European Society of Computational Methods in Sciences and Engineering (ESCMSE)

Journal of Numerical Analysis, Industrial and Applied Mathematics (JNAIAM)

vol. 1, no. 3, 2006, pp. 273-280
ISSN 1790–8140

Primitive Recursion and μ-Recursivity[1]

A. Garrido[2]

Department of Fundamental Mathematics, Faculty of Sciences, UNED, Madrid, Spain

Abstract: We analyze the different types of recursivity and their mutual relationships. Finally, we will use the Ackermann - Peter function to prove that the μ-recursivity does not imply the primitive recursion, a fact with crucial consequences in Automata Theory.

Keywords: Recursive Functions Theory, Mathematical Analysis, Automata Theory.

Mathematics Subject Classification: 03D20, 68Q1S

1 Primitive Recursion

We comment now some of Recursive Functions Theory.

As basic element, the function f. Its domain will be composed by m-uples of non-negative integers, and its range must be n-uples of the same type of numbers. So:

$$f : (N^*)^m \to (N^*)^n$$

where:

$$N^* = N \cup \{0\}$$

The first distinction will be between total and partial functions, according to that the original set coincides or not with their domain.

We can construct f, *primitive recursive function*, from the set of initial functions, clearly computable, by standard operations, such as combination, composition and primitive recursion, applied in finite number. Into the set of *initial functions,* we have the subsequent:

$$\textit{zero function } \zeta\text{; } \textit{successor function } \sigma\text{; and } \textit{projection functions, } \pi_j^i \ (i \succeq j)\,.$$

[1]Published electronically December 22, 2006
[2]Corresponding author. E-mail: agbmv@telefonica.net

The *recursivity* can be introduced, from f and g, by:

$$f(x,\ 0) = g(x)$$

$$f(x,\ y+1) = h(x,\ y,\ f(x,y))$$

Primitive recursive functions will be, for instance:

predecessor, *mult*, *exp*, *monus*, *eq*; their negation ($\neg eq$); the *tabular* functions; *coc*(not *div*)

and so on.

In the classification into the set of functions, according to more restricted conditions each time, we have:

$$\{initial\ functions\} \subset \{primitive\ recursive\ functions\} \subset \{computable\ functions\}$$

Also, we know that if f is primitive recursive, then f is total.

That is:

$$\{primitive\ recursive\ functions\} \subset \{total\ functions\}$$

For instance, the initial functions. But the converse of the inclusion above is not true:

There exist computable functions no primitive recursive.

Because they are only partial, no total functions.

So is the *div* function (note that is very different to the previous *coc*). Such function is not primitive recursive.

2 μ - Recursivity

At first, it might be thought that there is coincidence between primitive recursive function and computable total (also called μ *-recursive*) function classes.

But this is false. There exists a famous *counterexample*, due to *Wilhelm Ackerman*: the function A, named after him.

It is very easy to prove the existence of a $\mu - recursive$ *(total and computable)* function, $f : N^* \to N^*$, which is not recursive primitive.

Abridged proof: Any recursive primitive function, f, can be obtained from initial functions, combining, composing and by primitive recursion method, applied a finite number of times. Therefore, f can be defined by a finite string of symbols.

Such primitive recursive functions, according to this, can be ordered by their length, and amongst those with the same length, by lexicographic ordering. So, we reach an ordered collection:

$$F = \{f_i\}_{i \in N^*} = \{f_0,\ f_1,\ f_2 ...\}$$

But it is possible to construct a μ-recursive function such that it does not belong to F. Such function can be:

$$f(i) \equiv f_i(i) + 1, \forall i \in N^*$$

Our f is *total,* because for each $i \in N^*$, $f(i)$ exists.

And f is *computable,* because we can always find the value of $f(i)$. All the initial functions are computable, and also their composition, combination or primitive recursion, taken in finite number. By definition, it suffices to take the sucessor of $f_i(i)$.

But the primitive recursion fails:
If f is so, then:

$$f \equiv f_n, \text{ for some } n \in N^*$$

However, in such case:

$$f(n) = f_n(n)$$

And this falls in total contradiction with its definition.

In effect, because then we have simultaneously:

$$f(n) \equiv f_n(n) + 1$$

and the aforementioned equality. Therefore,

$$f_n(n) = f_n(n) + 1$$

for some $n \in N^*$.
But this implies obviously that:

$$0 = 1\,(!)$$

So, *μ-recursivity does not imply primitive recursion.*

□

The *Ackermann's function:*

$$A : (N^*)^2 \rightarrow N^*$$

can be defined by three equations:

$$A(0,\ y) = y + 1$$
$$A(x+1,\ 0) = A(x,\ 1)$$
$$A(x+1,\ y+1) = A(x,\ A(x+1,\ y))$$

Observe that this function was not actually written in such form by Wilhelm Ackermann.
Because Ackermann took the $z - fold$ exponentiation of x with y :

$$A(x,\ y,\ z) = x^{y^{y^{\cdot^{\cdot^{z}\cdot^{y}}}}} = (x \rightarrow y \rightarrow z)$$

The last, expressed in the Conway chained arrow notation.
If $z = 1$, we have:

$$A(x,\ y,\ 1) = x^y$$

But if $z = 2$, then:

$$A\,(x,\ y,\ 2) = x^{x^{x^{\cdot^{\cdot^{\cdot^{y\ \cdot x}}}}}} = {}^{y}\,x$$

the *tetration* of x by y. And so on.

This results a good example of $\mu-$*recursive function* which *is not primitive recursive.* Already conjectured by Hilbert, in his work *On the Infinite.* Later, Rosza Péter and Raphael Robinson expressed A as function of two variables.

A grows extremely fast. In each step, y will decrease. Because y is increases, then x would decrease. Each time that y reaches the value 0, x decreases. Therefore, also x must finally reach zero. But when x decreases, it does not exist upper bound to y. We can see that:

$$A\,(0,\ y) = y + 1$$
$$A\,(1,\ y) = y + 2 = 2 + (y + 3) - 3$$
$$A\,(2,\ y) = 2y + 3 = 2 \cdot (y + 3) - 3$$
$$A\,(3,\ y) = 8 \cdot 2^{y} - 3 = 2^{y+3} - 3$$

Observe the formation law, according to the steps:

$$+ \rightarrow \cdot \rightarrow\ \hat{}\ \rightarrow ...$$

We can prove easily its μ-recursivity (that is, *computability* and *total character*).

The *proof* of its total character is reached by induction on the pair of entries, lexicographically ordered.

And A is obviously computable, because A is a total function, according to its definition.

But A *fails in the primitive recursion* ($P\ R$):

Proof. We can see that given:

$$f : (N^{*})^{n} \rightarrow (N^{*})^{m}$$

there is a $\tau \in N$, such that for each $\mathbf{x} \in (N^{*})^{n}$,

$$f\,(\mathbf{x}) < A\,[\tau,\ \Sigma_{\mathbf{x}}]$$

where Σ_x is the sum of all the components of $\mathbf{x}$.

From this inequality, it is almost immediate. Because if A were $P\ R$, then we can define:

$$f \equiv A \circ \left(\pi_1^1 \times \pi_1^1\right)$$

which could be *also PR*.

Therefore:

$$\exists \tau \in N : f\,(\mathbf{x}) < A\,[\tau,\ \mathbf{x}]\,, \forall \mathbf{x} \in (N^{*})^{n}$$

Concretely, if we take:

$$\mathbf{x} = \tau \Rightarrow f(\tau) < A[\tau,\ \tau]$$

But as:

$$f(\tau) = \left\{A \circ \left(\pi_1^1 \times \pi_1^1\right)\right\}(\tau) = A\left[\pi_1^1(\tau),\ \pi_1^1(\tau)\right] = A[\tau,\ \tau]$$

we have:

$$A[\tau,\ \tau] < A[\tau,\ \tau]$$

And it is obviously impossible, because of the strictness of the inequality. Therefore: *A is not P R.*

For the steps, we need the subsequent mathematical expressions, which can be also enunciated as Lemmas:

$$\forall x, x',\ y \in N^*,$$

$$A(1,\ y) = y + 2,\ A(2,\ y) = 2y + 3,\ y + 1 \leq A(x,\ y),$$

$$A(x,\ y) \leq A(x,\ y+1),\ A(x,\ y+1) \leq A(x+1,\ y), A(x,\ y) < A(x+1,\ y),$$

$$A(x,\ y) + A(x',\ y) < A(\max\{x,\ x'\} + 4,\ y),\ A(x,\ y) + y < A(x+4,\ y)$$

easily provable by induction.

With the aforementioned steps, we can reach the final *result:*

$$\forall f : (N^*)^n \to (N^*)^m, \exists \tau : \forall \mathbf{x} \in (N^*)^n, f(\mathbf{x}) < A(\tau,\ \Sigma_{\mathbf{x}})$$

where if:

$$\mathbf{x} = (x_1,\ x_2,\ ...,\ x_n) \Rightarrow \Sigma_{\mathbf{x}} = x_1 + x_2 + ... + x_n$$

This last *result* can be proved in this way:

Suppose that f is primitive recursive. Then, there will be a *finite* number of compositions ($\circ$), combinations ($\times$) and primitive recursion acting on initial functions. But it is possible to apply the induction procedure on the number of such operations, N.

If we dispose of only one initial function, it can be sufficient with $\tau = 0$.

For instance, it would be:

$$f = \varsigma\,(null\ function)$$

In such case:

$$f(.) = \varsigma(.) \equiv 0 < 1 = A(0,\ 0) \equiv A(\tau,\ 0)$$

Also, it can be some projection function:

$$f = \pi_i^n,\ \text{ with } \mathrm{i} \leq \mathrm{n}$$

Then:

$$f(\mathbf{x}) = \pi_i^n(\mathbf{x}) = x_i \leq x_1 + x_2 + ... + x_n = \Sigma_{\mathbf{x}} < \Sigma_{\mathbf{x}} + 1 = A(0,\ \Sigma_{\mathbf{x}}) \equiv A(\tau,\ \Sigma_{\mathbf{x}})$$

If f is the successor function: $f = \sigma$, it suffices with $\tau = 1$, because:

$$f(\mathbf{x}) = \sigma(\mathbf{x}) \equiv \mathbf{x} + 1 = A(0,\ \mathbf{x}) < A(1,\ \mathbf{x}) \equiv A(\tau,\ \mathbf{x})$$

Suppose now that we obtain f by $\circ$, $\times$ or $P\ R$, or some of them, acting k times on initial functions. Also we can suppose that the *Propos.* is true from 0 to $k-1$ of such actions. As the output has only one component, the last operation will be or $\circ$ or $P\ R$.

In the next step, we will analyze both cases, applying induction and precedent inequalities, to obtain the proof. So, $f = h \circ g$ or:

$$f(\mathbf{x},\, 0) = g(\mathbf{x})\, f(\mathbf{x},\, y+1) = h(\mathbf{x},\, y,\ f(\mathbf{x},\, y))$$

but f could never be a combination, because: $f = h \times g$ implies more than one component as output.

First case:

Let f be composed from g and h. In general, we have $m \in N$ components for g. So:

$$g = g_1 \times g_2 \times ... \times g_m \equiv \Pi_{i=1}^{m} g_i$$

the combination of m functions $\{g_i\}_{i=1}^{m}$.

If we apply now the induction hypothesis:

$$\exists \{N_i\}_{i=1}^{m} \subset N :$$

$$\begin{array}{r} g_1(\mathbf{x}) < A(N_1,\ \Sigma_{\mathbf{x}}) \\ g_2(\mathbf{x}) < A(N_2,\ \Sigma_{\mathbf{x}}) \\ ..... \\ g_m(\mathbf{x}) < A(N_m,\ \Sigma_{\mathbf{x}}) \end{array}$$

But also:

$$\exists \tau_0 \in N : h(\mathbf{x}) < A(\tau_0,\ \Sigma_{\mathbf{x}})$$

Therefore:

$$f\ (\mathbf{x}) \equiv (h \circ g)(\mathbf{x}) = \{h \circ (g_1 \times g_2 \times ... \times g_m)\}(\mathbf{x}) = \{h \circ (\Pi_{i=1}^{m} g_i)\}(\mathbf{x}) =$$

$$= h\ [g_1(\mathbf{x}),\ g_2(\mathbf{x}),\ ...,\ g_m(\mathbf{x})] < A(\tau_0,\ g_1(\mathbf{x}) + g_2(\mathbf{x}) + ... +\ g_m(\mathbf{x})) =$$

$$= A(\tau_0,\ \Sigma_{i=1}^{m} g_i(\mathbf{x})) < A(\tau_0,\ A(M,\ \Sigma_{\mathbf{x}})) < A(\tau_0,\ A(M+1,\ \Sigma_{\mathbf{x}})) <$$

$$< A[\max\{\tau_0,\ M\},\ A(\max\{\tau_0,\ M+1\},\ \Sigma_{\mathbf{x}})] <$$

$$< A[\max\{\tau_0,\ M\},\ \Sigma_{\mathbf{x}} + 1] < A[\max\{\tau_0,\ M\} + 2,\ \Sigma_{\mathbf{x}}]$$

The referred M will be:

$$M \equiv (\max\ \{N_i\}_{i=1}^{m}) + 4(m-1)$$

And with these steps it is sufficient, because taking:

$$\tau \equiv \max\{\tau_0,\ M\} + 2$$

we obtain our result:

$$f(\mathbf{x}) < A(\tau,\ \Sigma_{\mathbf{x}}), \forall \mathbf{x} \in (N^*)^n$$

Second case:

Let f be defined by $P\ R$, on g and h,as expressed above. We know that g and h are obtained through at most k combinations and/or compositions and/or $P.\ R.$ Therefore:

$$\exists M_1, M_2 \in N^* :$$

$$g(\mathbf{z}) < A(M_1,\ \Sigma_z), \forall \mathbf{z} \in (N^*)^{n-1}$$

$$h(\mathbf{z}) < A(M_2,\ \Sigma_z), \forall \mathbf{z} \in (N^*)^{m+1}$$

and we proceed to prove the existence of $\tau \in N^*$ such that:

$$f(\mathbf{x},\ y) < A(\tau,\ \Sigma_{\mathbf{x}} + y)$$

for each pair $(\mathbf{x},\ y) \in (N^*)^n$.

It suffices taking:

$$\tau > \max\{M_1,\ M_2\} + 4$$

and applying induction on y.

If $y = 0$, then:

$$f(\mathbf{x},\ 0) \leq f(\mathbf{x},\ 0) + \Sigma_{\mathbf{x}} \equiv g(\mathbf{x}) + \Sigma_{\mathbf{x}} <$$

$$< A(M_1,\ \Sigma_{\mathbf{x}}) + \Sigma_{\mathbf{x}} < A(M_1 + 4,\ \Sigma_{\mathbf{x}}) < A(\tau,\ \Sigma_{\mathbf{x}})$$

Using now the induction hypothesis, we will suppose that the inequality is true until the value $y = k$, and we prove it for $y = k + 1$:

$$f(\mathbf{x},\ k+1) \equiv h[\mathbf{x},\ k,\ f(\mathbf{x},\ k)] < A[M_2,\ \Sigma_{\mathbf{x}} + k + f(\mathbf{x},\ k)] <$$

$$< A[M_2,\ A(0,\ \Sigma_{\mathbf{x}} + k) + f(\mathbf{x},\ k)] < A[M_2,\ A(0,\ \Sigma_{\mathbf{x}} + k) + A(\tau,\ \Sigma_{\mathbf{x}} + k)] <_{\tau > 0}$$

$$< A[M_2,\ A(\tau,\ \Sigma_{\mathbf{x}} + k) + A(\tau,\ \Sigma_{\mathbf{x}} + k)] < A[M_2,\ 2A(\tau,\ \Sigma_{\mathbf{x}} + k) + 3] =$$

$$= A[M_2,\ A(2,\ A(\tau,\ \Sigma_{\mathbf{x}} + k))] \leq A[M_2,\ A(M_2 + 1,\ A(\tau,\ \Sigma_{\mathbf{x}} + k))] =$$

$$= A[M_2 + 1, A(\tau,\ \Sigma_{\mathbf{x}} + k) + 1] < A[M_2 + 2, A(\tau,\ \Sigma_{\mathbf{x}} + k)] <$$

$$< A[\tau - 1, A(\tau,\ \Sigma_{\mathbf{x}} + k)] = A[\tau, \Sigma_{\mathbf{x}} + (k+1)]$$

providing validity to our inequality, for any $y \in N^*$.

□

So, the final classification will be established:

$$\{initial\ functions\} \subset \{primitive\ recursive\ functions\} \subset$$

$$\subset \{\mu - recursive\ functions\} \subset \{computable\ functions\}$$

improving the initial sequence by the new class of functions.

References

[1] Ambos-Spies, K., et al.: *Genericity and Measure for exponential time*. Theoretical Computer Science, 168 (1): 3-19, 1996.

[2] Borodin, A.: On relating time and space to size and depth. *SIAM Journal on Computing,* 6: 733-744, 1977.

[3] Buhrman, H., et al.: Separating complexity classes using autoreductibility. *SIAM Journal on Computing,* 29(5): 1497-1520, 2000.

[4] Fortnow, L.: Counting complexity in L. Hemaspaandra and A. Selman (Eds), *Complexity Theory Retrospective II,* pp. 81-107. Springer-Verlag, 1997.

[5] Garrido, A.: Complexity of Algorithms. Intern. Conf. Eps-MsO. Atenas, 2005. Published in their Proceedings, on CD-Rom and paper.

[6] Garrido, A.: *Hierarchy Theorems in Complexity Classes*, Conference EuroComb'05. TU Berlin, 2005.

[7] Garrido, A.: *Heuristic Functions, Primitive Recursion and Recursivity.* ICCMSE'2005. Lecture Series on Computation, VSP-Brill, vol 4A, pp 234-238.

[8] Hartmanis, J., et al.: On the computational complexity of algorithms. *Transactions of the AMS,* 117: pp. 285-306, 1965.

[9] Ladner, R.: On the structure of polynomial time reducibility. *Journal of the ACM,* 22, pp. 155-171, 1975.

[10] Lutz, J.: Category and measure in complexity classes. *SIAM Journal on Computing,* 19 (6): 1100-1131, 1990.

[11] Lutz, J.: The quantitative structure of exponential time. In *Complexity Theory Retrospective II,* pp. 225-260. Springer-Verlag, 1997.

[12] Toda, S., et al.: Counting classes are at least as hard as the polynomial-time hierarchy. *SIAM Journal on Computing,* 20, pp. 316-328, 1992.