REVISTA MATEMÁTICA de la
Universidad Complutense de Madrid
Volumen 10, número 1: 1997

New recursive characterizations of the elementary functions and the functions computable in polynomial space.

I. OITAVEM

Abstract

We formulate recursive characterizations of the class of elementary functions and the class of functions computable in polynomial space that do not require any explicit bounded scheme. More specifically, we use functions where the input variables can occur in different kinds of positions - normal and safe - in the vein of the Bellantoni and Cook's characterization of the polytime functions.

1 Introduction

This paper is concerned with two well-known sub-recursive classes. Firstly, the class $\mathcal{E}$ of elementary functions, introduced by Kalmar (1943) [9] and Csillag (1947) [6], and which can also be described as the class of the functions computable in iterated exponential time. Secondly, the class Pspace of functions computable in polynomial space: see, for instance, [1] for a more detailed characterization.

The usual inductive formulations of $\mathcal{E}$ and Pspace use explicit bounded schemes; here our proposal is to establish characterizations without any explicit bounded scheme. In order to accomplish this we use the techniques that Bellantoni employed to get a similar result for the class

---

1991 Mathematics Subject Classification: 03D15, 03D20, 68Q15.
ACM Classification: F.1.3.
Servicio Publicaciones Univ. Complutense. Madrid, 1997.

of functions computable in linear time and for the class of functions computable in polynomial time, [3] or [4] (recently and independently, Bellantoni announced in [4] a similar characterization for the elementary functions).

In the next section we are going to work in numeric notation, i.e., we will be talking about functions defined in cartesian products of ω and assuming values in ω. However, when studying the class Pspace, it is convenient to abandon the numeric notation and adopt, instead, the binary notation. Therefore, we will work with functions defined in cartesian product of $\{0,1\}^*$ and assuming values in $\{0,1\}^*$, where $\{0,1\}^*$ is the set of 0-1 words. This change of notation is not mandatory. In fact, we could rewrite all this work in numerical notation, although - in our opinion - the binary notation is more adequate to express simultaneously the recursion on x and the recursion on the length of x. The reasons for which we made this change of notations will become more clear along the work.

2 Characterizations of $\mathcal{E}$

2.1 Classic characterization

We use a characterization of the elementary functions which can be easily deduced from the characterization given in [10]. The class $\mathcal{E}$ of the elementary functions is the smallest class of functions containing the Projection functions, the Zero and Successor functions and which is closed under ordinary composition $f(\bar{x}) = h(\bar{g}(\bar{x}))$[1] and the following scheme:

Bounded primitive recursion

$f(0,\bar{x}) = g(\bar{x})$

$f(y',\bar{x}) = h(y,\bar{x},f(y,\bar{x}))_{|t(y,\bar{x})}$

where t is a bounding function and $u_{|v} = \min\{u,v\}$ is the truncation function. Throughout this section, we mean by a bounding function a function in $\mathcal{T}$, where $\mathcal{T}$ is the smallest class of functions, closed under composition, that includes the projection functions and the sum, product and exponential functions.

[1] We adopt the standard notation: $\bar{x}$ for a $n-$tuple of variables and $\bar{g}$ for a $k-$tuple of $n-$ary functions.

It can be proved, by induction on the complexity of f, that if $f \in \mathcal{E}$ then there exists $b_f \in \mathcal{T}$ such that $\forall \bar{x} f(\bar{x}) \leq b_f(\bar{x})$, i.e., the "size" of the $\mathcal{E}$ functions is dominated by the "size" of functions in $\mathcal{T}$. To obtain this conclusion it is essential the presence of the truncate at the $\mathcal{T}$ functions in the bounded primitive recursion scheme (otherwise, we would obtain all the primitive recursive functions). Observe that the bounding functions are monotone. We will use this fact several times.

2.2 A new characterization

Following ideas of Bellantoni and Cook, [2] or [3], the functions in E will have a "normal" input and a "safe" input. We separate the two kinds of inputs by a semicolon, putting the normal ones in the left and the safe ones in the right.

The class E is the smallest class of functions containing the following initial functions $1' - 6'$ which is closed under the schemes of safe composition and safe recursion:

$1'$) $E(;x) = 0$ (Zero)

$2'$) $P_j^{n;m}(x_1, \cdots, x_n; x_{n+1}, \cdots, x_{n+m}) = x_j, 1 \leq j \leq n+m$ (Projections)

$3'$) $S(;x) = x'$ (Successor)

$4'$) $D(;x) = 2x$ (Duplication)

$5'$) $P(;0) = 0\, P(;x') = x$ (Predecessor)

$6'$) $Q(;x,y,z) = \begin{cases} x \text{ if } z = 0 \\ y \text{ otherwise} \end{cases}$ (Conditional)

Safe composition: $f(\bar{x};\bar{y}) = h(\bar{r}(\bar{x};);\bar{s}(\bar{x};\bar{y}))$

Safe recursion: $f(0,\bar{x};\bar{y}) = g(\bar{x};\bar{y})$

$$f(z',\bar{x};\bar{y}) = h(z,\bar{x};\bar{y},f(z,\bar{x};\bar{y}))$$

In the safe composition scheme the absence of some the functions $\bar{r}, \bar{s}$ is allowed.

All initial functions can be contructed into $\mathcal{E}$, if we ignore the different kinds of variables. The asymmetry of the safe composition scheme allows us "to change" variables from safe positions to normal positions, but not the opposite. This means that if $f(\bar{x}, w; z, \bar{a}) \in$ E then $f(\bar{x}, w, z; \bar{a}) \in$ E but, in general, we cannot say that if $f(\bar{x}, w; z, \bar{a}) \in$ E then $f(\bar{x}; w, z, \bar{a}) \in$ E. Also remark that in the safe recursion scheme, the recursion is done on a variable in a normal position and the recursive value $f(z, \bar{x}; \bar{y})$ is substituted into a safe position of h. Finally, note that only the safe recursion scheme enables us to introductive into E

functions that grow "substantially" faster than the functions involved in their definitions. For example, we can construct the sum based on successor -$sum(0;x) = x$, $sum(y';x) = S(;sum(y;x))$ - or the product based on sum - $prod(0,x;) = 0$, $prod(y',x;) = sum(x;prod(y,x;))$ - but we cannot define something growing "substantially" faster than $prod$ based on it. More generally, functions without safe inputs do not produce great increasings because, as we have already pointed out, in the safe recursion scheme the recursive value is placed into a safe position; hence, the strength of the safe recursion scheme is lost if h is a function without variables in safe positions. If we return to the aforementioned examples, it will become clear that the special constraints of safe recursion prevent us from defining $sum(;x,y)$, and without it e can not define $prod(y;x)$. Therefore, by safe recursion, we are only able to define $prod(y,x;)$ and the asymmetry of safe composition does not allow us to change any variable from a normal position to a safe one. The basic idea is that each time we use safe recursion in an essential way we increase the complexity but we lose a safe position forever. When no safe position is available we can no longer "increase the growth" of the class. Therefore, to evaluate how far we can go into this class we just have to pick the most powerful initial function and apply repeatedly safe recursion. In this case we obtain $f(x;) = 2^x$ by safe recursion based on the function D, which is the "strongest" initial function of E. Since f does not have safe inputs, we are already at the top, i.e., no function in E grows "substantially" faster than 2^x. At this point it is clear that if $t_f \in \mathcal{T}$ then there exists $r \in$ E such that $\forall \bar{x}\, r(\bar{x}) \geq t_f(\bar{x})$.

Next, our purpose is to show that the class E above coincides with $\mathcal{E}$, in the sense that the elementary functions are exactly those functions of E which only have normal inputs.

2.2.1 E "contains" the class $\mathcal{E}$

In order to prove that E contains $\mathcal{E}$ we need the following lemma:

Lemma 2.1. *If $f \in \mathcal{E}$ then*

$$\exists F \in E \exists t_f \in \mathcal{T}\ s.t.\ f(\bar{x}) = F(w;\bar{x})\, \forall \bar{x} \forall w : w \geq t_f(\bar{x}) \qquad (*)$$

Proof. The result follows by induction on the complexity of f. For the initial functions the result is straightforward.

Assume that f is defined by the composition scheme, i.e. $f(\bar{x}) = h(\bar{g}(\bar{x}))$ where, by induction hypothesis, $h, \bar{g}$ satisfy $(*)$. Since $\bar{g}$ belongs to $\mathcal{E}$ there exists $\bar{b}_g$ in $\mathcal{T}$ such that $\forall \bar{x}\, \bar{g}(\bar{x}) \leq \bar{b}_g(\bar{x})$. Hence $F(w;\bar{x}) = H(w, \bar{G}(w;\bar{x}))$ and $t_f(\bar{x}) = t_h(\bar{b}_g(\bar{x})) + \Sigma t_{g_i}(\bar{x})$ satisfy $(*)$ trivially.

The most difficult case happens when f is defined by bounded primitive recursion. In this case, we are given G, H in E and t_g, t_h in $\mathcal{T}$ satisfying $(*)$ by induction hypothesis. Since $f \in \mathcal{E}$, there is b_f in $\mathcal{T}$ such that $\forall y, \bar{x} f(y, \bar{x}) \leq b_f(y, \bar{x})$. The natural course of action would be to define F by recursion on y, but that is not possible because y is not in a normal position. Hence, we introduce a "new variable" z and use it to simulate recursion on y. Since f is defined by recursion on y, in order to compute $f(y, \bar{x})$ we must calculate all the values $f(0, \bar{x}), f(1, \bar{x}), f(2, \bar{s})$, until $f(y, \bar{x})$: it is thus expected that the simulation we are seeking reproduces this process.

Preliminarily, we define some useful functions. We start with the "modified difference", which can be defined into E by $T_1(0;x) = x, T_1(y';x) = P(;T_1(y;x))$ or by $T_2(y,x;) = T_1(y;x)$. If our notation allowed it, we would certainly write $x \dot{-} y$ instead of $T_1(y;x)$ and $T_2(y,x;)$, but as we have to distinguish the positions occupied by the input variables we must leave the standard notation. We also define the auxiliary function $Y(z,w;y) = T_1(T_2(z,w;);y)$, which gives us the $(w \dot{-} z)-$th predecessor of y or, more informally, $Y(z,w;y) = y \dot{-} (w \dot{-} z)$. We should note that, for each $w, y (w > y)$, when z increases from $w - y$ to w, $Y(z,w;y)$ increases from 0 to y and, hence, there are functions $\widehat{f}, t_f$ such that $\widehat{f}(z,w;y,\bar{x}) = f(Y(z,w;y),\bar{x})$, provided $w - y \leq z \leq w$ and $w \geq t_f(y,\bar{x})$. Thus, defining $F(w;y,\bar{x}) = \widehat{f}(w,w;y,\bar{x})$, we have $F(w;y,\bar{x}) = f(y,\bar{x})$, since $Y(w,w;y) = y$. This finishes the argument.

The function, $\widehat{f}$ and t_f, can be defined as follows:

$$\widehat{f}(0,w;y,\bar{x}) = 0$$

$$\widehat{f}(z',w;y,\bar{x}) = \begin{cases} G(w;\bar{x}) & \text{if } Y(S(;z),w;y) = 0 \\ H(w;Y(z,w;y),\bar{x},\widehat{f}(z,w;y,\bar{x})) & \text{otherwise} \end{cases}$$

$$t_f(y,\bar{x}) = t_h(y,\bar{x},b_f(y,\bar{x})) + t_g(\bar{x}) + y + 1,$$

which are, by construction, in E and $\mathcal{T}$, respectively. Once fixed y and $\bar{x}$, let w be such that $w \geq t_f(y,\bar{x})$. We must check that for u such that $w - y \leq u \leq w$, we have $\widehat{f}(u,w;y,\bar{x}) = f(Y(u,w;y),\bar{x})$. This is proven

by induction on u. Take an arbitrary u such that $w - y \leq u \leq w$. Note that there exists $z \in w$ such that $u = z'$ (since $w - y \geq 1$).

Case 1: If $u = z' = w - y$ then $Y(S(;z), w; y) = 0$. Hence, by the definition of $\widehat{f}$, $\widehat{f}(z', w; y, \bar{x}) = G(w, \bar{x}) = f(0, \bar{x}) = f(Y(S(;z), w; y), \bar{x})$.

Case 2: If $u = z' > w - y$ then $Y(S(;z), w; y) \neq 0$. Using the fact that t_f and t_h are monotone we have

$$\begin{aligned} w &\geq t_f(y, \bar{x}) \\ &\geq t_h(Y(z, w; y), \bar{x}, b_f(Y(z, w; y)), \bar{x}) \\ &\geq t_h(Y(z, w; y), \bar{x}, f(Y(z, w; y), \bar{x})). \end{aligned}$$

Hence, using the induction hypothesis $\widehat{f}(z, w; y, \bar{x}) = f(Y(z, w; y), \bar{x})$, we get

$$\begin{aligned} H(w; Y(z, w; y), \bar{x}, \widehat{f}(z, w; y, \bar{x})) &= H(w; Y(z, w; y), \bar{x}, f(Y(z, w; y), \bar{x})) \\ &= h(Y(z, w; y), \bar{x}, f(Y(z, w; y), \bar{x})). \end{aligned}$$

It is easy to see that $Y(z', w; y) = (Y(z, w; y))'$. From this and from the definitions f and $\widehat{f}$, we may conclude that

$$\begin{aligned} \widehat{f}(z', w; y, \bar{x}) &= H(w; Y(z, w; y), \bar{x}, \widehat{f}(z, w; y'\bar{x})) \\ &= h(Y(z, w; y), \bar{x}, f(Y(z, w; y), \bar{x})) \\ &= f(Y(z', w; y), \bar{x}) \end{aligned}$$

as we wanted.

■

The inclusion we want to establish is readily deduced from the previous lemma.

Theorem 2.2. *Let $f(\bar{x})$ be in $\mathcal{E}$. Then $f(\bar{x};)$ is in E.*

Proof. Let $f(\bar{x})$ be in $\mathcal{E}$. F and t_f satisfying $(*)$ are given by the previous lemma. We have already observed that there exists $r \in \mathrm{E}$ s.t. $\forall \bar{x}\, r(\bar{x};) \geq t_f(\bar{x})$. Thus, defining $f(\bar{x}) = F(r(\bar{x};)\bar{x})$, we get the result.

■

2.2.2 $\mathcal{E}$ "contains" the class E

We saw that E is inclusive enough to contain all elementary functions. Now, we must see that it only contains the elementary functions. Firstly, we prove an important lemma, which enables us to bound each function in E by some function in $\mathcal{T}$.

Lemma 2.3. *If $f \in E$ then*

$$\exists q_f \in \mathcal{T}\ s.t.\ \forall \bar{x}, \bar{y}\ f(\bar{x};\bar{y}) \leq q_f(\bar{x}) \cdot max\{max_i\ y_i, 1\} \qquad (**)$$

Proof. We will argue by induction on the complexity of f. For the initial functions the result is trivial.

If f is defined by the safe composition scheme, then $f(\bar{x};\bar{y}) = h(\bar{r}(\bar{x};)\bar{s}(\bar{x};\bar{y}))$ where, by induction hypothesis, $h, \bar{r}, \bar{s}$ satisfy $(**)$. Therefore, we have $q_h, \bar{q}_r$ and $\bar{q}_s$ bounding $h, \bar{r}$ and $\bar{s}$ respectively. Thus,

$$\begin{aligned} f(\bar{x};\bar{y}) &= h(\bar{r}(\bar{x};);\bar{s}(\bar{x};\bar{y})) \\ &\leq q_h(\bar{r}(\bar{x};)) \cdot max\{max_i\ s_i(\bar{x};\bar{y}), 1\} \\ &\leq q_h(\bar{q}_r(\bar{x})) \cdot max\{max_i(q_{s_i}(\bar{x}) \cdot max\{max_j\ y_j, 1\}), 1\} \\ &\leq q_h(\bar{q}_r(\bar{x})) \cdot (\Sigma q_{s_i}(\bar{x})) \cdot max\{max_i\ y_i, 1\} \end{aligned}$$

and, therefore, we can take $q_f(\bar{x}) = q_h(\bar{q}_r(\bar{x})) \cdot \Sigma q_{s_i}(\bar{x})$.

If f is defined by the safe recursion scheme then, by induction hypothesis, we have q_g and q_h in $\mathcal{T}$ bounding g and h respectively. These bounds can be assumed to be positive. Hence, if we define $q_f(w, \bar{x}) = q_h(w, \bar{x})^w \cdot q_g(\bar{x})$ we will have $f(0, \bar{x};\bar{y}) = g(\bar{x};\bar{y}) \leq q_g(\bar{x}) \cdot max\{max_i\ y_i, 1\} = q_f(0, \bar{x}) \cdot max\{max_i\ y_i, 1\}$. Now, let us assume that $f(z, \bar{x};\bar{y}) \leq q_f(z, \bar{x}) \cdot max\{max_i\ y_i, 1\}$. Then we have

$$\begin{aligned} f(z', \bar{x};\bar{y}) &\leq h(z, \bar{x};\bar{y}, f(z, \bar{x};\bar{y})) \\ &\leq q_h(z, \bar{x}) \cdot max\{max_i\ y_i, f(z, \bar{x};\bar{y}), 1\} \\ &\leq q_h(z, \bar{x}) \cdot max\{max_i\ y_i, q_f(z, \bar{x}) \cdot max\{max_i\ y_i, 1\}, 1\} \\ &\leq q_h(z, \bar{x}) \cdot q_f(z, \bar{x}) \cdot max\{max_i\ y_i, 1\} \\ &= q_h(z, \bar{x}) \cdot q_h(z, \bar{x})^z \cdot q_g(\bar{x}) \cdot max\{max_i\ y_i, 1\} \\ &= q_h(z, \bar{x})^{z'} \cdot q_g(\bar{x}) \cdot max\{max_i\ y_i, 1\} \end{aligned}$$

$$\leq q_h(z', \bar{x})^{z'} \cdot q_g(\bar{x}) \cdot max\{max_i\, y_i, 1\}$$
$$\leq q_f(z', \bar{x}) \cdot max\{max_i\, y_i, 1\}.$$

Therefore such q_f satisfy (**), as expected.

■

This lemma is all we need to conclude that E$\subseteq \mathcal{E}$.

Theorem 2.4. *Let $f(\bar{x};\bar{y})$ be in E. Then $f(\bar{x},\bar{y})$ is in $\mathcal{E}$.*

Proof. Once more, the proof is by induction on the complexity of f. There is no problem if f is an initial function.

If f is defined by the safe composition scheme, i.e., $f(\bar{x};\bar{y}) = h(\bar{r}(\bar{x};);\bar{s}(\bar{x};\bar{y}))$, let $h, \bar{r}, \bar{s} \in \mathcal{E}$ be given by the induction hypothesis. At this point we just have to take $f(\bar{x},\bar{y}) = h(\bar{r}(\bar{x}), \bar{s}(\bar{x},\bar{y}))$.

If f is defined by the recursion scheme, i.e.,

$f(0,\bar{x};\bar{y}) = g(\bar{x};\bar{y})$

$f(z',\bar{x};\bar{y}) = h(z,\bar{x};\bar{y}, f(z,\bar{x};\bar{y}))$,

let $g, h \in \mathcal{E}$ and $q_f \in \mathcal{T}$ be given by induction hypothesis and the previous lemma, respectively. Thus, taking

$f(0,\bar{x},\bar{y}) = g(\bar{x},\bar{y})$

$f(z',\bar{x},\bar{y}) = h(z,\bar{x},\bar{y}, f(z,\bar{x},\bar{y}))_{|q_f(z',\bar{x})\cdot(1+\Sigma y_i)}$

we get the result.

■

3 Characterizations of Pspace

As we have already remarked, in this section we effect a change of notation to binary notation. Therefore we will have in our mind the binary tree and all standard notation related with it: $|\, x\,|$ for the length of the sequence/word x, ϵ for the sequence of length zero, xy for the concatenation of the sequence x with the sequence y, the "product" $x \times y = x \cdots x$ (similar in growth to Samuel Buss' smash function, see [5]) for the concatenation of x with itself $|\, y\,|$ times, and x' for the sequence that follows immediately after x when we consider the binary tree ordered according to length and, within the same

length, lexicographically. Finally, $x_{|y} = \begin{cases} x \text{ if } |\, x\,| \leq |\, y\,| \\ z \text{ if } z \subseteq x \wedge |\, z\,| = |\, y\,| \end{cases}$ for the

truncature of x to y, where $z \sqsubseteq x$ abbreviates $\exists y\, zy = x$.

3.1 Classic characterization

It is known that Ptime (the class of functions computable in polynomial time) is the smallest class of functions containing the Projection, i-Concatenation and Conditional functions, and that it is closed under the composition and bounded recursion on notation schemes (see [7] or [8]). It is also known that if we close Ptime under bounded primitive recursion we will get Pspace. This is the case because the number of steps that a Pspace machine may carry is exponential on the length of the input. In other words, Pspace is the smallest class of functions containing the initial functions 1-3 and that is closed under the composition, bounded recursion on notation and bounded primitive recursion:

1) $P_j^n(x_1, \cdots, x_n) = x_j,\ 1 \leq j \leq n$ (Projections)
2) $C_i(x) = xi\ ,\ i = 0, 1$ (i-Concatenation)
3) $Q(x, y, z, w) = \begin{cases} z \text{ if } x \sqsubseteq y \\ w \text{ otherwise} \end{cases}$ (Conditional)

Composition: $f(\bar{x}) = h(\bar{g}(\bar{x}))$

Bounded primitive recursion (exhaustive):
$f(\epsilon, \bar{x}) = g(\bar{x})$
$f(y', \bar{x}) = h(y, \bar{x}, f(y, \bar{x}))_{|t(y,\bar{x})}$

Bounded recursion on notation (over the branches):
$f(\epsilon, \bar{x}) = g(\bar{x})$
$f(yi, \bar{x}) = h_i(y, \bar{x}, f(y, \bar{x}))_{|t(y,\bar{x})},\ i = 0, 1$
where t is a bounding function, i.e., is a function of the smallest class of functions containing the projection functions and the concatenation and "product" functions and which is closed under composition and assignment of values to variables.

It is easy to prove that the bounding functions are monotone and that for all $f \in$ Pspace there exists a polynomial, p_f (with coefficients in $\mathbb{N}$), such that $\mid f(\bar{x}) \mid \leq p_f(\mid \bar{x} \mid)$.

3.1.1 A new characterization

We are going to consider a class of functions, Ps, where the input variables can, once more, occur in two kinds of positions: "normal" and

"safe". As we did before we will write the normal and safe inputs in this order and separate them using a semicolon as follows: $f(\bar{x};\bar{y})$. We say that Ps in the smallest class of functions containing the initial functions $1'-7'$ and which is closed under the safe composition, the safe recursion and the safe recursion on notation schemes:

$1'$) $P_j^{n;m}(x_1,\cdots,x_n;x_{n+1},\cdots,x_{n+m}) = x_j$, $1 \leq j \leq n+m$ (Projections)

$2'$) $C_i(z;x) = \begin{cases} xi \text{ if } \mid x \mid < \mid z \mid \\ x \text{ otherwise} \end{cases}, i = 0,1$ (Bounded i-Concatenation)

$3'$) $D(;\epsilon) = \epsilon \quad D(;xi) = x, i = 0,1$ (Deleting)

$4'$) $P(;\epsilon) = \epsilon \quad P(;x') = x$ (Predecessor)

$5'$) $Q(;x,y,z,w) = \begin{cases} z \text{ if } x \subseteq y \\ w \text{ otherwise} \end{cases}$ (Conditional$_1$)

$6'$) $U(;x) = \begin{cases} 1 \text{ if } \exists y \subseteq x : y1 = x \\ 0 \text{ otherwise} \end{cases}$ (Conditional$_2$)

$7'$) $\times(x,y;) = x \times y$ (Product)

Safe composition: $f(\bar{x};\bar{y}) = h(\bar{r}(\bar{x};);\bar{s}(\bar{x};\bar{y}))$

Safe recursion: $f(\epsilon,\bar{x};\bar{y}) = g(\bar{x};\bar{y})$

$$f(z',\bar{x};\bar{y}) = h(z,\bar{x};\bar{y},f(z,\bar{x};\bar{y}))$$

Safe recursion on notation: $f(\epsilon,\bar{x};\bar{y}) = g(\bar{x};\bar{y})$

$$f(zi,\bar{x};\bar{y}) = h_i(z,\bar{x};\bar{y},f(z,\bar{x};\bar{y})), i = 0,1$$

In the safe composition scheme the absence of some of the functions $\bar{r},\bar{s}$ is allowed.

The initial functions do not increase the length of the variables in safe positions, with exception of the functions $C_i(z;x)$, $i = 0,1$; however, even in this case the increase is bounded by the variable in the normal position . This fact is indispensable, since we have the recursion scheme involving safe positions, but is limits so much our capability to construct functions into Ps that we need to introduce U as an initial function. The asymmetry of safe composition scheme allows us "to change" variables from safe positions to normal positions, but not the opposite. Regarding the recursion schemes, we have the obvious separations between the positions occupied by the recursion variable and by the values obtained recursively.

To prove that Pspace=Ps we are going to follow, almost step by step, the reasoning used in the precedent section.

3.1.2 Ps "contains" Pspace

Firstly, we show:

Lemma 3.1. *If $f \in Pspace$ then there exists $F \in Ps$ and a polynomial p_f such that*

$$\forall \bar{x} \forall w : \mid w \mid \geq p_f(\mid \bar{x} \mid) \Longrightarrow f(\bar{x}) = F(w; \bar{x}). \qquad (*)$$

Proof. The proof is by induction on the complexity of f. We only discuss the case when the function f is obtained by bounded recursion on notation. The other cases are easy or follow methods already used in the previous section.

If f is defined by the bounded recursion on notation then, by induction hypothesis, there exists G, H_0, H_1 in Ps and polynomials p_g, p_{h_0}, p_{h_1} satisfying $(*)$. We assume, for simplicity, that $p_h \equiv p_{h_0} + p_{h_1}$. Since we cannot define $F(w; y, \bar{x})$ by recursion on notation on y, we introduce a "new variabe" z and use it to simulate the recursion on y. To accomplish this, we need to have some functions available in Ps. One of them is $Y(z, w; y) = T_1(T_2(z, w;); y)$, where T_1 is defined by $T_1(\epsilon; x) = x$, $T_1(yi; x) = D(; T_1(y; x))$, $i = 0, 1$ and $T_2(y, x;) = T_1(y; x)$. Informally we may say that $T_1(y; x)$ and $T_2(y, x;)$ are $x_{|y}$; therefore $Y(z, w; y)$ can be kept in mind as $y_{|(w_{|z})}$. To understand the importance of Y in the simulation process just notice that, for each $w, y(\mid w \mid > \mid y \mid)$, when $\mid z \mid$ increases from $\mid w \mid - \mid y \mid$ to $\mid w \mid, Y(z, w; y)$ increases from ϵ to y. Thus, our goal will be to construct into Ps a function $\widehat{f}$, satisfying $\widehat{f}(z, w; y, \bar{x}) = f(Y(z, w; y), \bar{x})$ when $\mid w \mid - \mid y \mid \leq \mid z \mid \leq \mid w \mid$ and $\mid w \mid \geq p_f(\mid y \mid, \mid \bar{x} \mid)$. Since f is defined by recursion on notation, this means that if $Y(z, w; y) = \epsilon$ then $f(Y(z, w; y), \bar{x}) = g(\bar{x})$; otherwise $f(Y(z, w; y), \bar{x})$ is given by h_0 or h_1, depending on whether the last digit of $Y(z, w; y)$ is 0 or 1. Therefore, we need to have in Ps the function that picks up the last digit of $Y(z, w; y)$, That functions is $I(z, w; y) = U(; Y(z1, w; y))$. Now, we can define

$\widehat{f}(\epsilon, w; y, \bar{x}) = \epsilon$

$$\widehat{f}(zj, w; y, \bar{x}) = \begin{cases} G(w; \bar{x}) \text{ if } Y(z1, w; y) = \epsilon \\ H_0(w; I(z, w; y), Y(z, w; y), \bar{x}, \widehat{f}(z, w; y, \bar{x}))) \text{ if } I(z, w; y) = 0 \\ H_1(w; I(z, w; y), Y(z, w; y), \bar{x}, \widehat{f}(z, w; y, \bar{x}))) \text{ if } I(z, w; y) = 1 \end{cases}$$

Or more formally

$\widehat{f}(\epsilon, w; y, \bar{x}) = \epsilon$
$\widehat{f}(zj, w; y, \bar{x}) = Q(; Y(z1, w; y), \epsilon, G(w; \bar{x}),$
$\widehat{h}(w; I(z, w; y), Y(z, w; y), \bar{x}, \widehat{f}(z, w; y, \bar{x})))$
where $\widehat{h}(w; i, a, \bar{b}, c) = Q(; i, 1, H_1(w; a, \bar{b}, c), H_0(w; a, \bar{b}, c))$.

Therefore, we just have to put $F(w; y, \bar{x}) = \widehat{f}(w, w; y, \bar{x})$ and $p_f(\mid y \mid, \mid \bar{x} \mid) = p_h(\mid y \mid, \mid \bar{x} \mid, b_f(\mid y \mid, \mid \bar{x} \mid)) + p_g(\mid \bar{x} \mid) + \mid y \mid +1$, where b_f is a polynomial bounding the lengths of the outputs. It is easy to see that when $\mid z \mid$ increases from $\mid w \mid - \mid y \mid$ to $\mid w \mid$, $\widehat{f}$ simulates the process of recursion on y in the function f.

Given y and $\bar{x}$, let w be such that $\mid w \mid \geq p_f(\mid y \mid, \mid \bar{x} \mid)$. We prove, by induction on $\mid u \mid$, that if $\mid w \mid - \mid y \mid \leq \mid u \mid \leq \mid w \mid$ then

$$\widehat{f}(u, w; y, \bar{x}) = f(Y(u, w; y), \bar{x}).$$

This implies that

$$F(w; y, \bar{x}) = \widehat{f}(w, w; y, \bar{x}) = f(y, \bar{x}),$$

as expected. Let u be such that $\mid w \mid - \mid y \mid \leq \mid u \mid \leq \mid w \mid$, and take $z \in \{0,1\}^*$ and $j \in \{0,1\}$ such that $u = zj$.

Case 1: If $\mid u \mid = \mid zj \mid = \mid w \mid - \mid y \mid$ then $Y(zj, w; y) = \epsilon$, and so $\widehat{f}(zj, w; y, \bar{x}) = G(w, \bar{x}) = f(\epsilon, \bar{x}) = f(Y(zj, w; y), \bar{x})$.

Case 2: If $\mid u \mid = \mid zj \mid > \mid w \mid - \mid y \mid$ then $Y(zj, w; y) \not\subseteq \epsilon$. Assuming $\widehat{f}(z, w; y, \bar{x}) = f(Y(z, w; y), \bar{x})$ we have

$$\begin{aligned} \mid w \mid &\geq p_f(\mid y \mid, \mid \bar{x} \mid) \\ &\geq p_{h_i}(\mid Y(z, w; y) \mid, \mid \bar{x} \mid, b_f(\mid Y(z, w; y) \mid, \mid \bar{x} \mid)) \\ &\geq p_{h_i}(\mid Y(z, w; y) \mid, \mid \bar{x} \mid, \mid f(Y(z, w; y), \bar{x}) \mid) \end{aligned}$$

and so, by the general induction hypothesis over h_i,

$$\begin{aligned} H_i(w; Y(z, w; y), \bar{x}, \widehat{f}(z, w; y, \bar{x})) &= H_i(w; Y(z, w; y), \bar{x}, f(Y(z, w; y), \bar{x})) \\ &= h_i(Y(z, w; y), \bar{x}, f(Y(z, w; y), \bar{x})). \end{aligned}$$

Hence, by definition of $\widehat{f}$ and by the observation that $Y(zj, w; y) = C_{I(z,w;y)}(; Y(z, w; y))$, we have

$$\begin{aligned} \widehat{f}(zj, w; y, \bar{x}) &= H_{I(z,w;y)}(w; Y(z, w; y), \bar{x}, \widehat{f}(z, w; y, \bar{x})) \\ &= h_{I(z,w;y)}(Y(z, w; y), \bar{x}, f(Y(z, w; y), \bar{x})) \\ &= f(Y(zj, w; y), \bar{x}) \end{aligned}$$

The proof of the lemma is finished.

■

Theorem 3.2. *Let $f(\bar{x})$ be in Pspace. Then $f(\bar{x};)$ is in Ps.*

Proof. Let $f(\bar{x})$ be in Pspace and, by the previous lemma, take F and p_f satisfying $(*)$. It is easy to show that there exists $r \in Ps$ such that $\forall \bar{x} \mid r(\bar{x};) \mid \geq p_f(\mid \bar{x} \mid)$. Thus, just put $f(\bar{x}) = F(r(\bar{x};);\bar{x})$.

■

3.1.3 Pspace "contains" Ps

The inclusion $Ps \subseteq Pspace$ is a simple consequence of the fact that it is possible to bound polynomially every functions in Ps:

Lemma 3.3. *If $f \in Ps$ then there exists a polynomial q_f such that*

$$\forall \bar{x}, \bar{y} \mid f(\bar{x};\bar{y}) \mid \leq max\{q_f(\mid \bar{x} \mid), max_i \mid y_i \mid\}. \qquad (**)$$

Proof. The proof is by induction on the complexity of f. The initial functions pose no problems.

If f is defined by the safe composition scheme, then we have $q_h, \bar{q}_r, \bar{q}_s$ satisfying $(**)$ and, therefore, we may take,

$$q_f(\mid \bar{x} \mid) = q_h(q_r(\mid \bar{x} \mid)) + \sum_i q_{s_i}(\mid \bar{x} \mid).$$

If f is defined by safe recursion then, considering $q_f(\mid z \mid, \mid \bar{x} \mid) = q_h(\mid z \mid, \mid \bar{x} \mid) + q_g(\mid \bar{x} \mid)$, the result follows by induction on the recursion variable, since q_g, q_h verify $(**)$. We have $\mid f(\epsilon, \bar{x};\bar{y}) \mid \leq max\{q_f(\mid \epsilon \mid, \mid \bar{x} \mid), max_i \mid y_i \mid\}$ and, since $\mid f(u, \bar{x};\bar{y}) \mid \leq max\{q_f(\mid u \mid, \mid \bar{x} \mid), max_i \mid y_i \mid\}$, we get,

$$\begin{aligned}
\mid f(u', \bar{x};\bar{y}) \mid &= \mid h(u, \bar{x};\bar{y}, f(u, \bar{x};\bar{y})) \mid \\
&\leq max\{q_h(\mid u \mid, \mid \bar{x} \mid), max\{max_i \mid y_i \mid, f(u, \bar{x};;\bar{y}))\}\} \\
&\leq max\{q_h(\mid u \mid, \mid \bar{x} \mid), max\{max_i \mid y_i \mid, q_h(\mid u \mid, \mid \bar{x} \mid) + q_g(\mid \bar{x} \mid)\}\} \\
&\leq max\{q_h(\mid u \mid, \mid \bar{x} \mid) + q_g(\mid \bar{x} \mid), max_i \mid y_i \mid\} \\
&\leq max\{q_h(\mid u' \mid, \mid \bar{x} \mid) + q_g(\mid \bar{x} \mid), max_i \mid y_i \mid\} \\
&\leq max\{q_f(\mid u' \mid, \mid \bar{x} \mid), max_i \mid y_i \mid\}
\end{aligned}$$

If f is defined by recursion on notation then, if we set $q_h \equiv q_{h_0} + q_{h_1}$ and $q_f(|\ z\ |, |\ \bar{x}\ |) = q_h(|\ z\ |, |\ \bar{x}\ |) + q_g(|\ \bar{x}\ |)$, we may carry on the result by induction on variable recursion length, since q_g, q_{h_0}, q_{h_1} satisfy (**). Thus,

$$\begin{aligned} |\ f(\epsilon, \bar{x}; \bar{y})\ | &= |\ g(\bar{x}; \bar{y})\ | \\ &\leq max\{q_g(|\ \bar{x}\ |), max_i\ |\ y_i\ |\} \\ &= max\{q_f(|\ \epsilon\ |, |\ \bar{x}\ |), max_i\ |\ y_i\ |\} \end{aligned}$$

Now, since $|\ f(z, \bar{x}; \bar{y})\ | \leq max\{q_f(|\ z\ |, |\ \bar{x}\ |), max_i\ |\ y_i\ |\}$, we get

$$\begin{aligned} |\ f(zi, \bar{x}; \bar{y})\ | &= |\ h_i(z, \bar{x}; \bar{y}, f(z, \bar{x}; \bar{y}))\ | \\ &\leq max\{q_h(|\ z\ |, |\ \bar{x}\ |), max(max_i\ |\ y_i\ |, |\ f(z, \bar{x}; \bar{y})\ |)\} \\ &\leq max\{q_h(|\ z\ |, |\ \bar{x}\ |), max(max_i\ |\ y_i\ |, max\{q_f(|\ z\ |, |\ \bar{x}\ |), max_i\ |\ y_i\ |)\})\} \\ &= max\{q_h(|\ z\ |, |\ \bar{x}\ |), max\{q_f(|\ z\ |, |\ \bar{x}\ |), max_i\ |\ y_i\ |\}\} \\ &= max\{q_h(|\ z\ |, |\ \bar{x}\ |), max\{q_h(|\ z\ |, |\ \bar{x}\ |) + q_g(|\ \bar{x}\ |), max_i\ |\ y_i\ |\}\} \\ &\leq max\{q_h(|\ z\ |, |\ \bar{x}\ |) + q_g(|\ \bar{x}\ |), max_i\ |\ y_i\ |\} \\ &\leq max\{q_h(|\ zi\ |, |\ \bar{x}\ |) + q_g(|\ \bar{x}\ |), max_i\ |\ y_i\ |\} \\ &\leq max\{q_f(|\ zi\ |, |\ \bar{x}\ |), max_i\ |\ y_i\ |\} \end{aligned}$$

Therefore, $\forall \bar{x}, \bar{y}\ |\ f(\bar{x}; \bar{y})\ | \leq max\{q_f(|\ \bar{x}\ |), max_i\ |\ y_i\ |\}$.

■

At this point is obvious that the length of the functions in Ps is polynomially bounded. It can also be easily shown that, for all polynomials, p there exists a bounding function t such that $p(|\ \bar{x}\ |) = |\ t(\bar{x})\ |$. For example, if $p(x_1, x_2) = x_1 \cdot x_2^2 + 2 \cdot x_2$ then, for $t(x_1, x_2) = (x_1 \times x_2 \times x_2)(11 \times x_2)$, we will have $p(|\ x_1\ |, |\ x_2) = |\ t(x_1, x_2)\ |$. Therefore, in order to prove the following theorem, we just have to check that it is possible to bound the safe recursion schemes.

Theorem 3.4. *Let $f(\bar{x}; \bar{y})$ be in Ps. Then $f(\bar{x}, \bar{y})$ is in Pspace.*

Proof. The proof is standard and need not be reported here in detail. The key step is when f is obtained by safe recursion or by safe recursion on notation. In both cases we know, by the previous lemma, that for some polynomial q_f we have

$$\forall z, \bar{x}, \bar{y}\ |\ f(z, \bar{x}; \bar{y})\ | \leq max\{q_f(|\ z\ |, |\ \bar{x}\ |), max_i\ |\ y_i\ |\}.$$

Thus, defining $p(|\ z\ |, |\ \bar{x}\ |, |\ \bar{y}\ |) = q_f(|\ z\ |, |\ \bar{x}\ |) + \Sigma_i\ |\ y_i\ |$, there is a bounding function t such that $p(|\ z\ |, |\ \bar{x}\ |, |\ \bar{y}\ |) = |\ t(z, \bar{x}, \bar{y})\ |$. Therefore, we define f by the correspondent bounded recursion scheme, with bound t.

■

We have just established a characterization of Pspace without bounds. To finish this paper some final remarks are in order.

Remarks.

Some people expressed their concern about our inclusion of the product function amongst the initial functions of Ps. They would prefer to have only initial functions of linear growth. We observed in subsection 3.1.1 that we must be very careful about operations involving safe positions because - having safe recursion involving safe position - if they increase the safe input lengths even just a bit, we would get functions of exponential growth, which lie outside Pspace. Therefore, in order to remove the product from the initial functions we seem to have to introduce an intermediate input position, say semi-safe, and use it to construct the product. Therefore, if we start with simpler initial functions we will arrive at a more elaborate characterization. Let us give a brief glance over this alternative characterization of Pspace. Here, there are three kinds of input positions in the functions: "normal", "semi-safe" and "safe". We write the normal, semi-safe and safe inputs by this order and separate them by semicolons. We say that Ps^* is the smallest class of functions containing the following initial functions $1' - 7'$ and which is closed under the safe composition, the safe recursion and the double recursion on notation schemes:

$1'$) $P_j^{n;m;l}(x_1, \cdots, x_n; x_{n+1}, \cdots, x_{n+m}; x_{n+m+1}, \cdots, x_{n+m+l}) = x_j$,
$1 \leq j \leq n+m+l$ (Projections)

$2'$) $C_i(z;;x) = \begin{cases} xi \text{ se } |\ x\ | < |\ z\ | \\ x \text{ otherwise} \end{cases}, i = 0, 1$ (Bounded i-Concatenation)

$3'$) $D(;;\epsilon) = \epsilon D(;;xi) = x, i = 0, 1$ (Deleting)

$4'$) $P(;;\epsilon) = \epsilon P(;;x') = x$ (Predecessor)

$5'$) $Q(;;x, y, z, w) = \begin{cases} z \text{ se } x \subseteq y \\ w \text{ otherwise} \end{cases}$ (Conditional$_1$)

$6'$) $U(;;x) = \begin{cases} 1 \text{ se } \exists y \subseteq x : y1 = x \\ 0 \text{ otherwise} \end{cases}$ (Conditional$_2$)

$7'$) $C_i(;x;) = xi, i = 0, 1$ (i-Concatenation)

Safe composition: $f(\bar{x};\bar{y};\bar{z}) = h(\bar{s}(\bar{x};;);\bar{r}(\bar{x};\bar{y};);\bar{t}(\bar{x};;\bar{z}))$

Safe recursion: $f(\epsilon,\bar{x};;\bar{y}) = g(\bar{x};;\bar{y})$
$f(z',\bar{x};;\bar{y}) = h(z,\bar{x};;\bar{y},f(z,\bar{x};;\bar{y}))$

Double recursion on notation:
$f(\epsilon,\bar{x};\bar{y};\bar{z}) = g(\bar{x};\bar{y};\bar{z})$
$f(wi,\bar{x};\bar{y};\bar{z}) = h_i(w,\bar{x};\bar{y},f(w,\bar{x};\bar{y};\bar{\epsilon});\bar{z},f(w,\bar{x};\bar{\epsilon};\bar{z})),\ i=0,1$

The goal of the double recursion on notation scheme is to join two schemes in one. However, it could be replaced by the two following schemes:

Semi-safe recursion on notation:
$f(\epsilon,\bar{x};\bar{y};) = g(\bar{x};\bar{y};)$
$f(wi,\bar{x};\bar{y};) = h_i(w,\bar{x};\bar{y},f(w,\bar{x};\bar{y};);),i=0,1$

Safe recursion on notation:
$f(\epsilon,\bar{x};;\bar{z}) = g(\bar{x};;\bar{z})$
$f(wi,\bar{x};;\bar{z}) = h_i(w,\bar{x};;\bar{z},f(w,\bar{x};;\bar{z})),i=0,1$

The basic facts are as follows:

Lemma* 3.1. *If $f \in Pspace$ then there exists $F \in Ps^*$ and a polynomial p_f such that*

$$\forall\bar{x}\forall w :\mid w \mid \geq p_f(\mid \bar{x} \mid) \Longrightarrow f(\bar{x}) = F(w;;\bar{x}). \qquad (*)$$

Theorem* 3.2. *Let $f(\bar{x})$ be in Pspace. Then $f(\bar{x};;)$ is in Ps^*.*

Lemma* 3.3. *If $f \in Ps^*$ then exists a polynomial q_f such that*

$$\forall\bar{x},\bar{y},\bar{z} \mid f(\bar{x};\bar{y};\bar{z}) \mid \leq max\{q_f(\mid \bar{x} \mid) + max_i \mid y_i \mid, max_i \mid z_i \mid\}. \qquad (**)$$

Theorem* 3.4. *Let $f(\bar{x};\bar{y};\bar{z})$ be in Ps^*. Then $f(\bar{x},\bar{y},\bar{z})$ is in Pspace.*

Acknowledgments.

I would like to thank Fernando Ferreira for introducing me to the subject and for his helpful suggestions.

References

[1] Balcázar J., Díaz J., Gabarró J. *Structural Complexity II*, Springer-Verlag, 1990.

[2] Bellantoni S. and Cook S. (1992). *A New Recursion-Theoretic Characterization of Polytime Functions*, Computational Complexity, vol. 2, pp. 97-110.

[3] Bellantoni S. (1993). *Predicative Recursion and Computational Complexity*, Ph. D. Dissertation, University of Toronto.

[4] Bellantoni S. (1995). *Predicative Recursion and The Polytime Hierarchy*, Feasible Mathematics II, ed. P. Clote and J. B. Remmel

[5] Buss S. *Bounded Arithmetic*, Ph. D. Dissertation, Bibliopolis (1986).

[6] Csillag P. (1947). *Eine Bemerkung zur Auflosung der eingeschachtelten Rekursion*, Acta Sci. Math. Szeged 11, pp. 169-173.

[7] Ferreira F. *Polynomial Time Computable Arithmetic and Conservative Extensions*, Ph. D. Dissertation, Pennsylvania State University (1988).

[8] Ferreira F. (1990). *Polynomial Time Computable Arithmetic*, Contemporary Mathematics, Volume 106, pp. 137-156.

[9] Kalmar L. (1943). *Egyszerü példa eldönthetetlen aritmetikai problémára*, (in Hungarian with German abstract), Mate és Fizikai Lapok 50, pp. 1-23.

[10] Rose H. *Subrecursion: functions and hierarchies*, Clarendon Press, Oxford, 1984.

CMAF - Universidade de Lisboa
Av. Prof. Gama Pinto 2
1699 Lisboa
Portugal
Fax: 351-1-7954288
e-mail: isarocha@ptmat.lmc.fc.ul.pt

Recibido: 30 de Junio de 1995
Revisado: 9 de Abril de 1996