A TERM-REWRITING CHARACTERIZATION OF PSPACE

N. Eguchi

Graduate School of Engineering, Kobe University,
Rokko-dai, Nada-ku, Kobe 657-8501, Japan
E-mail: eguchi@kurt.scitec.kobe-u.ac.jp

Isabel Oitavem has introduced a term rewriting system (TRS) which captures the class **FPS** of polynomial-space computable functions. We propose an alternative TRS for **FPS**. As a consequence, it is obtained that **FPS** is the smallest class containing certain initial functions and closed under specific operations. It turns out that our characterization is relatively simple and suggests an uniform approach to the space-complexity.

Keywords: Term rewriting; Space complexity; PSPACE; Safe nested recursion.

Introduction

Term rewriting is known as an abstract model of computation, since a term is rewritten by successively replacing subterms by equal terms until no further reduction is possible. It is also known that term rewriting forms a Turing complete model of computation. This paper is an application of term rewriting to small complexity classes which only involve feasibly computable functions.

The immediate motivation has come from a work by I. Oitavem. In 1, a term rewriting characterization of the class **FPS** of functions computed in polynomial space was given. This is based on a recursion-theoretic characterization of **FPS** which was obtained by herself. In 2 Oitavem reformulates **FPS** according to a principle which was initiated by S. Bellantoni and S. Cook. The Bellantoni-Cook principle separates variables in every function by semi-colon as follows: $f(\vec{x};\vec{y})$ – variables $\vec{x}$ occurring to the left of the semi-colon are called normal, while variables $\vec{y}$ to the right are called safe. Roughly speaking, this principle allows recursion only for normal positions (safe recursion), while composition only for safe positions (safe composition). In 3 a recursion-theoretic characterization of the polynomial-time computable functions is given with the use of safe recursion on notation.

Oitavem has shown that it is possible to characterize classes of computational complexity involving space constraints by techniques from the field of term rewriting. Oitavem's function class, however, contains a initial function with large growth rate like the product, and two recursion schemes.

In this paper we introduce a new term rewriting system for **FPS**. As a consequence, an alternative characterization of **FPS** is obtained. This system is also based on the Bellantoni-Cook principle. Nevertheless, our class need only elementary initial functions and one recursion scheme which is called safe nested recursion. Hence our characterization is relatively simple. It also turns out that specific safe recursion schemes capture various space-complexity classes in the presence of the same initial functions (see Remark 1.8). In this sense, our formulation is uniform to the space complexity.

One problem is that translating the safe nested recursion scheme into the rewriting rule results in handling terms of exponential size. This is not admissible to capture **FPS**. We solve this problem by employing the *innermost strategy*. Furthermore, for the converse direction, it is not trivial whether the poly-space computations can be simulated in the present formulation. A main task is to define a pairing and unpairing functions.

The scheme of safe nested recursion has been introduced by T. Arai and the author to characterize the exponential-time computable functions. This paper is not fully self-contained, but owes observations on some properties of safe nested recursion to 4.

In Section 1 we define a function class $\mathbf{N}$ via the operation of safe nested recursion. The main definition is given in Section 2. Based on $\mathbf{N}$, we introduce a rewrite system $R_{\mathbf{N}}$ over a class $\mathcal{F}$ of function symbols. It is shown that for any $f \in \mathcal{F}$, the length of every term occurring in a rewriting sequence which starts with $f(\vec{t})$ is bounded by a polynomial in the lengths of input terms $\vec{t}$ if an innermost strategy is used. As a corollary, we have $\mathbf{N} \subseteq \mathbf{FPS}$. In Section 3, conversely, it is shown that every polyspace computation is simulated in $\mathbf{N}$.

1. A function class N with safe nested recursion

The first section is devoted to introduce a class $\mathbf{N}$ via the operation of safe nested recursion on notation (SNRN). The scheme of SNRN is introduced by Arai and the author.[4]

Notation 1.1. Although the functions in $\mathbf{N}$ are defined over the natural numbers, numbers are denoted as binary strings. For instance, $2x + i$ is denoted by xi for each $i \in \{0, 1\}$. Similarly, xy denotes the concatenation

of numbers x and y in the binary representation.

Fix the signature $\Sigma = \{0, 1, \mathrm{Z}\}$ and put $\Sigma^k := \{\sigma_1 \cdots \sigma_k : \sigma_1, \ldots, \sigma_k \in \Sigma\} \setminus \{\mathrm{Z} \cdots \mathrm{Z}\}$. If $w \in \Sigma^k$ is of the form $\sigma_1 \cdots \sigma_k$, then $w(i)$ denotes σ_i for every $i = 1, \ldots, k$. Numerals are represented by the binary successor $C_0, C_1 \in \mathbf{N}$ such that $C_0(x;) = x0$ and $C_1(x;) = x1$. Extending this notation to $\Sigma = \{0, 1, \mathrm{Z}\}$, we mean $C_{\mathrm{Z}}(x;)$ for 0, and $C_w(\vec{y};)$ abbreviates $(C_{w(1)}(y_1;), \ldots, C_{w(k)}(y_k;))$. For $\vec{y} = (y_1, \ldots, y_k)$, $w \in \Sigma^k$ and $i \in \{k + 1, \ldots, k + k\}$, $C_{w(i)}(y_i;)$ denotes y_{i-k} if $w(i - k) \in \{0, 1\}$, otherwise 0.

Let $f(\vec{x}, z)[g(\vec{y})/z]$ denote $f(\vec{x}, g(\vec{y}))$, the result of a substitution. $|x|$ denotes the length of the binary representation of a number x, i.e., $|x| = \lceil \log_2(x + 1) \rceil$, which is called the binary length of x. Moreover, for $\vec{x} = (x_1, \ldots, x_k)$, let $|\vec{x}| := (|x_1|, \ldots, |x_k|)$ and $\max \vec{x} := \max\{x_i : i = 1, \ldots, k\}$.

The computation for every function defined by nested recursion runs along the lexicographic ordering. The scheme of SNRN is defined via a lexicographic ordering $\prec$ and the $\prec$-functions.

Definition 1.2 ($\prec$-predecessors). Suppose $k \geq 1$ and $w \in \Sigma^k$. For $\vec{v} = (v_1, \ldots, v_k)$ and $\vec{y} = (y_1, \ldots, y_k)$, we define $\vec{v} \prec^k \vec{y}$.

(1) The case $k = 1$ is defined by $v \prec^1 C_i(v;)$ for an $i \in \{0, 1\}$. We write $v \preceq^1 y$ if $v \prec^1 y$ or $v = y$.
(2) For the case $k > 1$, $(v_1, \ldots, v_k) \prec^k (y_1, \ldots, y_k)$ iff there exists $k_0 \in \{1, \ldots, k\}$ such that $\forall i < k_0 (v_i = y_i)$, $v_{k_0} \prec^1 y_{k_0}$ and $\forall i > k_0 \exists j \in \{1, \ldots, k\} (v_i \preceq^1 y_j)$.

If $\vec{v} \prec^k \vec{y}$, then $\vec{v}$ is called a $\prec^k$-predecessor of $\vec{y}$.

Note. The ordering $\prec$ is a natural restriction of the usual lexicographic ordering in the Bellantoni-Cook principle. To see this, for now modify $\prec^2$ as

- $(x, y) \prec^2 (x + 1, 0)$ if $y \in \{x, x + 1, 0, 1\}$,
- $(x + 1, y) \prec^2 (x + 1, y + 1)$, and
- $(x, v) \prec^2 (x + 1, y + 1)$ if $v \in \{x, x + 1, y, y + 1\}$.

Let us consider Ackermann function here. Ackermann function $A(x, y)$ is defined by nested recursion on (x, y): $A(0, y) = y + 1$, $A(x + 1, 0) = A(x, 1)$, $A(x + 1, y + 1) = A(x, A(x + 1, y))$. In the equations, $(x, 1) \prec^2 (x + 1, 0)$ and $(x + 1, y) \prec^2 (x + 1, y + 1)$, but $(x, A(x + 1, y)) \not\prec^2 (x + 1, y + 1)$ due to the presence of $A(x + 1, y)$. Recall that the Bellantoni-Cook principle forbids substituting the recursion terms into the recursion parameters. These

observations give rise to the definition of a weaker lexicographic ordering $\prec$.

To determine an operation of SNRN, the $\prec$-functions are introduced. Recall that $C_Z(y;)$ denotes 0. Every $\prec^k$-function $\mathbf{f}$ indicates which $\prec^k$-predecessor of $C_w(\vec{y};) = (C_{w(1)}(y_1;), \ldots, C_{w(k)}(y_k;))$ should be chosen for each $w \in \Sigma^k$.

Definition 1.3 ($\prec$-functions). Suppose that $\mathbf{f}$ is a finite functions such that $\mathbf{f} : \{1, \ldots, k\} \times \Sigma^k \to \{1, \ldots, 2k\}$. For $w \in \Sigma^k$, let $C_{w(\mathbf{f}(w))}(\vec{y}_{\mathbf{f}(w)};)$ abbreviate $(C_{w(\mathbf{f}(1,w))}(y_{\mathbf{f}(1,w)});), \ldots, C_{w(\mathbf{f}(k,w))}(y_{\mathbf{f}(k,w)});))$.

Then $\mathbf{f}$ is called a $\prec^k$-function, if $C_{w(\mathbf{f}(w))}(\vec{y}_{\mathbf{f}(w)};) \prec^k C_w(\vec{y};)$ for all $w \in \Sigma^k$ (not depending on choice of $\vec{y} = (y_1, \ldots, y_k)$).

Note. Let us recall a convention in Notation 1.1 that for $k_0 \in \{1, \ldots, k\}$ and $w \in \Sigma^k$, $C_{w(k+k_0)}(y_{k+k_0};)$ denotes y_{k_0} if $w(k_0) \in \{0, 1\}$. By the definition, any $\prec^k$-function $\mathbf{f}$ satisfies the condition: $\forall w \in \Sigma^k$, $\exists k_0 \in \{1, \ldots, k\}$ s.t. $w(k_0) \in \{0, 1\}$ and

$$\begin{cases} \mathbf{f}(i, w) = i & \text{if } i < k_0, \\ \mathbf{f}(k_0, w) = k + k_0, & \\ \mathbf{f}(i, w) \in \{1, \ldots, 2k\} & \text{if } i > k_0. \end{cases}$$

For examples of $\prec^k$-functions, see 4.

Now we introduce the class $\mathbf{N}$, which are defined over the natural numbers, contrary to the characterization of words in Oitavem 2. Let $a+1$ stand for the numeric successor of a, whereas the notation n' is used in 2. The notation $a \dot{-} 1$ is used to denote the corresponding predecessor of a.

Definition 1.4 (The class N). A class $\mathbf{N}^{k,l}$ of functions with k normal and l safe arguments is defined by the following initial functions and operations:

Zero $O^{k,l} \in \mathbf{N}^{k,l}$; $\quad O^{k,l}(x_1, \ldots, x_k; a_1, \ldots, a_l) = 0$
Projections $I_j^{k,l} \in \mathbf{N}^{k,l}$ $(1 \leq j \leq k + l)$;

$$I_j^{k,l}(x_1, \ldots, x_k; a_1, \ldots, a_l) = \begin{cases} x_j & \text{if } 1 \leq j \leq k, \\ a_{j-k} & \text{if } k < j \leq k + l. \end{cases}$$

Successor $S \in \mathbf{N}^{0,1}$; $\quad S(; a) = a + 1$
Predecessor $P \in \mathbf{N}^{0,1}$; $\quad P(; a) = a \dot{-} 1$

Cases $C \in \mathbf{N}^{0,3}$; $\quad C(;a,b,c) = \begin{cases} b \text{ if } a = 0 \\ c \text{ if } a > 0 \end{cases}$

i-Concatenation $C_i \in \mathbf{N}^{1,0}$ $(i = 0, 1)$; $\quad C_i(x;) = 2x + i = xi$

Deletion $D \in \mathbf{N}^{0,1}$; $\quad D(;a) = \lfloor a/2 \rfloor$

Safe composition If $h \in \mathbf{N}^{k',l'}$, $g_1, \ldots, g_{k'} \in \mathbf{N}^{k,0}$ and $\varphi_1, \ldots, \varphi_{l'} \in \mathbf{N}^{k,l}$, then $f \in \mathbf{N}^{k,l}$ is defined by $f(\vec{x};\vec{a}) = h(\vec{g}(\vec{x};);\vec{\varphi}(\vec{x};\vec{a}))$.

Safe nested recursion on notation (SNRN) Suppose that $g \in \mathcal{N}^{k',l+1}$, and h_w, $\varphi_w \in \mathcal{N}^{k+k',l+2}$ for each $w \in \Sigma^k$. Also suppose that $\mathbf{f}_1, \mathbf{f}_2$ are $\prec^k$-functions. Then $f \in \mathcal{N}^{k+k',l+1}$ is defined by

$$\begin{cases} f(\vec{0},\vec{x};\vec{a},b) = g(\vec{x};\vec{a},b), \\ f(C_w(\vec{y};),\vec{x};\vec{a},b) = h_w(\vec{v}_1,\vec{x};\vec{a},b,f(\vec{v}_1,\vec{x};\vec{a},c)) \\ \qquad\qquad [\varphi_w(\vec{v}_2,\vec{x};\vec{a},b,f(\vec{v}_2,\vec{x};\vec{a},b))/c] \quad (w \in \Sigma^k), \end{cases}$$

where, for every $i = 1, 2$, $\vec{v}_i$ abbreviates $C_{w(\mathbf{f}_i(w))}(\vec{y}_{\mathbf{f}_i(w)};)$, and hence $\vec{v}_i \prec^k C_w(\vec{y};)$.

Then we define $\mathbf{N} := \bigcup_{k,l\in\mathbb{N}} \mathbf{N}^{k,l}$ and $\mathbf{N}_{normal} := \bigcup_{k\in\mathbb{N}} \mathbf{N}^{k,0}$.

Remark 1.5. In the above scheme of SNRN, the restriction of the nesting is not crucial. Even if we allow any constant number of nestings, arguments in Section 2 work. Further, Arai and the author[4] employ a more general scheme of the form

$$\begin{cases} f(\vec{0},\vec{x};\vec{a}) = g(\vec{x};\vec{a}), \\ f(C_w(\vec{y}),\vec{x};\vec{a}) = h_w(\vec{v}_1,\vec{x};\vec{a},f(\vec{v}_1,\vec{x};\vec{\varphi}_w(\vec{v}_2,\vec{x};\vec{a},f(\vec{v}_2,\vec{x};\vec{a})))) \end{cases} \tag{1}$$

for $\vec{\varphi}_w = \varphi_{w,1}, \ldots, \varphi_{w,l}$ and an $l > 0$. If the projection function $I^{k+k',l}_{k+k'+i}$ is taken as $\varphi_{w,i}$ for each $i = 1, \ldots, l-1$, then this scheme is just the SNRN scheme. If the scheme (1) is contained instead of the one in Definition 1.4, Theorem 2.11 does not hold. Nevertheless, this restriction is not essential either. The same class will be generated even by (1). See also Remark 2.12.

Example 1.6.

(1) $\dot{-}(x;a) = D^{2^{|x|}}(;a)$.

$$\begin{aligned} \dot{-}(0;a) &= D(;a), \\ \dot{-}(xi;a) &= \dot{-}(x;\dot{-}(x;a)). \end{aligned}$$

(2) $+(x;a) = x + a$.

$$\begin{aligned} +(0;a) &= a, \\ +(x0;a) &= 2x + a = +(x;+(x;a)), \\ +(x1;a) &= 2x + 1 + a = S(;+(x;+(x;a))). \end{aligned}$$

Similarly, $\dot{-}(x;a) = a\dot{-}x$ is defined.

(3) $\times(x,y;a) = y \cdot 2^{|x|} + a$.

$$\begin{aligned} \times(0,y;a) &= y + a, \\ \times(xi,y;a) &= \times(x,y;\times(x,y;a)). \end{aligned}$$

(4) $f(x;a) = 2^{|x|} + a$.

$$\begin{aligned} f(0;a) &= S(;a), \\ f(xi;a) &= 2^{|x|+1} + a \\ &= f(x;f(x;a)). \quad (i = 0,1) \end{aligned}$$

(5) $f(x,y,z;a) = 2^{|x|\cdot|y|+|z|} + a$.

$$\begin{aligned} f(0,0,0;a) &= S(;a), \\ f(x,y,zi;a) &= 2^{|x|\cdot|y|+|z|+1} + a = f(x,y,z;f(x,y,z;a)), \\ f(x,yi,0;a) &= 2^{|x|(|y|+1)} + a = f(x,y,x;a), \\ f(xi,0,0;a) &= f(x,0,0;a). \quad (i = 0,1) \end{aligned}$$

Similarly, we can define $f(x,y,z,u,v,w;a) = 2^{|x||y||z|+|u||v|+|w|} + a$ and so on. Hence, a suitable application of safe composition yields $2^{p(|\vec{x}|)} \in \mathbf{N}_{normal}$ for any polynomial $p(\vec{x})$.

(6) $2^{2^{|x|}} \notin \mathbf{N}$, contrary to $f(x;a) = 2^{2^{|x|}} \cdot a \in \mathcal{N}$ in 4. This is due to the choice of initial functions. The binary successors $S_0, S_1 \in \mathcal{N}^{0,1}$ are defined so that $S_i(;a) = 2a + i$ $(i = 0,1)$. However $C_0, C_1 \in \mathbf{N}^{1,0}$ are defined on the normal argument, and therefore C_i cannot be g in the scheme of SNRN in Definition 1.4.

Recall that **FPS** is the class of functions computed by a deterministic Turing machine with the use of a number of cells bounded by $p(|\vec{x}|)$ for some polynomial $p(\vec{x})$. The following theorem is a direct consequence of results in Section 2 and 3.

Theorem 1.7. $\mathbf{N}_{normal} = \mathbf{FPS}$.

Remark 1.8. Let us recall briefly the scheme of **safe recursion**:

$$\begin{cases} f(0,\vec{x};\vec{a}) = g(\vec{x};\vec{a}), \\ f(S(;y),\vec{x};\vec{a}) = h(y,\vec{x};\vec{a},f(y,\vec{x};\vec{a})). \end{cases}$$

Via a work of Bellantoni (5 Chapter 5), it turns out that if the SNRN scheme is replaced by the scheme of safe recursion (and even if both C_i and D are omitted), the resulting class is identical to the class of linear-space computable functions. Moreover, replace SNRN by safe nested recursion (on unary notation), and restrict safe composition suitably, cf. 6 or 4. Then the resulting class will be identical to the class of functions computable in $2^{O(|x|)}$-space, i.e., EXPSPACE-computable functions. These observations together with Theorem 1.7 suggest an uniform approach to space-complexity classes.

2. Term rewriting system for PSPACE

In this section we introduce a term rewriting system $R_{\mathbf{N}}$ over a class $\mathcal{F}$ of function symbols corresponding to $\mathbf{N}$. We show that for any $f \in \mathcal{F}$, the size of every term occurring in a rewriting sequence which starts with $f(\vec{m};\vec{n})$ is bounded by a polynomial in the binary lengths of numerals $\vec{m}$ and $\vec{n}$, whenever an innermost strategy is used. This implies that every $R_{\mathbf{N}}$-reduction (innermost) strategy yields an algorithm for $\mathbf{N}$ running in polynomial space, i.e., $\mathbf{N}_{normal} \subseteq \mathbf{FPS}$.

Let $\mathcal{V}$ be a countably infinite set of variables. Variables are written as x, y, z. We use $\mathcal{V}ar(t)$ to denote the set of variables occurring in a term t. A finite set R of rewrite rules is called a term rewriting system (TRS for short) if every rewrite rule $l \to r \in R$ satisfies $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. A TRS R defines a rewriting relation $\to_R$ by $l \to r \in R \Rightarrow r[l\theta] \to_R r[r\theta]$ for any context $r[\cdot]$ and substitution θ. Then $l\theta$ is called a redex. The reflexive and transitive closure of $\to_R$ is denoted by $\to_R^*$.

$\mathcal{F}$ is the smallest class of symbols built up from $O^{k,l}, I_j^{k,l}, S, P, C, C_i, D$ by means of $\mathrm{SUB}^{k,l}$ and $\mathrm{SNRN}^{k,l}$, e.g., if $g \in \mathcal{F}^{k',l+1}$ and $h_w, \varphi_w \in \mathcal{F}^{k+k',l+2}$ for all $w \in \Sigma^k$, then $\mathrm{SNRN}^{k+k',l+1}[g, \{h_w, \varphi_w : w \in \Sigma^k\}] \in \mathcal{F}^{k+k',l+1}$. Superscripts of SUB and SNRN are always omitted if no confusion arises.

Let $\mathcal{T}(\mathcal{F},\mathcal{V})$ be the set of terms over $\mathcal{F}$ and $\mathcal{V}$. Let $\mathcal{T}(\mathcal{F})$ be the set of ground terms in $\mathcal{T}(\mathcal{F},\mathcal{V})$, which are built up from $O^{0,0}$ by means of elements of $\mathcal{F}$. Numerals are built up from $O^{0.0}$ by means of C_0, C_1. We write $n, m, \ldots$ for numerals and 0 instead of $O^{0,0}$. The binary length $|m|$ of a numeral m

is defined by $|0| = 1$, and $C_i(m;) = |m| + 1$. By replacing "=" occurring in the equations in Definition 1.4 by "$\to$", we obtain a schematic TRS for $\mathbf{N}$. However, relaxing the definition, we introduce a non-deterministic or non-confluent TRS $R_{\mathbf{N}}$ (according to the referee's suggestion).

Definition 2.1 (Term rewriting system $R_{\mathbf{N}}$).

(1) $O^{k,l}(\vec{x};\vec{y}) \to 0$ $(0 < k + l)$
(2) $I^{k,l}_j(x_1,\ldots,x_k;x_{k+1},\ldots,x_{k+l}) \to x_j$ $(1 \le j \le k + l)$
(3) $S(;0) \to C_1(0;)$
(4) $S(;C_0(x;)) \to C_1(x;)$
(5) $S(;C_1(x;)) \to C_0(S(;x);)$
(6) $P(;0) \to 0$
(7) $P(;C_0(x;)) \to C_1(P(;x);)$
(8) $P(;C_1(x;)) \to C_0(x;)$
(9) $C(;0,x,y) \to x$
(10) $C(;C_i(z;),x,y) \to y$
(11) $D(;0) \to 0$
(12) $D(;C_i(x;)) \to x$
(13) $\mathrm{SUB}[h,\vec{g},\vec{\varphi}](\vec{x};\vec{y}) \to h(\vec{g}(\vec{x};);\vec{\varphi}(\vec{x};\vec{y}))$
(14) $\mathrm{SNRN}[g,\{h_w,\varphi_w : w \in \Sigma^k\}](\vec{0},\vec{x};\vec{z},z') \to g(\vec{x};\vec{z},z')$
(15) $\mathrm{SNRN}[g,\{h_w,\varphi_w : w \in \Sigma^k\}](C_{w'}(\vec{y};),\vec{x};\vec{z},z') \to$
$h_{w'}(\vec{v}_1,\vec{x};\vec{z},z',\mathrm{SNRN}[g,\{h_w,\varphi_w : w \in \Sigma^k\}](\vec{v}_1,\vec{x};\vec{z},u))$
$[\varphi_{w'}(\vec{v}_2,\vec{x};\vec{z},z',\mathrm{SNRN}[g,\{h_w,\varphi_w : w \in \Sigma^k\}](\vec{v}_2,\vec{x};\vec{z},z'))/u]$
for some $\vec{v}_1,\vec{v}_2 \prec^k C_{w'}(\vec{y};)$ $(w' \in \Sigma^k)$.

The intended semantic π for $\mathcal{F}$ is obvious. For example, suppose that $f = \mathrm{SUB}[h,\vec{g},\vec{\varphi}]$. Then $\pi(f) \in \mathbf{N}$ is defined by safe composition from $\pi(h)$, $\pi(\vec{g})$ and $\pi(\vec{\varphi})$ in $\mathbf{N}$: $\pi(f)(\vec{x};\vec{a}) = \pi(h)(\pi(\vec{g})(\vec{x};);\pi(\vec{\varphi})(\vec{x};\vec{a}))$. However, one should be careful of the case $f = \mathrm{SNRN}[g,\{h_w,\varphi_w : w \in \Sigma^k\}]$. Clearly, the TRS $R_{\mathbf{N}}$ is not confluent due to absence of $\prec^k$-functions $\mathbf{f}_1,\mathbf{f}_2$. Let us recall that a $\prec$-function indicates which $\prec$-predecessor should be chosen. The mapping π is extended to $\mathcal{T}(\mathcal{F}) \to \mathbb{N}$ by $\pi(f(\vec{t};\vec{s})) := \pi(f)(\pi(\vec{t});\pi(\vec{s}))$.

Lemma 2.2. *$R_{\mathbf{N}}$ is terminating.*

Proof. Define a precedence $<_{\mathcal{F}}$ on $\mathcal{F}$ by

- $0 <_{\mathcal{F}} O^{k,l}$ if $k + l > 0$,
- $g <_{\mathcal{F}} f$ for each $g \in \{O^{k,l}, I^{k,l}_j, C_i\}$ and $f \in \{S,P,C,D\}$,
- $f <_{\mathcal{F}} \mathrm{SUB}[h,\vec{g},\vec{\varphi}]$ for each $f \in \{h,\vec{g},\vec{\varphi}\}$, and
- $f <_{\mathcal{F}} \mathrm{SNRN}[g,\{h_w,\varphi_w : w \in \Sigma^k\}]$ for each $f \in \{g,h_w,\varphi_w : w \in \Sigma^k\}$.

Then $R_{\mathbf{N}}$ is reducing under the lexicographic path order LPO induced by $<_{\mathcal{F}}$. Namely, $t \to_{R_{\mathbf{N}}} s \Rightarrow s <_{\mathrm{LPO}} t$ for each $t, s \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. The well-foundedness of LPO implies the termination of $R_{\mathbf{N}}$. □

By the termination of $R_{\mathbf{N}}$, a normal form always exists, i.e., for any $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, there exists a $t' \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ such that $t \to^*_{R_{\mathbf{N}}} t'$ and $\nexists s$ s.t. $t' \to_{R_{\mathbf{N}}} s$. It is not difficult to see that any normal form of each ground term is a numeral.

Notation 2.3. Since $R_{\mathbf{N}}$ is not confluent, there may be several normal forms of a ground term. At the risk of confusion, let $\underline{t}$ denote a normal form of $t \in \mathcal{T}(\mathcal{F})$. Hence, in paticular, $\underline{m} = m$ for any numeral m.

Definition 2.4. The length $\mathrm{lh}(f)$ of $f \in \mathcal{F}$ is defined by

- $\mathrm{lh}(f) = 1$ if $f \in \{O^{k,l}, I^{k,l}_j, S, P, C, C_i, D\}$,
- $\mathrm{lh}(\mathrm{SUB}[h, \vec{g}, \vec{\varphi}]) = \mathrm{lh}(h) + \sum_{i=1}^{k'} \mathrm{lh}(g_i) + \sum_{i=1}^{l'} \mathrm{lh}(\varphi_i) + 1$, and
- $\mathrm{lh}(\mathrm{SNRN}[g, \{h_w, \varphi_w : w \in \Sigma^k\}]) = \mathrm{lh}(g) + \sum_{w \in \Sigma^k} \mathrm{lh}(h_w) + \sum_{w \in \Sigma^k} \mathrm{lh}(\varphi_w) + 1$

Then, the length $\mathrm{lh}(t)$ of $t \in \mathcal{T}(\mathcal{F})$ is defined by

$$\mathrm{lh}(f(\vec{t}; \vec{s})) = \mathrm{lh}(f) + \sum_{i=1}^{k} \mathrm{lh}(t_i) + \sum_{i=1}^{l} \mathrm{lh}(s_i).$$

Hence, in particular, $|m| = \mathrm{lh}(m)$ for any numeral m. As a corollary, we observe Lemma 2.5.

Lemma 2.5. *If $t \to s \in R_{\mathbf{N}}$ by any of Definition* 2.1(1)—2.1(12)*, then $\mathrm{lh}(s\theta) \leq \mathrm{lh}(t\theta)$ for any ground substitution θ.*

In general, for $f \in \mathcal{F}$, $\max\{\mathrm{lh}(t) : f(\vec{m}; \vec{n}) \to^*_{R_{\mathbf{N}}} t\}$ is not bounded by a polynomial in $|\vec{m}|, |\vec{n}|$. Consider the rewriting rule $f(C_i(y;); x) \to f(y; f(y; x))$. For numerals m and n, it is possible to obtain the rewriting sequence $f(m; n) \to^*_{R_{\mathbf{N}}} f(0; f(0; \cdots f(0; n) \cdots))$. However, $\mathrm{lh}(f(0; f(0; \cdots f(0; n) \cdots))) = (\mathrm{lh}(f) + 1)2^{|m|} + |n|$, which is not admissible. Hence we employ the innermost rewriting strategy.

Notation 2.6. Let us use $t \to_{in} s$ to denote $t \to_{R_{\mathbf{N}}} s$ according to an innermost strategy. Namely, $t \to_{in} s$ if $t \to_{R_{\mathbf{N}}} s$ and the redex r is innermost, i.e., r contains no redex that is a proper subterm of r.

The rest of this section is devoted to show that for every $f \in \mathcal{F}$ there exists a polynomial p such that $\max\{\mathrm{lh}(t) : f(\vec{m};\vec{n}) \to_{in}^{*} t\} \leq p(|\vec{m}|,|\vec{n}|)$. Due to the innermost strategy, $\max\{\mathrm{lh}(t) : f(\vec{t};\vec{s}) \to_{R_{\mathbf{N}}}^{*} t\}$ can be reduced as follows.

Lemma 2.7. *For any $f \in \mathcal{F}^{k,l}$ and $t_1,\ldots,t_k,s_1,\ldots,s_l \in \mathcal{T}(\mathcal{F})$, $\max\{\mathrm{lh}(t) : f(\vec{t};\vec{s}) \to_{in}^{*} t\}$ is bounded by either $\mathrm{lh}(f) + \sum_{i=1}^{k} \max\{\mathrm{lh}(t_i^*) : t_i \to_{in}^{*} t_i^*\} + \sum_{i=1}^{l} \max\{\mathrm{lh}(s_i^*) : s_i \to_{in}^{*} s_i^*\}$, or $\max\{\mathrm{lh}(t) : f(\vec{t};\vec{s}) \to_{in}^{*} t\}$.*

In the following proofs, we use $\mathrm{Sp}(t)$ to abbreviate $\max\{\mathrm{lh}(s) : t \to_{in}^{*} s\}$, which intended to denote the *space* required to rewrite $t \in \mathcal{T}(\mathcal{F})$ according to $\to_{in}$.

Proof. Suppose that $f \in \{O^{k,l}, I_j^{k,l}, C_i, D, S, P, C\}$. From Lemma 2.5, we observe that

$$\mathrm{Sp}(f(\vec{t};\vec{s})) \leq \mathrm{lh}(f) + \sum_{i=1}^{k} \mathrm{Sp}(t_i) + \sum_{i=1}^{l} \mathrm{Sp}(s_i). \tag{2}$$

Otherwise, $f \in \{\mathrm{SUB}[h,\vec{g},\vec{\varphi}], \mathrm{SNRN}[g,\{h_w,\varphi_w : w \in \Sigma^k\}]\}$. Then, due to the innermost strategy, an arbitrary terminating rewriting which starts with $f(\vec{t};\vec{s})$ runs as

$$f(\vec{t};\vec{s}) \to_{in}^{*} f(\underline{\vec{t}};\underline{\vec{s}}) \to_{in}^{*} \underline{f(\underline{\vec{t}};\underline{\vec{s}})}.$$

Thus, $\mathrm{Sp}(f(\vec{t};\vec{s}))$ is bounded by either $\mathrm{lh}(f) + \sum_{i=1}^{k} \mathrm{Sp}(t_i) + \sum_{i=1}^{l} \mathrm{Sp}(s_i)$ or $\mathrm{Sp}(f(\underline{\vec{t}};\underline{\vec{s}}))$. This concludes the lemma. □

Definition 2.8. For $d, b_1, \ldots, b_k \in \mathbb{N}$, put

$$\sum(d,\vec{b}) = \sum(d,b_1,\ldots,b_k) := \sum_{i=1}^{k} (\max \vec{b} + 1)^{d-i} b_i.$$

Then, as in 4, one can prove a fundamental lemma concerning the descending lengths with respect to the ordering $\prec$.

Lemma 2.9. *Suppose that $\vec{n}$ and $\vec{m}$ are numerals. If $\vec{n} \prec^{k} \vec{m}$ and $k \leq d$, then $\sum(d,|\vec{n}|) < \sum(d,|\vec{m}|)$.*

Lemma 2.10. *For any $f \in \mathcal{F}$ there exist some constants c and d such that for numerals $\vec{m},\vec{n}$ and an arbitrary normal form $\underline{f(\vec{m};\vec{n})}$ of $f(\vec{m};\vec{n})$, $2^{|\underline{f(\vec{m};\vec{n})}|} \leq 2^{c(\max|m|^d+1)} + 2^{\max|n|}$, and hence $|\underline{f(\vec{m};\vec{n})}| \leq \max\{c(\max|m|^d+1), \max|n|\} + 1$.*

Proof. We prove the lemma by induction on $\mathrm{lh}(f)$. For the base case, if $\mathrm{lh}(f) = 1$, i.e., $f \in \{O, I, C_i, S, P, D, C\}$, then $2^{|\underline{f(\vec{m};\vec{n})}|} \leq 2^{\max|\vec{m}|+1} + 2^{\max|\vec{n}|}$.

Consider the induction step. Suppose that $f = \mathrm{SUB}[h, \vec{g}, \vec{\varphi}]$ for $h \in \mathbf{N}^{k',l'}$, $g_1, \ldots, g_{k'} \in \mathbf{N}^{k,0}$ and $\varphi_1, \ldots, \varphi_{l'} \in \mathbf{N}^{k,l}$. Then, by the induction hypothesis, there exist some constants c_0, d_0 for h, c_i, d_i for g_i $(i = 1, \ldots, k')$ and c_{k+i}, d_{k+i} for φ_i $(i = 1, \ldots, l')$ which enjoy the condition. Fix an normal form $\underline{g_i(\vec{m};)}$ of $g_i(\vec{m};)$ for each $i = 1, \ldots, k'$ and $\underline{\varphi_i(\vec{m};\vec{n})}$ of $\varphi_i(\vec{m};\vec{n})$ for each $i = 1, \ldots, l'$. Let $c' := \max\{c_0, c_1, \ldots, c_{k'+l'}\}$ and $d' := \max\{d_0, d_1, \ldots, d_{k'+l'}\}$. Then, by IH for $\vec{g}$ and $\vec{\varphi}$,

$$|\underline{g_i(\vec{m};)}| \leq c'(\max|\vec{m}|^{d'} + 1) \ (i = 1, \ldots, k'), \tag{3}$$

$$2^{|\underline{\varphi_i(\vec{m};\vec{n})}|} \leq 2^{c'(\max|\vec{m}|^{d'}+1)} + 2^{\max|\vec{n}|} \ (i = 1, \ldots, l'). \tag{4}$$

Fix an normal form $\underline{f(\vec{m};\vec{n})} = \underline{h(\underline{\vec{g}(\vec{m};)};\underline{\vec{\varphi}(\vec{m};\vec{n})})}$ of $f(\vec{m};\vec{n})$. By IH for h,

$$\begin{aligned} 2^{|\underline{f(\vec{m};\vec{n})}|} &= 2^{|\underline{h(\underline{\vec{g}(\vec{m};\vec{n})};\underline{\vec{\varphi}(\vec{m};\vec{n})})}|} \\ &\leq 2^{c'(\max|\underline{\vec{g}(\vec{m};)}|^{d'}+1)} + 2^{\max|\underline{\vec{\varphi}(\vec{m};\vec{n})}|}. \end{aligned} \tag{5}$$

Put $c := 2^{d'}c'^{d'+1} + 1$ and $d := d'^2$. Then,

$$\begin{aligned} c'(\max|\underline{\vec{g}(\vec{m};)}|^{d'} + 1) &\leq c'(c'(\max|\vec{m}|^{d'} + 1)^{d'} + 1) \quad \text{by (3)}, \\ &\leq c'(c'^{d'}2^{d'}\max|\vec{m}|^{d'^2} + 1) \\ &\leq 2^{d'}c'^{d'+1}(\max|\vec{m}|^{d'^2} + 1) \\ &\leq (c-1)(\max|\vec{m}|^{d} + 1) \end{aligned} \tag{6}$$

by the definitions of the constants c and d. Thus, by (5), (6) and (4),

$$\begin{aligned} 2^{|\underline{f(\vec{m};\vec{n})}|} &\leq 2^{(c-1)(\max|\vec{m}|^{d}+1)} + 2^{c'(\max|\vec{m}|^{d'}+1)} + 2^{\max|\vec{n}|} \\ &\leq 2^{(c-1)(\max|\vec{m}|^{d}+1)+1} + 2^{\max|\vec{n}|} \quad \text{by } c' \leq c-1, \\ &\leq 2^{c(\max|\vec{m}|^{d}+1)} + 2^{\max|\vec{n}|}. \end{aligned}$$

For the case for SNRN, it is convenient to consider a more general form (1) in Remark 1.5:

$$\begin{cases} f(\vec{0}, \vec{x}; \vec{a}) = g(\vec{x}; \vec{a}), \\ f(C_w(\vec{y}), \vec{x}; \vec{a}) = h_w(\vec{v}_1, \vec{x}; \vec{a}, f(\vec{v}_1, \vec{x}; \vec{\varphi}_w(\vec{v}_2, \vec{x}; \vec{a}, f(\vec{v}_2, \vec{x}; \vec{a})))) \\ (\vec{\varphi}_w = \varphi_{w,1}, \ \ldots, \ \varphi_{w,l}, \ l > 0) \end{cases}$$

Let $f = \mathrm{SNRN}[g, \{h_w, \vec{\varphi}_w : w \in \Sigma^k\}] \in \mathcal{F}^{k+k',l}$ for $g \in \mathcal{F}^{k',l}$ and h_w, $\varphi_{w,1}$, $\ldots$, $\varphi_{w,l} \in \mathcal{F}^{k+k',l+1}$. By IH, there exist some constants c_0, d_0 for g, c_1, d_1

for h_w, and c_2, d_2 for $\vec{\varphi}_w$ which enjoy the condition for all $w \in \Sigma^k$. Put $c := \max\{c_0, c_1, c_2, 2\}$ and $d := \max\{d_0, d_1, d_2, k\}$. Suppose that $m_1^y, \ldots, m_k^y$ and $m_1^x, \ldots, m_{k'}^x$ are numerals. Then, by side induction on $\sum(d, |\vec{m}^y|)$, we show that for any $\vec{t} = t_1, \ldots, t_l \in \mathcal{T}(\mathcal{F})$ and their arbitrary normal forms $\underline{\vec{t}} = \underline{t_1}, \ldots, \underline{t_l}$,

$$2^{|\underline{f(\vec{m}^y, \vec{m}^x; \vec{t})}|} \leq 2^{c((\max\{|\vec{m}^y|, |\vec{m}^x|\}+1)^d + \sum(d, |\vec{m}^y|)+1)} + 2^{\max|\vec{t}|}.$$

Put $b := \max\{|\vec{m}^y|, |\vec{m}^x|\} + 1$ and $b_i := |m_i^y|$ for each $i = 1, \ldots, k$. First consider the case $\vec{m}^y = \vec{0}$: $f(\vec{0}, \vec{m}^x; \vec{t}) \to_{in} g(\vec{m}^x; \vec{t})$. This case follows from IH as

$$\begin{aligned} 2^{|\underline{f(\vec{0}, \vec{m}^x; \vec{t})}|} &= 2^{|\underline{g(\vec{m}^x; \vec{t})}|} \\ &\leq 2^{c_0(b^{d_0}+1)} + 2^{\max|\vec{t}|} \quad \text{by IH for } g, \\ &\leq 2^{c(b^d + \sum(d, \vec{b})+1)} + 2^{\max|\vec{t}|} \quad \text{by } c_0 \leq c, \text{ and } d_0 \leq d. \end{aligned}$$

Next consider the case $\vec{m}^y \neq \vec{0}$:

$$\begin{aligned} f(\vec{m}^y, \vec{m}^x; \underline{\vec{t}}) &\to_{in} h_w(\vec{m}_1^v, \vec{m}^x; \underline{\vec{t}}, f(\vec{m}_1^v, \vec{m}^x; \vec{\varphi}_w(\vec{m}_2^v, \vec{m}^x; \underline{\vec{t}}, f(\vec{m}_2^v, \vec{m}^x; \underline{\vec{t}})))) \\ &\to_{in}^* h_w(\vec{m}_1^v, \vec{m}^x; \underline{\vec{t}}, f(\vec{m}_1^v, \vec{m}^x; \vec{\varphi}_w(\vec{m}_2^v, \vec{m}^x; \underline{\vec{t}}, \underline{f(\vec{m}_2^v, \vec{m}^x; \underline{\vec{t}})}))) \\ &\to_{in}^* h_w(\vec{m}_1^v, \vec{m}^x; \underline{\vec{t}}, f(\vec{m}_1^v, \vec{m}^x; \underline{\vec{\varphi}_w(\vec{m}_2^v, \vec{m}^x; \underline{\vec{t}}, \underline{f(\vec{m}_2^v, \vec{m}^x; \underline{\vec{t}})})})) \\ &\to_{in}^* h_w(\vec{m}_1^v, \vec{m}^x; \underline{\vec{t}}, \underline{f(\vec{m}_1^v, \vec{m}^x; \underline{\vec{\varphi}_w(\vec{m}_2^v, \vec{m}^x; \underline{\vec{t}}, \underline{f(\vec{m}_2^v, \vec{m}^x; \underline{\vec{t}})})})}). \end{aligned}$$

for some $\vec{m}_1^v, \vec{m}_2^v \prec^k \vec{m}^y$ and a suitable $w \in \Sigma^k$. By Lemma 2.9, for each $j = 1, 2$,

$$\sum(d, |\vec{m}_j^v|) < \sum(d, \vec{b}). \tag{7}$$

Hence, by the side induction hypothesis and the condition that $\max|\vec{m}_2^v| \leq \max|\vec{m}^y|$,

$$2^{|\underline{f(\vec{m}_2^v, \vec{m}^x; \vec{t})}|} \leq 2^{c(b^d + \sum(d, |\vec{m}_2^v|)+1)} + 2^{\max|\vec{t}|} \leq 2^{c(b^d + \sum(d, \vec{b}))} + 2^{\max|\vec{t}|}.$$

From this and IH for $\vec{\varphi}_w$,

$$\begin{aligned} 2^{\max|\underline{\vec{\varphi}_w(\vec{m}_2^v, \vec{m}^x; \underline{\vec{t}}, \underline{f(\vec{m}_2^v, \vec{m}^x; \underline{\vec{t}})})|}} &\leq 2^{c_2(b^{d_2}+1)} + 2^{\max\{|\underline{\vec{t}}|, |\underline{f(\vec{m}_2^v, \vec{m}^x; \underline{\vec{t}})}|\}} \\ &\leq 2^{c_2(b^{d_2}+1)} + 2^{c(b^d + \sum(d, \vec{b}))} + 2^{\max|\vec{t}|} \\ &\leq 2 \cdot 2^{c(b^d + \sum(d, \vec{b}))} + 2^{\max|\vec{t}|}, \end{aligned} \tag{8}$$

since $c_2 \leq c$, $d_2 \leq d$, and $1 \leq \sum(d, \vec{b})$. Similarly to the case for $\vec{m}_2^v$, by SIH together with (7) for $\vec{m}_1^v$,

$$\begin{aligned}
&2^{\underline{|f(\vec{m}_1^v,\vec{m}^x;\underline{\vec{\varphi}_w(\vec{m}_2^v,\vec{m}^x;\underline{\vec{t}},\underline{f(\vec{m}_2^v,\vec{m}^x;\underline{\vec{t}})})|}}} \\
&\leq 2^{c(b^d+\sum(d,|\vec{m}_1^v|)+1)} + 2^{\max|\underline{\vec{\varphi}_w(\vec{m}_2^v,\vec{m}^x;\vec{t},\underline{f(\vec{m}_2^v,\vec{m}^x;\underline{\vec{t}})})|}} \\
&\leq 2^{c(b^d+\sum(d,\vec{b}))} + 2\cdot 2^{c(b^d+\sum(d,\vec{b}))} + 2^{\max|\underline{\vec{t}}|} \quad \text{by (8)}, \\
&\leq 3\cdot 2^{c(b^d+\sum(d,\vec{b}))} + 2^{\max|\underline{\vec{t}}|}.
\end{aligned}$$

Therefore,

$$\begin{aligned}
&2^{|\underline{f(\vec{m}^y,\vec{m}^x;\underline{\vec{t}})}|} \\
&= 2^{\underline{|h_w(\vec{m}_1^v,\vec{m}^x;\underline{\vec{t}},\underline{f(\vec{m}_1^v,\vec{m}^x;\underline{\vec{\varphi}_w(\vec{m}_2^v,\vec{m}^x;\underline{\vec{t}},\underline{f(\vec{m}_2^v,\vec{m}^x;\underline{\vec{t}})})})})|}} \\
&\leq 2^{c_1(b^{d_1}+1)} + 3\cdot 2^{c(b^d+\sum(d,\vec{b}))} + 2^{\max|\underline{\vec{t}}|} \quad \text{by IH for } h_w, \\
&\leq 4\cdot 2^{c(b^d+\sum(d,\vec{b}))} + 2^{\max|\underline{\vec{t}}|} \quad \text{by } c_1 \leq c,\ d_1 \leq d \text{ and } 1 \leq \sum(d,\vec{b}), \\
&\leq 2^{c(b^d+\sum(d,\vec{b})+1)} + 2^{\max|\underline{\vec{t}}|} \quad \text{by } 2 \leq c.
\end{aligned}$$

This completes the proof of the lemma. □

Theorem 2.11. *For any $f \in \mathcal{F}^{k,l}$, there exist c and d such that for numerals $m_1, \ldots, m_k$ and $n_1, \ldots, n_l$, $\max\{\mathrm{lh}(t) : f(\vec{m};\vec{n}) \to_{in}^* t\} \leq c(\max|\vec{m}|^d + 1)(\max|\vec{n}| + 1)$.*

Proof. We prove the theorem again by induction on $\mathrm{lh}(f)$. If $\mathrm{lh}(f) = 1$, then, by (2),

$$\begin{aligned}
\mathrm{Sp}(f(\vec{m};\vec{n})) &\leq \mathrm{lh}(f) + \sum_{i=1}^{k}|m_i| + \sum_{i=1}^{l}|n_i| \\
&\leq (\mathrm{lh}(f)+k+l)(\max|\vec{m}|+1)(\max|\vec{n}|+1). \qquad (9)
\end{aligned}$$

The case $f = \mathrm{SUB}[h, \vec{g}, \vec{\varphi}]$ is seen from IH, Lemma 2.7, and Lemma 2.10.

Finally, consider the case $f = \mathrm{SNRN}[g, \{h_w, \varphi_w : w \in \Sigma^k\}]$. For simplicity, we only consider $f \in \mathcal{F}^{k,1}$ such that $f(\vec{0};x) \to g(;x)$ and $f(\vec{y};x) \to f(\vec{v}_1; f(\vec{v}_2;x))$ for some $\vec{v}_1, \vec{v}_2 \prec^k \vec{y}$. By IH, there exists a constant c_g for g enjoying the condition. Furthermore, by Lemma 2.10, there exist constants c', d' for f enjoying the condition in the lemma. Let $c_{\mathrm{lh}} := \mathrm{lh}(f)+k+1$. Now put $c := \max\{c_{\mathrm{lh}}, c_g, c'+1\}$ and $d := \max\{k, d'\}$. Suppose that $m_1, \ldots, m_k$ are numerals. Then, by side induction on $\sum(d, |\vec{m}|)$, we show that for any $t \in \mathcal{T}(\mathcal{V})$ and its arbitrary normal form $\underline{t}$,

$$2^{\mathrm{Sp}(f(\vec{m};\underline{t}))} \leq (2^{c(\sum(d,|\vec{m}|)+1)} + 2^{|\underline{t}|+1})^{c(\sum(d,|\vec{m}|)+1)}. \qquad (10)$$

This results in $\mathrm{Sp}(f(\vec{m};\underline{t})) \leq c^2(\sum(d,|\vec{m}|)+1)^2(|\underline{t}|+1)$.

Case 1. $\mathrm{Sp}(f(\vec{m};\underline{t})) \leq \mathrm{lh}(f) + \sum_{i=1}^{k}|m_i| + |\underline{t}|$: In this case, (10) follows, since

$$\begin{aligned}\mathrm{Sp}(f(\vec{m};\underline{t})) &\leq \mathrm{lh}(f) + k\max|\vec{m}| + |\underline{t}| \\ &\leq c_{\mathrm{lh}}(\max|\vec{m}| + |\underline{t}| + 1) \\ &\leq c(\max|\vec{m}|+1)(|\underline{t}|+1) \quad \text{by } c_{\mathrm{lh}} \leq c.\end{aligned}$$

Case 2. $\mathrm{Sp}(f(\vec{m};\underline{t})) \leq \mathrm{Sp}(g(;\underline{t}))$: In this case, (10) follows, since

$$\begin{aligned}\mathrm{Sp}(g(;\underline{t})) &\leq c_g(|\underline{t}|+1) \quad \text{by IH for } g, \\ &\leq c(\max|\vec{m}|^d+1)(|\underline{t}|+1) \quad \text{by } c_g \leq c.\end{aligned}$$

Case 3. Otherwise: By Lemma 2.7,

$$\mathrm{Sp}(f(\vec{m};\underline{t})) \leq \mathrm{Sp}(f(\vec{m}_1;\underline{f(\vec{m}_2;\underline{t})})) \tag{11}$$

for some $\vec{m}_1,\vec{m}_2 \prec^k \vec{m}$ and a normal form $\underline{f(\vec{m}_2;\underline{t})}$ of $f(\vec{m}_2;\underline{t})$. Again, by Lemma 2.9, for each $j=1,2$,

$$\sum(d,|\vec{m}_j|) < \sum(d,|\vec{m}|). \tag{12}$$

Hence, by SIH for $\vec{m}_1$,

$$\begin{aligned}2^{\mathrm{Sp}(f(\vec{m}_1;\underline{f(\vec{m}_2;\underline{t})}))} &\leq (2^{c(\sum(d,|\vec{m}_1|)+1)} + 2^{|\underline{f(\vec{m}_2;\underline{t})}|+1})^{c(\sum(d,|\vec{m}_1|)+1)} \\ &\leq (2^{c\sum(d,|\vec{m}|)} + 2^{|\underline{f(\vec{m}_2;\underline{t})}|+1})^{c(\sum(d,|\vec{m}|)+1)}\end{aligned} \tag{13}$$

by (12). On the other hand, by Lemma 2.10,

$$\begin{aligned}2^{|\underline{f(\vec{m}_2;\underline{t})}|+1} &\leq 2^{c'(\max|\vec{m}_2|^{d'}+1)+1} + 2^{|\underline{t}|+1} \\ &\leq 2^{c(\max|\vec{m}_2|^d+1)} + 2^{|\underline{t}|+1} \text{ by } c'+1 \leq c \text{ and } d' \leq d, \\ &\leq 2^{c\sum(d,|\vec{m}|)} + 2^{|\underline{t}|+1},\end{aligned} \tag{14}$$

since $\max|\vec{m}_2|^d + 1 \leq \sum(d,|\vec{m}|)$. Combining (13) and (14), we obtain

$$\begin{aligned}2^{\mathrm{Sp}(f(\vec{m}_1;\underline{f(\vec{m}_2;\underline{t})}))} &\leq (2^{c\sum(d,|\vec{m}|)} + 2^{c\sum(d,|\vec{m}|)} + 2^{|\underline{t}|+1})^{c(\sum(d,|\vec{m}|)+1)} \\ &= (2^{c\sum(d,|\vec{m}|)+1} + 2^{|\underline{t}|+1})^{c(\sum(d,|\vec{m}|)+1)} \\ &\leq (2^{c(\sum(d,|\vec{m}|)+1)} + 2^{|\underline{t}|+1})^{c(\sum(d,|\vec{m}|)+1)}\end{aligned} \tag{15}$$

by $1 \leq c$. Now (10) follows from (11) and (15).

Above argument is slightly extended to the case for the general form of SNRN. This completes the case for SNRN and, therefore, the proof of the theorem. □

Remark 2.12. As mentioned in Remark 1.5, if a more general form (1) of SNRN is taken instead, the proof of Theorem 2.11 does not work. However, with the use of a stronger rewriting strategy, e.g. head reduction, a similar argument will work.

Corollary 2.13. *If $f \in \mathbf{N}_{normal}$, then the space required to compute $f(\vec{x};)$ is bounded by $p(|\vec{x}|)$ for some polynomial p, i.e., $\mathbf{N}_{normal} \subseteq \mathbf{FPS}$.*

Proof. Assume that $f \in \mathbf{N}_{normal}$ and $\vec{m} \in \mathbb{N}$. Let us identify f with the corresponding function symbol in $\mathcal{F}$ and $\vec{m}$ with the corresponding numerals in $\mathcal{T}(\mathcal{F})$. Recall that $\mathrm{Sp}(m) = |m|$ for any numeral m. This together with Theorem 2.11 yields a polynomial p such that $\mathrm{Sp}(f(\vec{m};)) \leq p(|\vec{m}|)$. From this, there exists an algorithm that rewrites an arbitrary ground term into its reduct running in polynomial time and hence in polynomial space. And also, any normal form $\underline{f(\vec{m};)}$ of $f(\vec{m};)$ is a numeral and there is a normal form $\underline{f(\vec{m};)}$ such that $\pi(\overline{f(\vec{m};)}) = \pi(\underline{f(\vec{m};)})$. Therefore, we can construct a non-deterministic Turing program which computes f in polynomial space. Recall here Savitch's Theorem which states $\mathbf{NPSPACE} \subseteq \mathbf{PSPACE}$. This yields $\mathbf{FNPS} \subseteq \mathbf{FPS}$, and hence $f \in \mathbf{FPS}$. □

3. PSPACE-commutable functions belong to N

In this section we show that every function from **FPS** is a member of $\mathbf{N}_{normal}$. The proof is divided into two steps.

First we simulate a computation of a $O(|x|^k)$-space-bounded Turing machine by an $2^{O(|x|^l)}$-step-bounded register machine working over the unary representation (for some l). This is done by imitating an argument in Handley and Wainer 6, in which every $O(|x|)$-space Turing computation is simulated by a $2^{O(|x|)}(\approx O(x^k)$ for some $k)$-step-bounded register machine on unary notation.

Then, we simulate an action of the exponentially bounded register machine by functions in $\mathbf{N}_{normal}$. The argument is similar to one in Arai and the author 4, in which an exponential-time Turing computation is arithmetized with the use of SNRN. We notice that the situation is, however, easier. This is due to the fact that the class **PSPACE** is closed under composition whereas **EXP** is not.

To simulate computations within **N**, the closedness under a simultaneous SNRN is necessary:

Lemma 3.1. (Cf. 4 Theorem 3.1) *Suppose that $f_1, \ldots, f_l$ are defined from $h_1, \ldots, h_l \in \mathbf{N}^{k',l}$ and $\prec^k$-functions $\mathbf{f}_1, \mathbf{f}_2$ by a scheme of simultaneous*

SNRN such that for each $i = 1, \ldots, l$ *and* $w \in \Sigma^k$,

$$\begin{cases} f_i(\vec{0}, \vec{x}; \vec{a}) = h_i(\vec{x}; \vec{a}), \\ f_i(C_w(\vec{y};\,), \vec{x}; \vec{a}) = f_i(\vec{v}_1, \vec{x}; f_1(\vec{v}_2, \vec{x}; \vec{a}), \ldots, f_l(\vec{v}_2, \vec{x}; \vec{a})). \end{cases}$$

Then, for any $g_1, \ldots, g_l \in \mathbf{N}^{k',0}$ *and for each* $i = 1, \ldots, l$,

$$f_i(\vec{y}, \vec{x}; g_1(\vec{x};\,), \ldots, g_l(\vec{x};\,)) \in \mathbf{N}^{k+k',0}.$$

Proof. Although the usual simultaneous recursion is easily reduced to a single recursion by a pairing and unpairing functions, it is not a trivial task in the safe representation. Bellantoni[5] defines a pairing function in the Bellantoni-Cook class $\mathcal{B}$. This pairing function is not definable in $\mathbf{N}$, since the i-concatenation C_i is available only on the normal argument. A familiar pairing function $\langle x, y\rangle = \frac{1}{2}(x+y)(x+y+1)+x$ is definable in $\mathbf{N}_{normal}$, while it is not clear for the author whether unpairing functions for $\langle\cdot,\cdot\rangle$ can be defined in $\mathbf{N}_{normal}$. Therefore, we instead employ the following paring function.

For the function $\times$ in Example 1.6(3), $\times(x, y; a) = y \cdot 2^{|x|} + a = y\underbrace{0\ldots0}_{|x|-|a|}a$ if $|a| \leq |x|$. Hence we define a pairing function $\pi \in \mathbf{N}^{3,0}$ by

$$\pi(x, y_0, y_1;\,) = \times(x, y_0; y_1),$$

which works as a pairing function if $|y_1| \leq |x|$. For the function $\dot{-}$ in Example 1.6(1), the corresponding unpairing function π_0 is defined by

$$\pi_0(x, y;\,) = \dot{-}(x; y),$$

which works as $\pi_0(x, \pi(x, y_0, y_1;\,);\,) = y_0$. We mention that the deletion function D is needed only to define π_0. And another one π_1 is defined by

$$\pi_1(x, y;\,) = y \dot{-} \pi_0(x, y;\,) \cdot 2^{|x|}.$$

which works as $\pi_1(x, \pi(x, y_0, y_1;\,);\,) = \pi(x, y_0, y_1;\,) \dot{-} y_0 \cdot 2^{|x|} = y_1$.

Let $p(\vec{x}, \vec{y})$ be a polynomial such that

$$\max\{|f_i(\vec{y}, \vec{x}; \vec{g}(\vec{x};\,))| : i = 1, \ldots, l\} \leq p(|\vec{x}|, |\vec{y}|).$$

The existence of the polynomial $p(\vec{x}, \vec{y})$ is guaranteed by Lemma 2.10. Then, $\pi(2^{p(|\vec{x}|,|\vec{y}|)}, z_0, z_1;\,)$ works as a pairing function for $f_i(\vec{y}, \vec{x}; \vec{g}(\vec{x};\,))$ $(i = 1, \ldots, l)$, and $\pi_i(2^{p(|\vec{x}|,|\vec{y}|)}, z;\,)$ $(i = 0, 1)$ as the corresponding unpairing functions. Recall Example 1.6(5) here. Since $2^{p(|\vec{x}|,|\vec{y}|)}$ is defined in $\mathbf{N}_{normal}$, we thus conclude $f_i(\vec{y}, \vec{x}; \vec{g}(\vec{x};\,)) \in \mathbf{N}^{k+k',0}$ for each $i = 1, \ldots, l$. □

Theorem 3.2. (Cf. 4 Theorem 4.1) $\mathbf{FPS} \subseteq \mathbf{N}_{normal}$.

Proof. Assume that $f \in \mathbf{FPS}$ and the arity of f is k. Let $\vec{x} = x_1, \ldots, x_k$ be inputs. Then the number of possible configurations is bounded by $2^{p(|\vec{x}|)}$ for some polynomial $p(\vec{x})$. Hence $f(\vec{x})$ is computed within a number of steps bounded by $2^{p(|\vec{x}|)}$, since the same configuration does not repeat in any terminating computations. As in the proof of Theorem 3.9 in 6, actions of the Turing machine can be simulated by a register machine working over the unary representation as follows.

- One register contains the number representing the tape-configuration to the left of the reading head, and
- another register contains the number representing the tape-configuration to the right of the reading head.

We consider the following (unlimited) register machines. A register machine $\max|\vec{m}|$ has registers $R_0, R_1, \ldots$ which store natural numbers $r_0, r_1, \ldots$. A register machine program is a finite list $\{I_j : j = 1, \ldots, j_M\}$ of instructions.

- Each instruction has one of four basic types: (Zero) $Z(k)$; $r_k = 0$, (Successor) $S(k)$; $r_k = r_k{+}1$, (Predecessor) $P(k)$; $r_k = r_k \dot{-} 1$, (Transfer) $T(k,l)$; $r_k = r_l$, (Jump) $J(k,l,j)$; if $r_k = r_l$ then go to instruction I_j else go to next instruction.
- When a computation starts, the inputs $x_1, \ldots, x_k$ are stored in registers $R_1, \ldots, R_k$, respectively, and 0 in all the other registers.
- When the computation halts, the output is the number in register R_0.

Functions like $x \mapsto 2x$, $x \mapsto 2x+1$ and $x \mapsto \lfloor x/2 \rfloor$ are computed by a register machine within a number of steps bounded by the exponential of a polynomial in $|x|$. Therefore, $f(\vec{x})$ is computed by a register machine M within $2^{q(|\vec{x}|)}$-steps for some polynomial $q(\vec{x})$.

Now we simulate this computation in $\mathbf{N}$. For a constant $l \geq 3 + k$ depending on M, information on M in step $|y|$ of the computation on inputs $\vec{x}$ is represented by some functions $f'_j(y, \vec{x};)$ $(j = 1, \ldots, l)$, which are defined by simultaneous safe recursion on notation of the form

$$\begin{cases} f'_j(0, \vec{x};) = g_j(\vec{x};), \\ f'_j(C_i(y;), \vec{x};) = h_j(; f_1(y, \vec{x};), \ldots, f_l(y, \vec{x};)) \quad (i = 0, 1) \end{cases}$$

for each $j = 1, \ldots, l$. For the sake of completeness, we give an outline of this arithmetization. Let us assume a (finite) encoding $\lceil I \rceil$ for an instruction I.

We can define the following functions in $\mathbf{N}_{normal}$:

$$\begin{aligned} index(y,\vec{x};) &= \text{the index of the instruction performed next,}\\ inst(y,\vec{x};) &= \text{a code } \lceil I\rceil \text{ of the instruction } I \text{ performed next,}\\ r_j(y,\vec{x};) &= \text{the number } r_j \text{ stored in the register } R_j \text{ in step } |y|.\\ &\quad (j=0,\dots,l-3) \end{aligned}$$

Namely, $f'_1 = index$, $f'_2 = inst$, and $f'_j = r_{j-3}$ for each $j = 3,\dots,l$. For the base case of recursion, $index(0,\vec{x};) = 1$, $inst(0,\vec{x};) = \lceil I_1\rceil$, and

$$r_j(0,\vec{x};) = \begin{cases} 0 & \text{if } j=0,\\ x_j & \text{if } 1\le j\le k,\\ 0 & \text{otherwise.}\end{cases}$$

The recursion step is of the form

$$\begin{aligned} index(C_i(y;),\vec{x};) &= \delta_1(;index(y,\vec{x};),inst(y,\vec{x};)),\\ inst(C_i(y;),\vec{x};) &= \delta_2(;index(y,\vec{x};),inst(y,\vec{x};)),\\ r_j(C_i(y;),\vec{x};) &= \delta_{3+j}(;inst(y,\vec{x};),r_0(y,\vec{x};),r_1(y,\vec{x};),\dots,r_l(y,\vec{x};)), \end{aligned}$$

where $\delta_1,\dots,\delta_l$ are determined according to the program, which depend also on the encoding $\lceil\cdot\rceil$ for the instructions. Obviously, each δ_j is defined only on safe arguments from O, S, P, I, C by a suitable number of application of safe composition. For the same reason, the above recursion step does not depend on $i = 0, 1$ of $C_i(y;)$.

By the output convention, $f_3(t,\vec{x};) = r_0(t,\vec{x};) = f(\vec{x})$ if $|t| \ge 2^{q(|\vec{x}|)}$. Next, via the functions $h_1,\dots,h_l$, we define functions $f_j(y,\vec{x};\vec{a})$ $(j = 1,\dots,l)$ by the following equations of simultaneous SNRN:

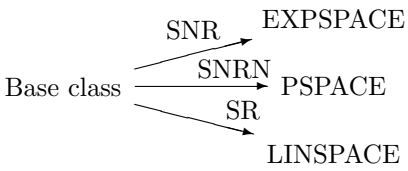
$$\begin{cases} f_j(0;\vec{a}) = h_j(;\vec{a}),\\ f_j(C_i(y;);\vec{a}) = f_j(y;f_1(y;\vec{a}),\dots,f_l(y;\vec{a})) \quad (i=0,1)\end{cases}$$

Lemma 3.1 ensures that $F_j(y,\vec{x};) := f_j(y;g_1(\vec{x};),\dots,g_l(\vec{x};)) \in \mathbf{N}_{normal}$ for each $j = 1,\dots,l$. By the definition, the functions $F_j(y,\vec{x};)$ $(j = 1,\dots,l)$ represent information on M in step $2^{|y|}$. Therefore, $F_3(t,\vec{x};) = r_0(2^t,\vec{x};) = f(\vec{x})$ whenever $t \ge 2^{q(|\vec{x}|)}$. Thus an application of safe composition yields that $f(\vec{x}) = F_3(2^{q(|\vec{x}|)},\vec{x};) \in \mathbf{N}_{normal}$. □

We mention that the formation of $F_3(2^{q(|\vec{x}|)},\vec{x};)$ is now easy contrary to the construction in the proof of Theorem 4.1 in 4. This is due to the formulation of safe composition. A more restrictive composition of the form $f(\vec{x};\vec{a}) = h(\vec{I}^{k,0}_j(\vec{x};);\vec{\varphi}(\vec{x};\vec{a}))$ for projection functions $I^{k,0}_{j_1},\dots,I^{k,0}_{j_k}$ is necessary in 4.

Conclusions and final comments

In this paper we give a term-rewriting characterization of the polyspace functions. As a consequence, a recursion-theoretic characterization of the polyspace functions is obtained. This suggests a uniform approach to the space complexity in the sense that adding specific safe recursion schemes to a suitable base class capture various space complexity classes (cf. Remark 1.8):



We introduce a two-sorted function class $\mathbf{N}^{k,l}$ such that $\mathbf{N}_{normal} = \mathbf{FPS}$. The present work is inspired by Oitavem's works, which also give a term-rewriting and recursion-theoretic characterization of **FPS**. However, it is not clear for the author how to prove the equality of $\mathbf{N}$ and the Oitavem class. Nor proper inclusion relation between $\mathbf{N}$ and $\mathcal{N}$ in 4 which captures the exptime functions. For example, it seems quite difficult to simulate safe nested recursion within Oitavem's formulation as well as the converse is. These are future works.

The research in such a framework has been initiated by Cichon and Weiermann,[7] and followed by Beckmann and Weiermann.[8] The underlying idea in 7 and 8 is that term rewriting can be used to prove inclusion relations between complexity classes. In particular, non-trivial closure properties are obtained. An application in this direction will be to prove Savitch's Theorem in the present context (as suggested by the referee).

Clearly, the present result is related to a work by Bonfante, Marion and Moyen.[9] In 9 it is shown that if a program is polynomially quasi-interpretable and its termination is proved via the lexicographic path order LPO, then the program is polyspace-computable. This together with the proof of Lemma 2.2 suggests that the TRS $R_{\mathbf{N}}$ can be polynomially quasi-interpreted (assuming a suitable rewriting strategy).

Acknowledgments

First of all, I would like to acknowledge Georg Moser. He called my attention to Oitavem's works[1,2] and a related work by M. Avanzini.[10] He also pointed out some errors in an earlier draft. I would like to thank the referee for

careful reading and invaluable comments. In particular, the introduction of the non-confluent TRS $R_{\mathbf{N}}$ and the use of the innermost strategy are due to him or her. I would also like to thank my thesis advisor Prof. Toshiyasu Arai for his comments on this work.

References

1. I. Oitavem, A term rewriting characterization of the functions computable in polynomial space, *Archive for Mathematical Logic* **41**, 35 (2002).
2. I. Oitavem, New recursive characterizations of the elementary functions and the functions computable in polynomial space, *Revista Mathematica de la Universidad Complutense de Madrid* **10**, 109 (1997).
3. S. Bellantoni and S. Cook, A new recursion-theoretic characterization of the polytime functions, *Computational Complexity* **2**, 97 (1992).
4. T. Arai and N. Eguchi, A new function algebra of EXPTIME functions by safe nested recursion, to appear in *ACM Transactions on Computational Logic* .
5. S. Bellantoni, Predicative recursion and computational complexity, PhD thesis, University of Toronto1992.
6. W. G. Handley and S. S. Wainer, Complexity of primitive recursion, in *Computational Logic, NATO ASI Series F: Computer and Systems Science*, eds. U. Berger and H. Schwichtenberg (Springer, 1999) pp. 273–300.
7. E. A. Cichon and A. Weiermann, Term rewriting theory for the primitive recursive functions, *Annals of Pure and Applied Logic* **83**, 199 (1997).
8. A. Beckmann and A. Weiermann, A term rewriting characterization of the polytime functions and related complexity classes, *Archive for Mathematical Logic* **36**, 11 (1996).
9. G. Bonfante, J.-Y. Marion and J.-Y. Moyen, On lexicographic termination ordering with space bounded certifications, *Perspectives of system informatics, Lecture Notes in Computer Science* **2244**, 482 (2001).
10. M. Avanzini, *Term rewriting characterizations of complexity classes*, seminar report, University of Innsbruck, Institute of Computer Science (July 2007).