

# A Decidable Characterization of the Classes between Lintime and Exptime

Salvatore Caporaso

Dipartimento di Informatica dell'Università di Bari

caporaso@ di.uniba.it

**Abstract** A language is defined by closure under *safe iteration* and under a new form of *safe diagonalization* that, unlike other forms of diagonalization used in literature to define sub-recursive hierarchies, is constructive and decidable. By counting the nesting levels of these schemes, an ordinal is assigned to each program. This yields a hierarchy  $\mathcal{T}_\alpha$  ( $\alpha < \omega^\omega$ ) that singles-out the complexity classes  $\text{DTIMEF}(n^{cn^d+e})$  for all  $c, d, e \geq 0$ .

## 1 Introduction

**1 Position of the Problem** Complexity may be better understood in terms of *resource-free operators*, than of heterogeneous couples of the form TM + *clock* or *meter*. Insight may be further increased if such operators produce hierarchies collecting and connecting, under a same criterion, as many classes as possible.

In the last decade a number of imperative languages have been defined that characterize certain classes by closure under resource-free schemes of a small stock of basic operators, not including *ad hoc* functions; and that are decidable, in the sense that one can read-off the computational complexity of a program from its syntax. After the celebrated results [2, 8] on  $\text{PTIMEF}$ , all other classes captured in this way deal with polynomial resources:  $\text{PSPACEF}$  [10],  $\text{PTIME}$ ,  $\text{PSPACE}$ ,  $\text{PH}$  [4],  $\text{LINSPECF}$  [1],  $\text{LINTIMEF}$ ,  $\text{PTIMEF}$  and  $\text{NP}$  [3] (see the proceedings of the yearly Implicit Computational Complexity (ICC) workshops 1997-2001 for other results). Moreover, one finds hierarchies that single-out the classes  $\text{DTIMEF}(n^{2^c})$  in [9], and  $\text{DTIMESPECF}(n^c, n^d)$  in [6].

In our previous contributions to the ICC program aimed at extending the ICC approach to higher classes ([5] and elsewhere), we borrowed the *diagonalization*

definition scheme from the theory of provable subrecursivity (see [7] for a systematic presentation). Like in the transfinite Grzegorzcyk hierarchy, one obtains function  $f(n)$  at  $\lambda$  if, for a function  $g$  defined at  $\lambda[1]$  (Rogers notations)

$$f(n) = \varphi_{g(n)}(n) \quad \text{and each } \varphi_{g(n)} \text{ has been defined at } \lambda[n].$$

This method however presents two serious drawbacks. One *can* generate constructively new functions but *cannot* decide uniformly whether  $g$  does actually enumerate functions defined at the appropriate stages  $\lambda[n]$ . Moreover, an *apply* function is implicit in this scheme, but it is not clear the ordinal of such a universal function.

**2 Statement of the Result** In this paper we introduce two resource-free operators, and show that their modulation yields a hierarchy that singles-out several subclasses of PTIMEF and EXPTIMEF. To this purpose, a (class of) language(s)  $\mathbf{P}_{\mathbf{A}}$  is defined by closure of a finite and generic stock  $\mathbf{A}$  of acyclic programs, under *safe iteration* and *safe diagonalizations* (see Lemma §15 for consistence of this name with subrecursive terminology). An  $\omega^\omega$ -type hierarchy  $\mathcal{T}_\alpha$  is obtained by assigning to  $\mathcal{T}_{\omega^d c + e}$  all functions that can be computed with  $e$  iterations,  $c$  diagonalizations, and diagonalization depth  $d$  (for some  $\mathbf{A}$ ).

**Notation**  $C(f(n)) =_1 \mathcal{T}$  means that  $\mathcal{T}$  sandwiches between  $C(f(n))$  and  $C(f(n+1))$ .

**Theorem** For all  $c, d$  and  $e \geq 0$  we have  $\text{DTIMEF}(n^{cn^d+e}) =_1 \mathcal{T}_{\omega^d c + e}$ .

*Proof.* By simulation (see Lemmata 18 and 17).

For example, we have  $\mathcal{T}_1 = \text{LINTIMEF}$ ,  $\mathcal{T}_{<\omega} = \text{PTIMEF}$ ,  $\mathcal{T}_{<\omega^2} = \text{DTIMEF}(n^{O(n)})$  and  $\mathcal{T}_{<\omega^\omega} = \text{DTIMEF}(n^{n^{O(1)}})$ .

To get easier definitions and proofs, a very simple form of iteration has been adopted. However, all results keep holding with more sumptuous (safe) recursion schemes.

## 2 Polynomial Time

**3 Acyclic Programs** Assume given a generic class  $\mathbf{D}$  of *data*  $x$  of length  $|x|$ . Let  $\mathbf{A}$  denote an alphabet whose letters are (associated with) generic *acyclic programs* (AP) such that for each  $\mathbf{A} \in \mathbf{A}$  (1) we have  $\llbracket \mathbf{A} \rrbracket : \mathbf{D} \mapsto \mathbf{D}$ ; and (2) there exists a TM that simulates  $\mathbf{A}$  in a constant time.

**4 Example** One can define (see for example [6, §2.1 and 3]) for any TM  $M$  the classes  $\mathbf{D}$  and  $\mathbf{A}$ , and an AP  $\text{NXT}_M \in \mathbf{A}$  such that (for an adequate representation

$\rho D$  of the instantaneous descriptions (ID)  $D$  of  $M$ )

$$\text{NXT}_M(\rho D) = \begin{cases} \rho(D_1) & \text{if } D_1 \text{ is the next ID after } D \\ \rho(D) & \text{if } D \text{ is final.} \end{cases}$$

Moreover, since  $\mathbf{A}$  is finite, there exists  $\mathbf{A}^{(c)}$  that includes, among all compositions of  $c$  APs belonging to  $\mathbf{A}$ , the AP  $\text{NXT}_{M^c}$  which returns the ID of  $M$  after  $c$  steps.

**5 Notation** (1) All symbols in **typewriter** font are programs of the language we are defining; in particular, capitals will denote APs.

(2) According to context,  $E^n$  is  $E$  to the  $n$ , or is the string  $E$  repeated  $n$  times.  $f \uparrow n(s, t)$  is the  $n$ th *left* (outer) *iterate* of  $f(s, t)$ , defined by ( $t$  possibly absent)

$$f \uparrow 0(s, t) = s; \quad f \uparrow n + 1(s, t) = f(f \uparrow n(s, t), t).$$

(3) Note that we obviously have  $(f \uparrow n) \uparrow m(s, t) = f \uparrow nm(s, t)$  for  $n, m > 0$ .

**6 Language for Polytime** If  $\mathbf{p}$  is a program then expression

$$\mathbf{p} \mathbf{I}$$

is a program, defined by *safe iteration* in  $\mathbf{p}$ , under the following semantics

$$\begin{cases} \llbracket \mathbf{A} \mathbf{I} \rrbracket(s, t) & = \llbracket \mathbf{B} \rrbracket \uparrow |t|(s) \\ \llbracket \mathbf{A} \mathbf{I}^{c+1} \rrbracket(s, t) & = \llbracket \mathbf{A} \mathbf{I}^c \rrbracket \uparrow |t|(s, t) \end{cases}$$

**7 Example** Take the unary numerals  $1^n = \bar{n}$  for  $n > 0$  as  $\mathbf{D}$ , and let  $\mathbf{S} \in \mathbf{A}$  compute the successor. We show that  $\mathbf{S} \mathbf{I}^d(\bar{n}, \bar{m})$  repeats  $\mathbf{S}$  for  $m^d$  times. Induction on  $d$ . Basis.  $\llbracket \mathbf{S} \mathbf{I} \rrbracket(\bar{n}, \bar{m}) = \llbracket \mathbf{S} \rrbracket \uparrow m(\bar{n}, \bar{m}) = \overline{n + m}$ . Step.

$$\llbracket \mathbf{S} \mathbf{I}^{d+1} \rrbracket(\bar{n}, \bar{m}) = \llbracket \mathbf{S} \mathbf{I}^d \rrbracket \uparrow m(\bar{n}, \bar{m}) =_{\text{I.H.}} (\llbracket \mathbf{S} \rrbracket \uparrow m^d) \uparrow m(\bar{n}, \bar{m}) =_{\S 5(3)} \overline{n + m^{d+1}}.$$

**8 The Poly-Time Classes** Define

$$\begin{aligned} \mathcal{T}_1 &= \{f : f(s) = \llbracket \mathbf{A} \mathbf{I} \rrbracket \text{ for some } \mathbf{A} \text{ in some } \mathbf{A}\}; \\ \mathcal{T}_{c+1} &= \{f : f(s) = \llbracket \mathbf{p} \mathbf{I} \rrbracket(s, s) \text{ for some } \llbracket \mathbf{p} \rrbracket \in \mathcal{T}_c\}. \end{aligned}$$

**9 Lemma**  $\text{DTIMEF}(n^c) = \mathcal{T}_c$ .

*Proof.*  $\subseteq$ . Let  $F : \mathbf{D} \mapsto \mathbf{D}$  be computed by the TM  $M$  in time  $dn^c$ . From Ex. 4, by arguments like in Ex. 7, we see that  $\llbracket \text{NXT}_{M^d} \mathbf{I}^c \rrbracket(s, t)$  returns the ID of  $M$  after  $d|t|^c$  steps. We then have  $F(s) = \llbracket \text{NXT}_{M^d} \mathbf{I}^c \rrbracket(s, s)$ .

$\supseteq$ . By Lemma 18, after observing that the *spread* 1 in the statement of the theorem vanishes when we talk about PTIMEF.

### 3 Exponential Time

**10 Language** Let  $X, Y, \dots$  denote strings over  $\{\mathbf{I}, \langle, \rangle\}$  whose angle parentheses, if any, properly match around a not empty *scope*.

Given  $\mathbf{A}$ , define  $\mathbf{P}_{\mathbf{A}}$  to be the class of all *programs* in the form

$$\mathbf{A} X \quad (\mathbf{A} \in \mathbf{A})$$

where program of the form  $\mathbf{p}\langle X \rangle$  is defined by *safe diagonalization* in  $\mathbf{p}X$ , under the following semantics

$$\llbracket \mathbf{p}\langle X \rangle \rrbracket(s, t) = \llbracket \mathbf{p}X^{|\mathbf{t}|} \rrbracket(s, t).$$

**11 Example 7 continued** We have ( $\cdot$  is short for  $\bar{n}, \bar{m}$ ) and

$$\begin{aligned} \llbracket \mathbf{S}\langle \mathbf{I}\mathbf{I} \rangle \rrbracket(\bar{n}, \bar{m}) &= \llbracket \mathbf{S}\mathbf{I}^{2m} \rrbracket(\cdot) &= \overline{n + m^{2m}} && \text{by Ex. 7 with } 2m \text{ as } d \\ \llbracket \mathbf{S}\langle \mathbf{I} \rangle \langle \mathbf{I} \rangle \rrbracket(\cdot) &= \llbracket \mathbf{S}\langle \mathbf{I} \rangle \mathbf{I}^m \rrbracket(\cdot) &= \llbracket \mathbf{S}\langle \mathbf{I} \rangle \mathbf{I}^{m-1} \rrbracket \uparrow m(\cdot) &= \llbracket \mathbf{S}\langle \mathbf{I} \rangle \mathbf{I}^{m-2} \rrbracket \uparrow m^2(\cdot) \\ &= \llbracket \mathbf{S}\langle \mathbf{I} \rangle \rrbracket \uparrow m^m(\cdot) &= \llbracket \mathbf{S}\mathbf{I}^{m-1} \rrbracket \uparrow m^{m+1}(\cdot) &= \llbracket \mathbf{S} \rrbracket \uparrow m^{2m}(\cdot) \\ &= \overline{n + m^{2m}} \\ \llbracket \mathbf{S}\langle \langle \mathbf{I} \rangle \rangle \rrbracket(\cdot) &= \llbracket \mathbf{S}\langle \mathbf{I} \rangle^m \rrbracket(\cdot) &= \llbracket \mathbf{S}\langle \mathbf{I} \rangle^{m-1} \rrbracket \uparrow m^m(\cdot) &= \llbracket \mathbf{S}\langle \mathbf{I} \rangle^{m-2} \rrbracket \uparrow m^{2m}(\cdot) \\ &= \overline{n + m^{m^2}}. \end{aligned}$$

Hereinafter  $\mathbf{p}(s, t)$  is short for  $\llbracket \mathbf{p} \rrbracket(s, t)$ .

**12 Ordinals** Let  $\alpha, \beta, \dots, \lambda, \mu$  denote ordinals and limit ordinals below  $\omega^\omega$ . We write  $\alpha = (\beta, \gamma)_{\text{NF}}$  to mean that we have  $\beta = 0$  or the lowest addend in Cantor Normal Form (NF) for  $\beta$  is not below the highest one of  $\gamma$ . Assign a *fundamental sequence* to  $\lambda = (\alpha, \omega^{c+1})_{\text{NF}}$  by  $\lambda[n] = \alpha + \omega^c n$ .

The *commutative sum* of  $\alpha$  and  $\beta$  is obtained by sorting according to size the addends of their NF. That is, given  $\alpha = (\gamma, \omega^c)_{\text{NF}}$  and  $\beta = (\delta, \omega^d)_{\text{NF}}$  define

$$\alpha \# \beta = \begin{cases} \gamma \# \beta + \omega^c & \text{if } d > c \\ \alpha \# \delta + \omega^d & \text{otherwise.} \end{cases}$$

For example, we have  $(\omega \# (\omega^3 + \omega^2))[1] = \omega^3 + \omega^2 + \omega[1] =_{\text{NF}} \omega^3 + \omega^2 + 1$ .

**13 Program Ordinal and Length** The *length*  $|X|$  of  $X$  is the number of  $\mathbf{I}$  occurring in  $X$ . Let  $\mathbf{I}_{(i)}$  denote the  $i$ -th occurrence of  $\mathbf{I}$  in a given  $X$  ( $1 \leq i \leq |X|$ ). The *nesting depth*  $\Delta_i$  of  $\mathbf{I}_{(i)}$  is the number of couples  $\langle \cdot \rangle$  having  $\mathbf{I}_{(i)}$  in their scope. Define the *ordinal* of any  $\mathbf{p}$  and  $X$  by

$$oX = \omega^{\Delta_1} \# \dots \# \omega^{\Delta_{|X|}}; \quad o\mathbf{A}X = oX; \quad |\mathbf{A}X| = |X|.$$

For example, we have  $o\mathbf{A}\mathbf{I}((\mathbf{I}\mathbf{I}\mathbf{I})\mathbf{I})(\mathbf{I}) = \omega^2\mathfrak{3} + \omega 2 + 1$  and  $|\mathbf{A}\mathbf{I}((\mathbf{I}\mathbf{I}\mathbf{I})\mathbf{I})(\mathbf{I})| = 6$ . The ordinal of the first two programs of §11 is  $\omega 2$ , and  $\omega^2$  is the ordinal of the third.

**14 The Hierarchy** Define  $(0 < \alpha < \omega^\omega)$

$\mathcal{T}_\alpha = \{f : f(s) = \llbracket \mathbf{A} X \rrbracket (s, s) \text{ for some } \mathbf{A}, \mathbf{A} \text{ and } X \text{ such that } \mathbf{A} \in \mathbf{A} \text{ and } oX \leq \alpha\}$ .

Thus all  $f \in \mathcal{T}_\alpha$  ( $\alpha > 1$ ) are obtained by identifying the two variables of the interpretation of a program whose ordinal is  $\alpha$ .

**15 Lemma**  $o\mathbf{q}\langle X \rangle = \lambda$  implies  $o\mathbf{q}X^n \leq \lambda[n]$ . Hence  $\mathbf{q}\langle X \rangle$  and  $\mathbf{q}X^n$  compute respectively a function  $f \in \mathcal{T}_\lambda$  and a function  $f_n \in \mathcal{T}_\lambda[n]$  such that  $f(n) = f_n(n)$ .

*Proof.* Define  $\lambda = o\mathbf{q}\langle X \rangle$ ,  $h = |\mathbf{q}|$  and  $k = |X|$ . We have to show  $o\mathbf{q}X^n \leq \lambda[n]$  for

$$o\mathbf{q} =_{\text{NF}} \omega^{a_1+1} + \dots + \omega^{a_h+1}; \quad o\langle X \rangle =_{\text{NF}} \omega^{b_1+1} + \dots + \omega^{b_k+1}.$$

Case 1.  $\lambda = (\gamma, \omega^{b_k+1})_{\text{NF}}$  for some  $\gamma$ . We have

$$\begin{aligned} o\mathbf{q}X^n &= (\omega^{a_1+1} + \dots + \omega^{a_h+1})\sharp(\omega^{b_1}n + \dots + \omega^{b_k}n) \\ &\leq (\omega^{a_1+1} + \dots + \omega^{a_h+1})\sharp(\omega^{b_1+1} + \dots + \omega^{b_{k-1}+1}) + \omega^{b_k}n = \lambda[n]. \end{aligned}$$

Case 2.  $\lambda = (\gamma, \omega^{a_h+1})_{\text{NF}}$ . We have

$$\begin{aligned} o\mathbf{q}X^n &= (\omega^{a_1+1} + \dots + \omega^{a_{h-1}+1})\sharp(\omega^{b_1}n + \dots + \omega^{b_k}n) + \omega^{a_h}n \\ &\leq (\omega^{a_1+1} + \dots + \omega^{a_{h-1}+1})\sharp(\omega^{b_1+1} + \dots + \omega^{b_k+1}) + \omega^{a_h}n = \lambda[n] \end{aligned}$$

because  $d < c$  implies  $\eta + \omega^c n + \omega^{d+1} < \eta + \omega^{c+1} + \omega^d n$ .

**16** Define a variant  $B_\alpha$  of the *slow-growing majorization* functions  $G_\alpha$  [7] by

$$B_{\omega^{a_1}\sharp\dots\sharp\omega^{a_k}}(n) = n^{n^{a_1+\dots+n^{a_k}}} \quad (k, a_i \geq 0).$$

Note that we have

- (1)  $B_{\alpha\sharp\beta}(n) = B_\alpha(n)B_\beta(n)$ ;
- (2)  $B_{\omega^{c+1}}(n) = B_{\omega^c n}(n)$ ;
- (3)  $kn \leq B_{\omega^{a_1}\sharp\dots\sharp\omega^{a_k}}(n)$ .

## 4 Simulations

**17 Lemma**  $\text{DTIMEF}(B_\alpha(n)) \subseteq \mathcal{T}_\alpha$ .

*Proof.* Let  $\langle d \rangle$  be short for  $\langle \dots \langle \mathbf{I} \rangle \dots \rangle$  ( $d \geq 0$  diagonalizations). For each  $\alpha$  define the string  $[\alpha]$  by

$$[\omega^{a_1} + \dots + \omega^{a_k}] = \langle a_1 \rangle \dots \langle a_k \rangle \quad (k, a_i \geq 0).$$

We show that for any AP  $\mathbf{A}$  we have

$$\mathbf{A}[\alpha](s, t) = \mathbf{A} \uparrow B_\alpha(|t|)(s, t).$$

The result will follow like in proof of Lemma 9, since we obviously have  $o[\alpha] = \alpha$ . Induction on  $\alpha$ . Basis.  $\mathbf{A}[0] = \mathbf{A}$  and  $B_0(n) = 1$ . Step. Case 1.  $\alpha = \beta + 1$ . We have

$$\begin{aligned} \mathbf{A}[\alpha](s, t) &= \mathbf{A}[\beta] \mathbf{I}(s, t) \\ &= \mathbf{A}[\beta] \uparrow |t|(s, t) && \text{semantics of } \mathbf{I} \\ &= (\mathbf{A} \uparrow B_\beta(|t|)) \uparrow |t|(s, t) && \text{I.H. if } \beta > 0 \\ &= \mathbf{A} \uparrow (B_\alpha(|t|))(s, t) && \S 5(3) \text{ and } \S 16(1) \text{ with } \beta = 1. \end{aligned}$$

Case 2.  $\alpha = \beta + \omega^{d+1}$  ( $\beta, d \geq 0$ ). We have

$$\begin{aligned} \mathbf{A}[\alpha](s, t) &= \mathbf{A}[\beta] \langle d+1 \rangle(s, t) \\ &= \mathbf{A}[\beta] \langle d \rangle^{|t|}(s, t) && \text{semantics of diagonalization} \\ &= \mathbf{A}[\beta + \omega^d |t|](s, t) && \text{def. of } [\beta + \omega^d + \dots + \omega^d] \\ &= \mathbf{A} \uparrow B_{\beta + \omega^d |t|}(|t|)(s, t) && \text{I.H.} \\ &= \mathbf{A} \uparrow B_\alpha(|t|)(s, t) && \S 16(2). \end{aligned}$$

**18 Lemma**  $\mathcal{T}_\alpha \subseteq \text{DTIMEF}(B_\alpha(n+1))$ .

*Proof. Construction* Since diagonalizations involve variable programs, we need an interpreter rather than a single TM for each  $\mathbf{p}$ . Let  $\mathbf{A}$  be given, together with a TM  $M_{\mathbf{A}}$  with  $T$  tapes over an alphabet  $\Gamma \supset \mathbf{A}$  that, by input any  $\mathbf{A} \in \mathbf{S}$  and  $x$ , returns  $\mathbf{A}(x)$ . Let  $\mathbf{p}^*$  denote the string over an alphabet  $\Theta$  of cardinality  $(c+4)\text{card}(\Gamma)$ , that codes  $\mathbf{p}$ , by associating each  $\mathbf{I}_{(i)}$  with a letter  $\mathbf{I}_{(i)}^* \in \Theta$  that encodes the nesting depth  $\Delta_i$ , and the value  $\mathbf{I}, \langle, \rangle, \mathbf{A}$  of its left and right neighbours. For each  $c$  one can define a TM  $\text{INT}_{\mathbf{A}c}$  which, by input  $\mathbf{p}^*$ ,  $s$  and  $t$  ( $o\mathbf{p} < \omega^{c+1}$ ) returns  $\mathbf{p}(s, t)$ . Each  $\text{INT}$  is supplied with  $T+4$  tapes over  $\Theta$ . Tapes  $1, \dots, T$  are used for the calls to  $M_{\mathbf{A}}$ ;  $T+1$  is permanently assigned with  $\bar{n} := 1^{|t|}$ ;  $T+2$  and  $T+3$  are for scratch; in  $T+4$  a stack  $\Sigma$  of records  $\mathbf{p}_1^* \dots \mathbf{p}_k^*$  is stored. The behaviour of  $\text{INT}_{\mathbf{A}c}$  is outlined

---

by input  $\mathbf{p}$ ,  $s$  and  $t$   
 $\Sigma := \mathbf{p}$ ;  $n := |t|$   
while  $\Sigma$  is not empty  
  case of  $\text{top}(\Sigma)$   
     $\mathbf{A I}$  : apply  $M_{\mathbf{A}}$  for  $n$  times; pop  $\mathbf{A I}$ ;  
     $\mathbf{I}$  : pop  $\mathbf{q I}$ ; push  $\mathbf{q}^n$ ;  
     $\rangle$  : pop  $\langle X \rangle$ ; push  $X^n$ .

---

Figure 1: The Interpreter

in Fig.1. Hereinafter we identify any  $\mathbf{p}$  with its code  $\mathbf{p}^*$  and any  $l \in \Theta$  with the information it conveys.

**Complexity of the invariant** Note that:

- (1) since each  $\mathbf{A}$  is finite, we may assume that  $M_{\mathbf{A}}$  takes one step;
- (2)  $INT$  can pop in  $|\mathbf{q}|$  steps the scope of an iteration  $\mathbf{q I}$ . Indeed, separation between the records is ensured by the  $\mathbf{I}_{(1)}^*$ s, since they are the only letters saying that their neighbour is an AP. Also it may pop in  $|X|$  steps the scope of a diagonalization  $\langle X \rangle$ , since the diagonalization depths in  $X$  are  $\leq c$  and the finite control of  $INT_{\mathbf{A}c}$  can be used to count them.

Let  $R_c[\mathbf{p}_1 \dots \mathbf{p}_k \mathbf{p}](n)$  denote the runtime of  $INT_c$  for  $\Sigma = \mathbf{p}_1 \dots \mathbf{p}_k \mathbf{p}$ , and  $n$  as above, and let  $R[\dots \mathbf{p}](n)$  be short for  $R_c[\mathbf{p}_1 \dots \mathbf{p}_k \mathbf{p}](n) - R_c[\mathbf{p}_1 \dots \mathbf{p}_k](n)$ . We have

$$R[\dots \mathbf{A I}](n) \leq n \tag{1}$$

$$R[\dots \mathbf{q I}](n) \leq nR[\dots \mathbf{q}](n) + |\mathbf{q}|n \tag{2}$$

$$R[\dots \mathbf{q} \langle X \rangle](n) \leq R[\dots \mathbf{q} X^n](n) + |X|n. \tag{3}$$

**Overall Complexity** We show by induction on  $\alpha_1 \# \dots \# \alpha_k \# \alpha$  that we have

$$R_c[\mathbf{p}_1 \dots \mathbf{p}_k \mathbf{p}](n) \leq \sum_{i \leq k} B_{\alpha_i}(n+1) + B_{\alpha}(n+1) \quad (\alpha_i = o \mathbf{p}_i; \alpha = o \mathbf{p}).$$

The result then follows for  $k = 0$ . Step. Case 1.  $\mathbf{p} = \mathbf{A I}$ . By I.H. and (1) since  $|\mathbf{p}| = 1$  and  $B_1(n) = n$ . Case 2.  $\mathbf{p} = \mathbf{q I}$ . We have  $\alpha = \beta + 1 \geq 2$  and

$$\begin{aligned} R[\dots \mathbf{q I}](n) &\leq nB_{\beta}(n+1) + |\mathbf{q}|n && \text{I.H. and (2)} \\ &\leq nB_{\beta}(n+1) + B_{\beta}(n) && \S 16(3) \\ &\leq B_{\beta+1}(n+1) && \S 16(1) \text{ with } \beta = 1. \end{aligned}$$

Case 3.  $\mathbf{p} = \mathbf{q}\langle X \rangle$  and  $\alpha = \beta\#\omega^{c_1+1}\#\omega^{c_k+1}$  ( $k = |X|$ ). We have

$$\begin{aligned}
R[\dots \mathbf{q}\langle X \rangle](n) &\leq B_{\beta\#\omega^{c_1}n\#\dots\#\omega^{c_k}n}(n+1) + |\mathbf{X}|n && \text{I.H. and (3)} \\
&\leq B_{\beta}(n+1)B_{\omega^{c_1}n}(n+1) \dots B_{\omega^{c_k}(n+1)}(n+1) && \text{16(1) and (3)} \\
&\leq B_{\beta}(n+1)B_{\omega^{c_1}(n+1)}(n+1) \dots B_{\omega^{c_k}(n+1)}(n+1) \\
&\leq B_{\alpha}(n+1) && \text{16(2) and (1)}.
\end{aligned}$$

Note that I.H. can be applied with  $\beta\#\omega^{c_1}n\#\dots\#\omega^{c_k}n$  as  $\alpha$  by Lemma 15.

## References

- [1] S.J.Bellantoni, *Predicative recursion and the polytime hierarchy*. In P. Clote and J. Remmel (eds), Feasible mathematics II. Birkhäuser, 1994.
- [2] S.J.Bellantoni and S.Cook, *A new recursion-theoretic characterization of the poly-time functions*. Computational Complexity 2(1992)97-110.
- [3] S. Caporaso, *Safe TMs, Grzegorzcyk classes and polytime*. Intern. J. Found. Comp. Sc. 7.3(1996)241-252.
- [4] S. Caporaso, M. Zito, N. Galesi, *A predicative and decidable characterization of the polynomial classes of languages*. Theoretical Comp. Sc. 250(2001)83-99.
- [5] S. Caporaso, G. Pani, E. Covino, *A predicative approach to the classification problem*. J. Funct. Programming 11.1(2001)95-116.
- [6] E. Covino and G. Pani *An Implicit Recursive Language for the Polynomial Time-space Complexity Classes*. J. Univ. Comp. Sc. 8.1(2002)75-84.
- [7] M. Fairtlough and S.S. Wainer, *Hierarchies of provably recursive functions*. In S. Buss (ed.) Handbook of Proof Theory. Elsevier, 1998.
- [8] D. Leivant, *A foundational delineation of computational feasibility*. Information and Computation 110(1994)391-420.
- [9] D. Leivant, *Ramified recurrence and computational complexity I: word recurrence and polytime*. In P. Clote and J. Remmel (eds), Feasible mathematics II. Birkhäuser, 1994.
- [10] D. Leivant and J.-Y. Marion, *Ramified recurrence and computational complexity II: substitution and polyspace*. In J. Tiuryn and L. Pacholski (eds), Computer Science Logic. Springer, LNCS 933(1994) 486-500.