Implicit Computational Complexity and the Exponential Time-Space Classes

Salvatore Caporaso
Università di Bari
Dipartimento di Informatica
Via Orabona, I-70125 Bari
Italy

Emanuele Covino
Università di Bari
Dipartimento di Informatica
Via Orabona, I-70125 Bari
Italy

Paolo Gissi
Università di Bari
Dipartimento di Informatica
Via Orabona, I-70125 Bari
Italy

Giovanni Pani
Università di Bari
Dipartimento di Informatica
Via Orabona, I-70125 Bari
Italy

Abstract: We extend the *Implicit Computational Complexity* program, promoted by Leivant and by other scholars, to all complexity classes DTIMESPACEF$(f(n), g(n))$, between DTIMEF(n) and DSPACEF(n^{n^c}). Let $clps(\alpha, n)$ denote the result of replacing ω by n in Cantor normal form for $\alpha < \omega^{\omega^\omega}$. A hierarchy $\mathcal{TS}_{\alpha\beta}$ is defined by means of a very restricted form of substitution, and of two un-limited operators (simultaneous predicative recurrence and *constructive diagonalization*), and it is proved that DTIMESPACEF$(n^{clps(\beta,n)}, n^{clps(\alpha,n)}) = \mathcal{TS}_{\alpha\beta}$. For example DTIMESPACEF$(n^2, n^n) = \mathcal{TS}_{\omega^\omega,2}$.

Key–Words: Implicit Computational Complexity, Exponential time, Exponential space, Time-Space Classes

1 Introduction

In terms of *Implicit Computational Complexity*, there are at least two ways to deal at the same time with time and space. One may add to the usual forms of safe PR on notations a stronger scheme which, though saving the distinction between harmless and harmful positions, repeats the recursion invariant for *all* values of the recursion variable, and not only for those coming from destruction (that is, $2^{O(n)}$ steps instead of $O(n)$). This approach is not fully satisfactory, since we wish to have *few* definition schemes, but, if adopted together with safe recursion only, doesn't allow to reach the higher classes. Add that modulating the power of such scheme in order to cope with the intermediate classes may be not trivial.

On the other hand if safe recursion alone is used, essentially *unary counters* are involved in proofs by simulation, which are too large for some space classes. Of course one might look for proofs discarding simulation in favour of a very insight to the complexity classes. This approach actually works with single classes; however, it appears to be beyond the present understanding of the complexity phenomenon when applied to the problem of a unified taxonomy.

These difficulties are solved in the present paper by integrating safe recursion with an use *from below* of diagonalization.

2 T-functions

Constants Define $\mathbf{B} := \{1, 2\}$. $X, \dots, Z$ are words in $\mathbf{B}^*$. The *dyadic (modified) numeral* $\overline{n}$ for $n = \Sigma_{0\leq i\leq m} b_i 2^i$ is $b_0 \dots b_m$ (thus $\overline{0}$ is the empty word). $\mathbf{T}^+$ is the class of all *ternary numerals* $p, \dots, s$, that is of all words over $\mathbf{T} := \{0, 1, 2\}$ which are 0 or do not begin by 0. Following a method in Schwichtenberg [2], we use the ternary numerals to represent n-ples of dyadic numbers, with the zero playing the role of *comma*.

Definition 1 Given a word s in the form $X_m 0 X_{m-1} 0 \dots X_2 0 X_1$, we call X_i the i-th *component* of s, we denote it by $(s)_i$, and we say that the *number of components* $cn(s)$ of s is m. If s is a word over $\mathbf{B}$, then s is its only component, and hence $cn(s) = 1$. If s is 0 then $\overline{0}$ is its only component.

Variables and functions $a, b, a_1, \dots$ are *digits* of the current alphabet. Unlike $X, s, X_1, s_1, \dots$ which form a potential infinity of informal variables, x, y, z are three fixed syntactic objects, called, respectively, the *auxiliary variable,* the *parameter,* and the *recursion variables*. They play a distinct and precise role in the construction of the **T**-functions (see Note 14 for the rationale of this convention). u, v, w are variables defined on the syntactic objects x, y, z and **u**, **w** are tuples of such variables.

When we write $f(\mathbf{u})$ we always assume that some of the indicated variables may be absent. Given $f(x, y, z)$, we denote by $f(s, t, r)$ the value of f when the values s, t, r are assigned to x, y, z.

Though in principle a **T**-function takes $n \leq 3$ ternary numerals into a numeral, in practice we understand it as taking n tuples of dyadic numerals into a tuple of dyadic numerals.

Notation 2 Let a class $\mathcal{C}$ of **T**-functions be given, together with a collection $\Sigma_1, \dots, \Sigma_n$ of definition schemes (i.e. of functionals taking tuples of **T**-functions into a new function). We write

$$(\mathcal{C}; \Sigma_1^*, \dots, \Sigma_l^*, \Sigma_{l+1}, \dots, \Sigma_n) \quad (0 \leq l \leq n)$$

for the class of all functions definable from $\mathcal{C}$ by closure under $\Sigma_1, \dots, \Sigma_l$ and by at most one application of each of the Σ_j's for $l + 1 \leq j \leq n$.

Definition 3 (1) The *basic functions* are the the *constructors* $c_i^a(x)$ and the *destructors* $d_i(x)$ $(a = 1, 2)$ which by input s:

(a) return 0 if $n \neq cn(s)$, or $(s)_i = \overline{0}$, or $s = 0$;

(b) respectively add digit a at the right of, and cancel the rightmost digit of $(s)_i$, in all other cases. (We don't formulate here the marginal clauses like $c_1^1(0) = 1$, needed to make this definition consistent with the part of Def. 1 which doesn't allow non-significant left zeroes.)

(2) The *simple (definition) schemes* smpl are:

$f = cmp(g, h)$ is the *composition* $g(h(u))$ of $h(u)$ with $g(u)$, provided that g is an initial function (i.e., it belongs to the class $\mathcal{T}_0$ below);

$f = case_j^a(g, h)$ is defined by *branching* in g and h if we have

$$f(s,t,r) \;=\; \textit{if } (s)_j = Xa \;\; \textit{then } g(s,t,r) \textit{ else } h(s,t,r)$$

$f = ext_u(g)$ $(u = x, z)$ is the result of the *explicit transformation* of u into y in g.

$f = agn(s, g)$ is the result of the *assignment* of constant s to x in g.

(3) A *modifier* is an element of the closure $\mathcal{M}$ of the basic functions under cmp.

(4) The starting class of our hierarchy is $\mathcal{T}_0 := (\mathcal{M}; cmp^*, case^*, agn^*)$.

Example 4 1. Sometimes, to improve readability, we write hg for $cmp(g, h)$. Define the *dummy* function by $du := c_1^1 d_1 = cmp(d_1, c_1^1)$. We have $du(s) = s$ unless $(s)_i = \overline{0}$.
2. Define the modifiers $wr[X]$ (one for each X) by $wr[b] := c_1^b; wr[b_{m+1} \dots b_1] := c_1^{b_{m+1}} wr[b_m \dots b_1]$.

Coding To describe our functions we use expressions which may be regarded as readable transcriptions of words in Polish suffix form over the alphabet

$$\mathbf{U} \;:=\; \{c^a, d, agn, ext_u, cmp, isbst, case^a, sr, sr_2, cdiag, \circ, *, nagn, \overline{0}, 0, 1, 2\},$$

where an *arity* is implicitly associated to each letter. $cd_{\mathbf{U}}$ will denote the cardinality of **U**. In particular, when coding a **B**-word X, arity 0 is associated with the left-most letter of X, and arity 1 to its other letters (if any).

Definition 5 The code $\ulcorner L \urcorner$ for the h-th letter of **U** is $1^h 2^{cd_{\mathbf{U}} - h}$.

Every subscript $n + 1$ occurring in the construction of a function is coded by $\circ *^n$.

For all $L \in \mathbf{U}$ of arity $m \geq 0$, and for all $E = E_1 \dots E_m L$ define the *code* for E by $\ulcorner E \urcorner := \ulcorner E_1 \urcorner \dots \ulcorner E_m \urcorner \ulcorner L \urcorner$; however, to save space (cf. Notat. 23 and Lemma 30: a code for the assignment $agn(\ulcorner X \urcorner, f)$ of $\ulcorner X \urcorner$ to x in f is $\ulcorner X \urcorner \ulcorner f \urcorner \ulcorner nagn \urcorner$ too.

Example 6 We have $X := \ulcorner c_3^1 \urcorner = \ulcorner \circ \urcorner \ulcorner * \urcorner \ulcorner * \urcorner \ulcorner c_3^1 \urcorner$. Define $f := agn(\ulcorner c_3^1 \urcorner, sr(c_1^3, c_1^3))$. f is coded by $Y := \ulcorner X \urcorner X X \ulcorner sr \urcorner \ulcorner agn \urcorner$. and by $X X X \ulcorner sr \urcorner \ulcorner nagn \urcorner$.

Note 7 Univocal parsing of the codes is not disturbed by this use of nagn, since the leftmost letter of $\ulcorner X \urcorner$ is associated in **U** with a letter of arity 0.

Notation 8 $\{X\}$ is the function coded by X. We often use the identities $\{\ulcorner f \urcorner\} = f$ and $\ulcorner \{X\} \urcorner = X$.

Definition 9 (1) The *rate of growth* $rg(f)$ of function $f \in \mathcal{T}_0$ is $m - n$ if f is a modifier built-up from $m \geq 0$ constructors and $n \geq 0$ destructors; it is $rg(g) + rg(h)$ if we have $f = cmp(g, h)$; it is $\max(rg(g_i))$ for $f = case(g_1, g_2)$; and it is $rg(g)$ for $f = agn(s, g_1)$.

(2) The *length* $|E|$ of word $E \in \mathbf{U}^*$ is the number of letters of **U** occurring in E.

Lemma 10 For all $f \in \mathcal{T}_0$ we have $|f(s)| \leq |s| + rg(f)$ (cut-off subtraction if $rg(f) < 0$).

Proof. Induction on $|f|$. Step. Assume for example that we have $f(s) = g(h(s))$, and that g is a modifier, since else the result is an obvious consequence of the ind. hyp. Define $t := h(s)$, and assume $t \neq 0$, since else $f(s) = 0$. By the ind. hyp. we have $|t| \leq |s| + rg(h)$. g consists of n destructors and m constructors, with $rg(f) = (m-n)+rg(h)$. Either each destructor of g actually erases a digit of t, or some of them return 0. In the former case we have $|f(s)| \leq |s| + rg(h) + m - n \leq |s| + rg(f)$. In the latter, all constructors of g return 0 too.

Definition 11 $f = sr(g, h)$ is defined by *safe-recursion* in $g(x, y)$ and $h(x, y, z)$ if we have

$$\begin{cases} f(s,t,a) &= g(s,t) \\ f(s,t,ra) &= h(f(s,t,r),t,ra). \end{cases}$$

Sometimes, given $h(x, y)$, we write $sr^*(h)$ for $ext_z(sr(h, h))$.

Notation 12 The *n-th (left) iterate* of function $F(E, \ldots)$ (not necessarily a **T**-function) is given by $F^1(E, \ldots) := F(E, \ldots)$; $F^{n+1}(E, \ldots) := F(F^n(E, \ldots), \ldots)$.

Example 13 1. Given $g(x, y)$, define $h := sr(g, g)$. We have $h(s, t, a) = g(s, t)$ and $h(s, t, ra) = g(h(s, t, r), t)$. Thus, by definition of ext_z and by a straightforward induction on $|r|$ we may conclude that we have $h(s, t) = g^{|t|}(s, t)$ for all $g(x, y), s, t$.
2. For every Y define (cf. Ex. 4 for $wr[X]$) $e[Y] := sr(wr[Y], wr[Y])$. We have $e[Y](X, r) = XY^{|r|}$ $(= XY \ldots Y)$ ($|r|$ times). Since $e[Y]$ is defined by sr in two modifiers, it belongs to the class $\mathcal{T}_1$ to be defined below.

Note 14 By ext and agn we may take $g(x, y, z)$ into functions like, say, $g(y, y, z)$, $g(x, y, y)$ or $g(10, y, z)$. However we cannot obtain $g(x, y, x)$. In general, the restriction of substitution to explicit transformations which, in turn, do not allow renaming x as z avoids affecting the recursion variable of a safe recursion with the previous value of the function being recursed upon. In this way a variable which is *safe* or *dormant* according to Simmons or Bellantoni & Cook keeps such.

Notation 15 Ordinals (1) $\alpha, \beta, \gamma, \delta, \lambda, \mu, \nu, \xi$ are ordinals below ω^{ω^ω}. In particular, λ, μ and ν are limits.

(2) Given an ordinal function $\theta(\alpha)$, define $\theta^\omega(\alpha) := \sup_{n<\omega}\{\theta^n(\alpha)\}$.

(3) TH is the smallest class of ordinal functions definable by closure of 0 and the identity under successor, sum and θ^ω for all $\theta \in TH$. θ, η, ζ will denote elements of TH. The *continuity* property $(\theta^\omega(\alpha))_n = \theta^n(\alpha)$ holds for all $\theta \in TH$.

(4) For all $\alpha > 0$, we write $\alpha =_{NF} \beta + \omega^\gamma$ when $\omega^\gamma \geq 1$ is the rightmost term of the *Cantor normal form* CNF(α) for α, and $\beta \geq 0$ is the sum of the other terms.

(5) $\alpha \# \beta$ is the natural, or commutative, or *component-wise* sum of α and β, such that, for example, $1\#\omega = \omega + 1$, and $\omega\#(\omega^\omega + 1) = \omega^\omega + \omega + 1$.

(6) The *collapse* of α at n is the function $clps(\alpha, n)$ obtained by replacing all occurrences of ω in CNF(α) by n. For example $clps(\omega^{\omega^3} + \omega + \omega + \omega^0 + \omega^0 + \omega^0, n)$ is $n^{n^3} + 2n + 3$.

(7) The *standard* assignment of fundamental sequence λ_m to the limit $\lambda =_{NF} \alpha + \omega^\gamma$ is $\alpha + \omega^\delta m$ if $\gamma = \delta + 1$, and is $\alpha + \omega^{\mu_m}$ if $\gamma = \omega^\mu$.

We shall deal with couples (α, β) of ordinals, whose left-side element $(\alpha, \beta)_\tau := \alpha$ refers to time and whose right one $(\alpha, \beta)_\sigma := \beta$ refers to space. We now fix some conventions allowing to deal with the elements of these couples, using ρ as a meta-variable defined on the meta-constants τ, σ.

Notation 16 1. $(\alpha, \beta)\#_\tau\gamma = (\alpha\#\gamma, \beta)$, and $(\alpha, \beta)\#_\sigma\gamma = (\alpha, \beta\#\gamma)$.
2. $(\gamma, \delta) \prec_\tau (\alpha, \beta) := \gamma < \alpha \wedge \delta \leq \beta$; $(\gamma, \delta) \prec_\sigma (\alpha, \beta) := \gamma \leq \alpha \wedge \delta < \beta$.
3. Lim_ρ is the class of all couples (α, β) such that $(\alpha, \beta)_\rho$ is a limit.
4. If $(\alpha, \beta) \in Lim_\rho$ then $(\alpha, \beta)_{\rho,n}$ is the result of replacing $\lambda = (\alpha, \beta)_\rho$ with λ_n in (α, β).
For example $(\omega, \omega)_{\tau,1} = (1, \omega)$.

Definition 17 Assume defined the elements of a hierarchy $\mathcal{C}_{(\alpha,\beta)}$ for all $(\alpha, \beta) \prec_\rho (\gamma, \delta) \in Lim_\rho$. $f = cdiag_\rho(e)$ is defined by *ρ-(constructive) diagonalization* at (γ, δ) in the *enumerator* $e \in \mathcal{T}_1$ if for all s, t, r we have

$$f(s,t,r) = \{e(r)\}(s,t,r) \quad \text{and} \quad \{e(r)\} \in \mathcal{C}_{(\gamma,\delta)_{\rho,|r|}}.$$

Definition 18 (1) $f = isbst(g, h)$ is the result of the *inessential substitution* of $h(x, y, z)$ for x in $g(x, y, z)$.

The *degree* $dg(f)$ of $f \in \mathcal{T}_0$ is 0 if $rog(f) \leq 0$, and is 1 otherwise.

For all f defined by smpl,isbst,sr in g_1, g_2, define $dg(f) := \max(dg(g_1, g_2))$.)

For all $f = cdiag(e)$ define $dg(f) := \sup(dg(\{e(r)\})$.

(2) Function $f = nisr(g, h)$ is defined by *not-increasing safe recursion* in g and h if we have $f = sr(g, h)$ and $dg(h) = 0$.

2.1 The hierarchy and the result

Definition 19 For all $\beta < \omega^{\omega^\omega}$, and for all $\alpha < \omega^\omega$ define the *time-space hierarchy* $\mathcal{TS}_{(\alpha,\beta)}$ by

$$\mathcal{TS}_{(\alpha,\beta)} := \mathcal{TS}_{(\alpha,\alpha)} \text{ if } \alpha < \omega;$$

$$\mathcal{TS}_{(0,0)} := \mathcal{T}_0$$

and in all other cases:

$$\begin{array}{rcl} \mathcal{TS}_{(\alpha,\beta)\#_\rho 1} & := & (\mathcal{TS}_{(\alpha,\beta)}; smpl^*, cmp^*, isbst^*, sr) \\ & & (\ sr{=}nisr \text{ when } \rho = \sigma) \\ \mathcal{T}_{(\alpha,\beta)} & := & (\bigcup_{(\gamma,\delta)\prec_\rho(\alpha,\beta)} \mathcal{T}_{(\gamma,\delta)}; \\ & & smpl^*, cmp^*, cdiag_\rho, isbst^*) \\ & & (\alpha,\beta) \in Lim_\rho. \end{array}$$

Definition 20 $B_0(n) := 1$;
$B_\alpha(n) := \max(2, n^{clps(\alpha,n)})$ $(0 < \alpha)$.

Under an appropriate notion of equivalence, we have the following result.

Theorem 21 DTIMESPACEF$(B_\beta(n), B_\alpha(n)) \subseteq 3$ $\mathcal{TS}_{(\alpha,\beta)} \subseteq$ DTIMESPACEF$(B_\beta(n{+}1), B_\alpha(n{+}1))$.

Proof. By the simulations of next sections (see Lemmas 35 and 33).

3 Simulation of TM's

Without any loss of generality, we may restrict ourselves to tm's M^{kQ} with $Q{+}1$ states and k semi-tapes over the tape alphabet **B**. 0 (1) is its initial (final) state.

In one *step* M^{kQ} executes an *instruction* of the form (i, X, i^*, j, h) where

$$(i, i^* \leq Q; j \leq k; X = b_1 \dots b_k; h = -1, 0, b)$$

to be understood as: if the current state is i and if the observed symbols of (tapes) $1, \dots, k$ are $b_1, \dots, b_k$ then enter stater i^* and *move left, right, write* b on tape j^*.

An *instantaneous description* (id) of M^{kQ} is a word

$$1U_1, \dots, 1U_Q, Y_1, \dots, Y_k, 1O_1, \dots, 1O_k, 1Z_1, \dots, 1Z_k$$

such that:

U_i is 2 if the current state is $i+1$ and is 1 otherwise;

Y_j is the part of tape j at the left of the observed symbol O_j (excluded);

Z_j is the contents of j at the right of O_j, read in reverse order.

Lemma 22 (1) For all M^{kQ} a function nxt_M can be defined in $\mathcal{T}_0$ which takes the id's of M into the next ones.

(2) We may replace nxt_M by a new function 0-nxt_M such that *rog*(0-nxt_M) = 0, and which works under the assumption that M in its next step, will not enter a right-side part of tape represented by the empty word.

Proof. 1. Define the functions $ex[I](s)$ which execute instructions $I = (i, b_1 \dots b_k, i^*, h)$ of a given $M^{(kQ)}$ by composition of two destructors and two constructors taking i into i^* with: (a) one destructor and one constructor, if $h = a$; or (b) one case, two destructors and two constructors if $h = -1$; or (c) if $h = 0$ then (cf. Ex. 4 for the dummy function du)

$$\begin{array}{l} d_{2k-j+1} c^{b_j}_{3k-j+1} case^1_{k-j+1} \\ (c^1_{2k-j+1}, c^2_{2k-j+1}) d_{k-j+1} \\ case^0_{k-j+1}(c^1_{k-j+1}, du). \end{array}$$

The code above may be *translated* into

> **begin** erase O_j and append it to Y_j;
> *if* the first symbol b of Z_j is 1 *then* $O_j := 1$
> *else* $O_j := 2$; erase b;
> *if* $Z_j = \overline{0}$ *then* $Z_j := 1$ *else* $Z_j := Z_j$ **end**.

We see from the translation that the purpose of last line is to *grant new space* when M visits for the first time a cell on the right-side of tape j.

We can now build-up function nxt_M from functions $ex[I]$ by means of a sequence of at most $Q + k$ case's allowing to know i and $b_1 \dots b_k k$, and to select the appropriate $ex[I]$ accordingly.

2. The rate of growth of nxt_M is +1, since the rate of growths of part (a),(b), (c) and of lines 1 and 2 of the translation is 0. Thus the overall rate of growth of nxt_M raises by one only when we grant new space to M. Thus function 0-nxt_M can be obtained by just dropping the part $case^0_{k-j+1}(c^1_{k-j+1}, c^1_{k-j+1} d_{k-j+1})$ from part (d) of the definition of nxt_M. Note that a consequence of this change is that, if M moves right on a tape whose right-side representation is empty, the corresponding destructor d_{k-j+1} causes 0-nxt_M to return 0. Obviously, further applications of 0-nxt_M would not work.

Notation 23 In view of further work with the codes we adopt the following abbreviations. For all Y and U define (cf. Ex. 13 for $e[Y]$)

$$\begin{array}{rcl} R & := & \ulcorner sr_2 \urcorner; \\ \tilde{Y} & := & \ulcorner e[Y] \urcorner \ \ulcorner nagn \urcorner \ \ulcorner cdiag \urcorner \ \ulcorner ren_z \urcorner; \\ Y^*_{-1} & := & Y\underbrace{Y}; \\ Y^*_n & := & \widetilde{Y^*_{n-1}} \quad (n \geq 0). \end{array}$$

Definition 24 1. An *α-ρ-iterator* for $h(x,y)$ is a function $[h;\alpha]_\rho(x,y) \in \mathcal{TS}_{(\gamma,\delta)}$, with $(\gamma,\delta)_\rho = \alpha$, such that for all s,t we have

$$\begin{array}{ll}(a) & [h;\alpha]_\rho(s,t) = h^{B_\alpha(|t|)}(s,t);\\ (b) & dg(h)=0, \text{ when } \rho=\tau.\end{array} \quad (1)$$

2. X is an *α-open code* if for all $[h;\beta]_\rho \in \mathcal{TS}_{(\gamma,\delta)}$ we have $\{\ulcorner[h;\beta]_\rho\urcorner X\} = [h;\alpha\#\beta]_\rho \in \mathcal{TS}_{(\gamma,\delta)\#_\rho\alpha}$.

Note 25 By inspection to next definitions and proofs one sees that the hypothesis (b) of equation (1) allows ignoring in definition $R := \ulcorner sr_2 \urcorner$ the distinction between sr and nisr.

Notation 26 $\mathcal{C}_\alpha$ denotes any class $\mathcal{TS}_{\gamma\delta}$ such that $(\gamma,\delta)_\rho = \alpha$. If α is $(\gamma,\delta)_\rho$ then $\mathcal{C}_{\alpha\#\eta}$ is $\mathcal{TS}_{(\gamma,\delta)\#_\rho\eta}$.

Note 27 1. We have $[[g;\alpha];\beta](s,t) = ((g^{B_\alpha(|t|)})^{B_\beta(|t|)})(s) = g^{B_\alpha(|t|)B_\beta(|t|)}(s) = [g;\alpha\#\beta]$.
2. The concatenation XY of an α-open code X with a β-open code Y is an $\alpha\#\beta$-open code. Indeed, by part 1, for all γ-ρ-iterator $[h;\gamma] \in \mathcal{C}_\gamma$, we have $\{\ulcorner h\urcorner XY\} = [h;\gamma\#\alpha\#\beta] \in \mathcal{C}_{\gamma\#\alpha\#\beta}$.

We now associate each α with a **B**-word $\langle\alpha\rangle$; we then show that each $\langle\alpha\rangle$ is an α-open code.

Definition 28 For $\alpha=0$ and for all $\alpha =_{NF} \beta+\omega^\gamma < \omega_3$ define inductively $\langle\alpha\rangle$ by:

$$\begin{array}{llll}1 & \langle 0\rangle & := & \overline{0}\\ 2 & \langle\beta+1\rangle & := & \langle\beta\rangle R\\ 3 & \langle\beta+\omega^\gamma\rangle & := & \langle\beta\rangle\langle\omega^\gamma\rangle \quad (0<\beta,\gamma)\\ 4 & \langle\omega^{\delta+\omega^n}\rangle & := & \langle\omega^\delta\rangle^*_n\\ & & & (\delta\ge\omega^n \text{ or } \delta=0;\ n\ge 0).\end{array}$$

Example 29 1. $\langle 0\rangle$ is a 0-open code. Indeed we have $\{\ulcorner h\urcorner\langle 0\rangle\} = h = [h;0]$ (since $B_0(n)=1$).
2. $\langle 1\rangle$ is a 1-open code. Indeed the Ex. 13 gives $\{\ulcorner h\urcorner\langle 1\rangle\} = [h;1] \in \mathcal{C}_{\alpha+1}$ for all $h(x,y)\in\mathcal{C}_\alpha$.
3. Note that line 4 says that $\langle R\rangle^*_n$ is an ω^{ω^n}-open code.

Lemma 30 If for a given θ and for all α-open codes X we have that XY is a $\theta(\alpha)$-open code, then $\tilde{Y}$ is a $\theta^\omega(\alpha)$-open code.

Proof. Notation. Assume given an α-open code X; and a β-iterator $\{U\} = [h;\beta] \in \mathcal{C}_\beta$ (cfr. notation 26). By Def. 24 we have $\{UX\} = [h;\beta\#\alpha] \in \mathcal{C}_{\beta\#\alpha}$. Define $W := \ulcorner e[Y]\urcorner \ulcorner cdiag\urcorner \ulcorner agn\urcorner$. We have to show that $\{UXWren_z\} = [h;\beta\#\theta^\omega(\alpha)] \in \mathcal{C}_{\beta\#\theta^\omega(\alpha)}$.

Indeed since $\{UXW\}(s,t,r) = cdiag(agn(UX,e[Y]))(s,t,r) = \{e[Y](UX,r)\}(s,t,r)$, we have

$$\begin{array}{rl}\{UXW\}(s,t,r) & = \{UXY^{|r|}\}(s,t,r)\\ & \quad \text{by Ex. 13}\\ & = \{\ulcorner[h;\beta\#\theta(\alpha)]\urcorner Y^{|r|-1}\}(s,t,r)\\ & \quad \text{hyp. on } Y \text{ and } \theta\\ & = \{\ulcorner[h;\beta\#\theta^{|r|}(\alpha)]\urcorner\}(s,t,r)\\ & \quad \text{as above for } |r|-1 \text{ times}\\ & = [h;\beta\#\theta^{|r|}(\alpha)](s,t)\\ & \quad z \text{ is absent in all } [h;\beta]\\ & = h^{B_{\beta\#\theta^{|r|}(\alpha)}(|t|)}(s,t)\\ & \quad \text{definition of } h\text{-}\alpha\text{-iterator}\\ \{UX\tilde{Y}\}(s,t) & = \{UXW\}(s,t,t)\\ & \quad \text{by definition of } W \text{ and } \tilde{Y}\\ & = [h;\beta\#\theta^{|t|}(\alpha)](s,t) \in \mathcal{C}_{\beta\#\theta^{|t|}(\alpha)}\\ & \quad \text{hyp. on } \theta \text{ and } Y\\ & = h^{B_{\beta\#\theta^\omega(\alpha)}(|t|)}(s)\\ & \quad \text{continuity of } \theta \text{ and } \#\\ & = [h;\beta\#\theta^\omega(\alpha)](s,t) \in \mathcal{C}_{\beta\#\theta^\omega(\alpha)}\\ & \quad \text{since defined by cdiag and ren}_z\\ & \quad \text{in functions in } \mathcal{C}_{(\beta\#\theta^\omega(\alpha))_{|r|}}.\end{array}$$

Lemma 31 1. Define $\eta_{-1}(\alpha) := \alpha\#\alpha$; $\eta_{n+1}(\alpha) := \eta_n^\omega(\alpha)$. We have $\eta_n(\alpha) = \alpha\omega^{\omega^n}$ for all $n\ge 0$.
2. If Y is an α-open code, then Y^*_n is an $\eta_n(\alpha)$-open code.

Proof. 1. Induction on n. Basis. We have $\eta^m_{-1}(\alpha) = \alpha m$. Hence $\eta_0(\alpha) = \sup\{\alpha m\} = \alpha\omega$.
Step. We show by induction on m that we have $\eta^m_n(\alpha) = \omega^{\omega^n m}$. Indeed, $\eta^{m+1}_n(\alpha) = \eta_n(\eta^m_n(\alpha)) =$ (ind. hyp. on n) $\eta^m_n(\alpha)\omega^{\omega^n}$ = (ind.hyp. on m) $\alpha\omega^{\omega^n m}\omega^{\omega^n}$.
2. We obtain the basis by applying last lemma with η_{-1} as θ and (cf. Note 27) with Y^*_{-1} as Y. Step. $n = m+1$. Again by last lemma, with Y^*_m as Y and η_m as θ.

Lemma 32 Every $\langle\alpha\rangle$ is an α-open code. Hence, for all $h\in\mathcal{T}_0$ we can define in $\mathcal{T}_\alpha$ function $[h;\alpha]_\sigma$; and, for all $h\in\mathcal{T}_0$ such that $dg(h)=0$, we can define in $\mathcal{T}_\alpha$ function $[h;\alpha]_\tau$.

Proof. Induction on α and cases like in Def. 28. Case 1 (basis of the induction) and case 2. Already proved in Ex. 29. Case 3. By Note 27 and the ind.hyp. Case 4. By last lemma, with $\omega^\delta\ge 1$ as α.

Lemma 33 $\mathrm{DTIMESPACEF}(B_\beta(n), B_\alpha(n)) \subseteq \mathcal{TS}_{(\alpha,\beta)}$.

Proof. Notations like in sect. 3. Let function $f(y)$ be computed according to an appropriate standard by a k-tapes tm M in time $cB_\beta(n)$ and in space $cB_\alpha(n)$ for a constant c. Let M_1 be a tm which by input t moves right on all tapes for $B_\alpha(n)$ steps and comes back to the input. Clearly f is also computed by the composition of M_1 with M.

For all functions nxt_M of Lemma 22, let nxt_M^c be the function (defined by c cmp's) which simulates c steps by M. By last lemma we may define:
(a) a function $g(x,y) := [nxt_{M_1};\alpha]_\sigma(x,y) \in \mathcal{TS}_{(\alpha,\alpha)}$ which, by input s,t, returns the id of M_1 by input s after $B_\alpha(|t|)$ steps.
(b) a function $h(x,y) := [0\text{-}nxt_M;\beta]_\tau(x,y) \in \mathcal{TS}_{(1,\beta)}$, which, by input s,t, returns the id of M by input s after $B_\beta(|t|)$ steps, provided that M doesn't require new space.

Define $e(x,y) := [h(g(x,y),y)]$. $f(y)$ is computed by $e(y,y)$. We have $e(y,y) \in \mathcal{TS}_{(\alpha,\beta)}$ because this function is defined by ren_x in a function which, in turn, is defined by isbst of $g \in \mathcal{TS}_{(\alpha,\alpha)} \subseteq \mathcal{TS}_{(\alpha,\beta)}$ for x in $h \in \mathcal{TS}_{(1,\beta)} \subseteq \mathcal{TS}_{(\alpha,\beta)}$.

4 Simulation by TM's

We first estimate the size of the functions in our hierarchy.

Lemma 34 $f(x,y,z) \in \mathcal{TS}_{(\alpha,\beta)}$ implies $|f(s,t,r)| \leq |s| + |\ulcorner f \urcorner| B_\beta(|t|+|r|+1)dg(f)$, for all s,t,r.

Proof. Define $m := |s|; n := |t|+|r|; c := |\ulcorner f \urcorner| dg(f)$. Induction on α,β and on the construction of f. Basis. Immediate. Step of the three inductions.
Case 1. f begins by a simple scheme or isbst. Immediately by the ind. hyp. on f.
Case 2. $f = sr(g_1,g_2)$ and $\beta = \gamma+1$. Define $c_i := dg(g_i)|\ulcorner g_i \urcorner|$. We show that we have $|f(s,t,r)| \leq m + cB_\gamma(n+1)|r| \quad (\leq m + cB_\beta(n+1))$. Induction on $|r|$:

$$\begin{aligned} |f(s,t,0)| &= |g_1(s,t)| \leq m + c_1B_\gamma(n+1) \\ &\qquad \text{by the ind. hyp. on } \beta \\ |f(s,t,ra)| &= |g_2(f(s,t,r),t,ra)| \\ &\leq |f(s,t,r)| + c_2B_\gamma(n+2) \\ &\qquad \text{by the ind. hyp. on } \beta \\ &\leq m + cB_\gamma(n+1)|r| + c_2B_\gamma(n+2) \\ &\qquad \text{by the ind.hyp. on } |r| \\ &\leq m + cB_\gamma(n+2)(|r|+1) \\ &\leq m + cB_\beta(n+2) \\ &\qquad \text{since } B_\gamma(l)l = B_{\gamma+1}(l). \end{aligned}$$

Case 3. $f = nisr(g_1,g_2)$. Define c_i as under case 2. α or β is a successor, and we have $c_2 = 0$. Assume for example that: $c_1 > 0$ and hence $c > 0$ too; and that $\beta = \gamma+1$, with $g_1,g_2 \in \mathcal{TS}_{(\alpha,\gamma)}$. We show that $|f(s,t,r)| \leq |g_1(s,t)|$. The result then follows by the ind. hyp. on β, since $c_1 < c$. Induction on $|r|$. Basis. Immediate. Step. We have

$$\begin{aligned} |f(s,t,ra)| &= |g_2(f(s,t,r),t,ra)| \leq |f(s,t,r)| \\ &\qquad \text{ind. hyp. on } \beta, \text{ since } dg(g_2)=0 \\ &\leq |g_1(s,t)| \\ &\qquad \text{ind. hyp. on } |r|. \end{aligned}$$

Case 4. $f = cdiag_\rho(e)$. Assume for example $\rho = \tau$. We have $\alpha = \lambda$. We have $|f(s,t,r)| =$

$$\begin{aligned} &|\{e(r)\}(s,t,r)| \leq m + |\ulcorner\{e(r)\}\urcorner| B_{\lambda_{|r|}}(n+1) \\ &\qquad \text{ind. hyp. on } \lambda_{|r|} \\ &\leq m + |\ulcorner f \urcorner| B_1(|r|+1) B_{\lambda_{|r|}}(n+1) \\ &\qquad \text{ind. hyp. for } \beta = 1 \text{ and } |\ulcorner e \urcorner| < |\ulcorner f \urcorner|. \end{aligned}$$

The result follows since we have $B_1(n+1)B_{\lambda_n}(n+1) \leq B_{\lambda_{n+1}}(n+1) = B_\lambda(n+1)$, as one sees by considering that in the worst case ($\lambda = \omega$), this is tantamount to $(n+1)(n+1)^n \leq (n+1)^{n+1}$.

Lemma 35 $\mathcal{TS}_{(\alpha,\beta)} \subseteq \mathrm{DTIMESPACEF}(B_\beta(n+1), B_\alpha(n+1))$.

The proof of this lemma is based on an analysis of the time and space complexity of two interpreters. We don't report it here for brevity (they will be included in the final version of this work). The reader might believe now our assertion by considering that: (a) Simulation of $f = sr(g,h)$ for $g,h \in dtimef(f(n))$ requires time $f(n)n$ (n repetitions of h on an argument in a *safe* position); standard of computation by means of stacks may be adopted which allow the computation to be carried-out on site, in the safe position, thus avoiding the waste of time to move the value of f back and forth between the safe area and the area reserved to the arguments of h. (b) By Lemma 34, simulation of $f = nisr(g,h)$ respects the promised space bounds. (c) Simulation of $f = cdiag(e)$ requires neglectable resources for e since this function is in lintimef; and resources for the computation of $\{e(r)\}$ may be checked by a straightforward transfinite induction on α and β.

References:

[1] D. Leivant, Ramified recurrence and computational complexity I: word recurrence and polytime. In P. Clote and J. Remmel (eds), Feasible mathematics II. (Birkhäuser, 1994).320-343.

[2] H. Schwichtenberg, *Eine Klassification der ϵ_0-rekursiven Funktionen,* Zeitschr. math. Logik u. Grundl. d. Math. 17(1971)61-74.