_n A)D' \vdash w(a_i A)D'$.*

Proof. Our argument uses induction on the length of A. If A is the empty word, then the result is obvious.

If $A = a_j A'$, then

$$w(a_n A)D' = a_n w(a_j A')D' \vdash a_i w(a_j A')D' = w(a_i A)D'.$$

If $A = c_j A'$, then we consider three cases.

If $j < i$, then

$$w(a_n A)D' = a_n w(a_j A')D' \vdash a_i w(a_j A')D' = w(a_i A)D'.$$

If $j \geqslant n$, then by the inductive hypothesis

$$w(a_n A)D' = w(a_n A')D' \vdash w(a_i A')D' = w(a_i A)D'.$$

If $i \leqslant j < n$, then by the inductive hypothesis

$$w(a_n A)D' = a_n w(a_j A')D' \vdash w(a_j A')D' \vdash w(a_i A')D' = w(a_i A)D'.$$

The proof is complete.

Note also that when the application of this rule is critical, $\overline{v}(t_1) = a_0 \overline{v}(t_2)$, so that $\overline{o}(t_1) > \overline{o}(t_2)$.

7. *The rule* $f(c(x)) \to c(f(x))$. In this case $v(t_1)$ has the form $Cc_{n+1}AD$, where $A \in S'_{n+1}$, and $v(t_2) = Cc_n AD$. Let $w(C) = C'a_i$; then $w(Cc_{n+1}AD)$ has the form $C'w(a_i c_{n+1} A)D'$ and $w(Cc_n AD) = C'w(a_i c_n A)D'$. It is sufficient to show that

$$w(a_i c_{n+1} A)D' \vdash w(a_i c_n A)D'.$$

We examine three cases. If $i \leqslant n$, then

$$w(a_i c_{n+1} A)D' = w(a_i A)D' = w(a_i c_n A)D'.$$

If $i > n+1$, then by Lemma 10

$$w(a_i c_{n+1} A)D' = a_i w(a_{n+1} A)D' \vdash a_i w(a_n A)D'.$$

If $i = n+1$, then by Lemma 10

$$w(a_i A)D' \vdash w(a_n A)D',$$

and therefore

$$w(a_ic_{n+1}A)D' = w(a_iA)D' \vdash a_i \wedge w(a_nA)D' \vdash a_iw(a_nA)D' = w(a_ic_nA)D'.$$

8. *The rule* $c(x)\cdot y \to c(x\cdot y)$. This does not change $v(t_1)$.

9. *The rule* $b(c(x)) \to c(c(x))$. In this case $v(t_1)$ has the form Cc_nAD, where $A \in S'_n$, and $v(t_2) = Cc_nc_nAD$. Let $w(C) = C'a_i$; then $w(Cc_nAD)$ has the form $C'w(a_ic_nA)D'$ and $w(Cc_nc_nAD) = C'w(a_ic_nc_nA)D'$. It is sufficient to show that

$$w(a_ic_nA)D' \vdash w(a_ic_nc_nA)D'.$$

If $i > n$, then

$$w(a_ic_nA)D' = w(a_ia_nA)D' \vdash a_i \wedge a_nw(a_nA)D' \vdash a_ia_nw(a_nA)D' = w(a_ic_nc_nA)D'.$$

If $i \leqslant n$, then

$$w(a_ic_nA)D' = w(a_iA)D' = w(a_ic_nc_nA)D'.$$

Lemma 9 is proved.

From Lemmas 8 and 9 we obtain the following.

Corollary 3. *Assume that d does not occur in t. Then there is no infinite chain of transformations in W_3 starting with the term dt.*

Remark 1. The involved definitions of the functions $\overline{v}$ and $\overline{o}$ and the complicated proof of Lemma 9 are related to noncritical applications of the zero reduction rule $0\cdot x \to c(x)$. The example $f(0)\cdot(0\cdot f(0)) \to f(0)\cdot f(0)$ shows that converting terms in W_3 into terms in the system W_2 by forgetting all the auxiliary symbols (including c) we can increase the ordinals of the corresponding words. There are no similar reduction rules in W_2 and W_1, so this problem does not arise.

Now we complete the proof of Theorem 4. Using induction on the number of occurrences of symbols d in an arbitrary term s we show that there can be no infinite chain of transformations of the form $s = s_0 \to s_1 \to \cdots$. Without loss of generality assume that s is a closed term. We have already dealt with the case when s does not contain d.

We look at the deepest occurrence of d in s, that is, we represent s in the form $s = u[d(t)]$, where t does not contain d. Note that any transformation of s using the rules of W_3 either occurs in the subterm $d(t)$ or preserves this subterm. In the second case the transformation $s \to s_1$ reduces to transforming the context $u[x] \to u_1$ using the same rule and then replacing all the occurrences of x in u_1 with $d(t)$. (There can be more than one such occurrence of x in u_1.) Hence to each term s_i we can connect a term u_i and a finite string of terms of the form $dt_{i1}, dt_{i2}, \dots, dt_{ik_i}$ such that s_i is obtained by replacing the consecutive occurrences of x in u_i with dt_{ij} (the terms t_{ij} do not contain d). Here the sequence of terms u_i can be produced using the same rules as were used to produce the s_i, excluding their application to the distinguished terms dt_{ij}, which do not affect the u_i.

Since $u = u_0$ contains fewer occurrences of d than s, it follows from the inductive assumption that starting from some step n the term u_i does not change. Hence for $i \geqslant n$ each transformation proceeds within one of the terms $dt_{i1}, dt_{i2}, \dots, dt_{ik}$, where $k = k_n$ is fixed. By Corollary 3 the chain of transformations in each term dt_{ij} terminates, so the whole sequence also terminates.

The authors are indebted to M. R. Pentus, who found several flaws in the first version of this paper.

Bibliography

[1] J. W. Klop, "Term rewriting systems", *Handbook of logic in computer science*, vol. 2, Handb. Log. Comput. Sci., vol. 2, Oxford Univ. Press, New York 1992, pp. 1–116.

[2] M. Bezem, J. W. Klop and R. de Vrijer (eds.), *Term rewriting systems. Terese*, Cambridge Tracts Theoret. Comput. Sci., vol. 55, Cambridge Univ. Press, Cambridge 2003, xxii+884 pp.

[3] H. E. Rose, *Subrecursion. Functions and hierarchies*, Oxford Logic Guides, vol. 9, The Clarendon Press, Oxford Univ. Press, New York 1984, xiii+191 pp.

[4] L. Kirby and J. Paris, "Accessible independence results for Peano arithmetic", *Bull. London Math. Soc.* **14**:4 (1982), 285–293.

[5] N. Dershowitz and J.-P. Jouannaud, "Rewrite systems", *Handbook of theoretical computer science*, vol. B, Elsevier, Amsterdam 1990, pp. 243–320.

[6] N. Dershowitz and G. Moser, "The hydra battle revisited", *Rewriting computation and proof*, Lecture Notes in Comput. Sci., vol. 4600, Springer, Berlin 2007, pp. 1–27.

[7] H. Touzet, "Encoding the hydra battle as a rewrite system", *Mathematical foundations of computer science* 1998 (Brno), Lecture Notes in Comput. Sci., vol. 1450, Springer, Berlin 1998, pp. 267–276.

[8] L. D. Beklemishev, "The Worm principle", *Logic Colloquium* '02, Lect. Notes Log., vol. 27, Assoc. Symbol. Logic, La Jolla 2006, pp. 75–95.

[9] M. Hamano and M. Okada, "A relationship among Gentzen's proof-reduction, Kirby-Paris' Hydra game, and Buchholz's Hydra game", *Math. Logic Quart.* **43**:1 (1997), 103–120.

[10] L. D. Beklemishev, "Representing worms as a term rewriting system", Mini-workshop: Logic, combinatorics and independence results, Report No. 52/2006, Mathematisches Forschungsinstitut Oberwolfach, *Oberwolfach Rep.* **3**:4 (2006), 3093–3095.

[11] W. Buchholz, "Another rewrite system for the standard hydra battle", Mini-workshop: Logic, combinatorics and independence results, Report No. 52/2006, *Oberwolfach Rep.* **3**:4 (2006), 3099–3101.

[12] H. Zankl, S. Winkler and A. Middeldorp, "Beyond polynomials and Peano arithmetic — automation of elementary and ordinal interpretations", *J. Symbolic Comput.* **69** (2015), 129–158.

[13] F. Baader and T. Nipkow, *Term rewriting and all that*, Cambridge Univ. Press, Cambridge 1998, xii+301 pp.

[14] J. H. Gallier, "What's so special about Kruskal's theorem and the ordinal Γ_0? A survey of some results in proof theory", *Ann. Pure Appl. Logic* **53**:3 (1991), 199–260.

[15] L. D. Beklemishev, "Reflection principles and provability algebras in formal arithmetic", *Uspekhi Mat. Nauk* **60**:2(362) (2005), 3–78; English transl. in *Russian Math. Surveys* **60**:2 (2005), 197–268.

[16] L. Beklemishev, "Calibrating provability logic: from modal logic to reflection calculus", Advances in modal logic, vol. 9, Coll. Publ., London 2012, pp. 89–94.

Lev D. Beklemishev
Steklov Mathematical Institute,
Russian Academy of Sciences, Moscow
E-mail: bekl@mi.ras.ru

Anastasija A. Onoprienko
Faculty of Mechanics and Mathematics,
Moscow State University
E-mail: ansidiana@yandex.ru

Received 25/MAR/15 and 21/JUN/15
Translated by IPS(DoM)