SOBRE EL CONCEPTO DE RECURSIÓN Y SUS USOS

Sergio Mota
Universidad Autónoma de Madrid, España

Resumen

La noción de recursión se emplea en diferentes sentidos, adquiriendo una variedad de significados. Así, en unos casos se usa para caracterizar una regla esencial que constituye un modo de definición en un sistema. Este sentido tiene su origen en la Lógica Matemática y la Teoría de la Computabilidad. En otros casos se aplica para indicar la organización interna de una estructura, tal como sucede en la Ciencia Cognitiva y la Ciencia de la Computación, al tiempo que también se emplea en el sentido anterior en estas mismas disciplinas. El objetivo de este trabajo es mostrar en cada caso las diferencias en el uso de ambos sentidos.

Palabras clave: Recursión; regla gramatical; estructura; auto-inclusión; auto-referencia

Recibido: diciembre 17 de 2014 - Aprobado: abril 4 de 2015

Praxis Filosófica Nueva serie, No. 40, enero-junio 2015: 153 - 181

ISSN (I): 0120-4688 / ISSN (D): 2389-9387

On the concept of recursion and its uses

Abstract

The notion of recursion is commonly used in different ways, thus yielding a variety of meanings. In some cases it is used to characterize an essential rule in a system, which constitutes a mode of definition. This sense has its origin in Mathematical Logic and Computability Theory. In other cases it is applied to identify the internal configuration of a structure, as in Cognitive and Computer Science, where it is also used in the former sense. The goal of this paper is to clarify the differences between both uses or senses.

Keywords: Recursion; Grammatical Rule; Structure; Self-embedding; Self-reference

Sergio Mota. Doctorando en la Universidad Autónoma de Madrid, España. Licenciado en Psicología y Máster en Crítica y Argumentación Filosófica por la misma Universidad. Sus áreas de investigación son: el papel de la recursión en la obra de Wittgenstein. La recursión en la Ciencia Cognitiva del Lenguaje. Revisión de la Biolingüística chomskiana. Filosofía de la Lógica y de la Matemática. Filosofía de la Ciencia y del Lenguaje. Filosofía de la Mente.

Dirección posta: Departamento de Psicología Básica, Universidad Autónoma de Madrid, Campus de Cantoblanco, CP: 28049 Madrid, España.

Dirección electrónica: sergio.mota.v@gmail.com

SOBRE EL CONCEPTO DE RECURSIÓN Y SUS USOS*

Sergio Mota
Universidad Autónoma de Madrid, España

El sentido original de 'recursión'

El sentido original de recursión está claramente establecido dentro de la teoría de la computabilidad y de la lógica matemática (Soare, 1996), si bien dentro de estos dominios parece haberse empleado en dos sentidos diferentes. Uno de ellos se refiere a la definición por recursión (o definición recursiva) y el otro a la noción de computabilidad (y lo mismo vale para los adjetivos 'recursivo' y 'computable'; véase, por ejemplo, Soare, 2009).

El primero de los sentidos mencionados, el de la definición por recursión, hace referencia a un método usado para definir funciones y predicados. Su origen se remonta al siglo XIX y fue ya empleada por autores como Dedekind o Peano (ver Soare, 1996, 2009 para más detalles y referencias).

Así, una función (o predicado) está definida, en un sentido técnico, por recursión cuando para definirla para un argumento *y* hacemos uso de sus propios valores previamente computados para argumentos menores que *y*; pudiendo emplearse también funciones previamente definidas (Soare, 1996, 2009; véase también Gödel, 1931; Kleene, 1952; Cutland, 1980).[1]

* Mi gratitud a José Manuel Igoa y David J. Lobina por su lectura atenta y valiosos comentarios a una versión previa. Asimismo, mi agradecimiento a los revisores anónimos de *Praxis Filosófica*, cuyas observaciones han ayudado a mejorar el texto.

[1] Esta definición supera algunas imprecisiones en Mota (2013, 2014); aunque bien es cierto que en tales trabajos se sigue correctamente a Cutland (1980, p. 32): una función *f* es recursiva; esto es, está definida por recursión, si cada (nuevo) valor (de *f*) se especifica/define usando sus propios valores previamente definidos (computados), siguiendo, obviamente, la regla recursiva que corresponda.

Este es el sentido en el que Dedekind y Peano usaron dicho concepto y que constituye su significado original (Soare, 1996, pp. 286-287). En este sentido, Epstein y Carnielli (1989) indican que en su forma más simple, la definición recursiva de una función *f* sería como sigue (siendo *g* una función previamente definida): $f(0) = m$; $f(n+1) = g(f(n))$.[2]

Por otro lado, conviene no confundir la definición inductiva, la cual sirve para definir términos como 'número natural', 'objeto', etc., con una definición por recursión (o 'recursiva'). Sólo la segunda, y no la primera, es la empleada en la Lógica Matemática y la Teoría de la Computabilidad para definir funciones y/o predicados (Kleene, 1952, p. 217). Por consiguiente, definir un término mediante una definición inductiva no convierte al objeto definido en recursivo (ver infra). La recursión es una propiedad de la definición, de la regla (Mota, 2013, 2014).

Así, desde la lógica matemática, la matemática y la ciencia de la computación, podemos distinguir entre las definiciones inductivas y co-inductivas (que proporcionan los principios duales a los de las primeras).[3] Las primeras proporcionan, o están en la base de, dos principios diferentes que, *a priori*, no hay que confundir, los principios de inducción (que permite probar ciertas propiedades internas de los objetos tratados, por ejemplo de los números naturales o de las fórmulas proposicionales), y de recursión (que, como se verá, garantiza la buena definición de las funciones recursivas), las segundas proporcionan los principios de co-inducción (que también permiten probar ciertas propiedades internas de los objetos tratados) y de co-recursión (que garantizan la buena definición de funciones a través de procesos). Desde un punto de vista formal, estas definiciones se pueden contextualizar de varias maneras: en la Teoría de Conjuntos, en la Teoría de categorías, entre otras. Sin embargo, no es el objetivo de este trabajo entrar en cada una de estas aplicaciones.

Para una adecuada contextualización de la noción de recursión, creo conveniente precisar la distinción entre 'definición inductiva' y 'definición recursiva' más detalladamente. Así, en el marco particular de una teoría axiomática de conjuntos, por ejemplo, Zermelo-Fraenkel (ZF) con el axioma de elección, se puede formalizar la noción de Conjunto Inductivo. Un conjunto I se dice inductivo si (a) el conjunto vacío '∅' pertenece a I y

[2] Así, las funciones no son intrínsecamente recursivas, sino que lo son en virtud de cómo se definen. Como ya he señalado en otro trabajo (Mota, 2014, p. 22) una definición recursiva tal es una regla gramatical en el sentido establecido por Wittgenstein (*Cf.* Mota, 2013, nota 3).

[3] Agradezco a los revisores anónimos de *Praxis Filosófica* sus comentarios y sugerencias en este punto, los cuales e intentado seguir con toda fidelidad y precisión.

(b) siempre que un conjunto X pertenece a I, entonces también el sucesor S(X), obtenido de la unión de X con el conjunto que tiene como único elemento a X, esto es, $S(X) = X \cup \{X\}$. Así, pueden definirse cómodamente el conjunto de los números naturales. Esta definición parece estar relacionada con la concepción iterativa de un conjunto (Boolos, 1971). La definición inductiva arriba presentada proporciona de inmediato el llamado Principio de Inducción (matemática), el cual permite mostrar propiedades internas de los objetos así construidos (Boolos, *Op. cit.*, pp. 223-224). Así, hay ciertamente una analogía entre la definición inductiva de los números naturales con la definición inductiva aplicada a la formación de conjuntos (op.cit., p. 223), pero no hay que confundir la definición inductiva con el Principio de Recursión, que garantiza la buena definición de distintas operaciones como la suma o el producto (Hrbacek y Jech, 1999). Siguiendo a Hrbacek y Jech (1999, p. 46 y ss.), el Principio de Recursión permite definir funciones sobre N. Por ejemplo, una de tales funciones es la adición (ver nota 6). Una interesante distinción que estos autores hacen (*Ibid*.) es entre la definición recursiva de tales funciones y la prueba por inducción que sirve para probar ciertas propiedades de la adición, como la conmutativa. En este caso, para $m, n \in N$, se prueba que $m+n = n+m$, para después probar, mediante inducción sobre n, que $m+(n+1) = (n+1)+m$. finalmente se prueba, mediante inducción sobre m, que $(m+1)+(n+1) = (n+1)+(m+1)$.[4]

Por otro lado, una forma de expresar la inducción matemática es como sigue (véase Boolos, 1971 para otras versiones):[5]

$$(P)\,[P0\ \&\ (n)\,[Pn \rightarrow PSn] \rightarrow (n)Pn]$$

Las distintas clases de funciones recursivas hacen uso, por tanto, del Principio de Recursión, como muestra claramente Kleene (1943), mediante el esquema V. Tales clases diferentes de funciones así definidas son: las primitivas, en las que la inducción se aplica sobre una variable (véase Skolem, 1923; Gödel, 1931);[6]

[4] El principio de inducción matemática es, por tanto, diferente al de recursión. Por otro lado, una regla de sustitución como (a) 3+3 = ((2+2)+1)+1 es diferente a una regla recursiva como (b) 3+3 = (3+2)+1 = ((3+1)+1)+1. La segunda posibilita el paso recursivo de '3+3' a '(2+3)+1' y, sólo desde de ahí, a '((2+2)+1)+1'. Sin embargo, conviene advertir, una cosa es el paso desde '3+3' a '(2+3)+1' (permitido por la regla recursiva), o desde '(2+3)+1' a '((2+2)+1)+1' (también permitido, aunque cambiando el parámetro) y otra cosa es el paso desde '3+3' a '((2+2)+1)+1', el cual no es un paso recursivo directo; es decir, (a) no instancia directamente a '$a+(b+1) = (a+b)+1$'. Por otro lado, una prueba por inducción matemática también puede hacer uso de la iteración (véase Wittgenstein, 1974, 32).

[5] En palabras: si 0 tiene la propiedad *P* y si para todo n que tiene *P* su sucesor también, entonces todo *n* tiene *P*.

[6] Un ejemplo de función recursiva primitiva es el de la suma: $a+0 = a/a+1 = a'$ Def., [caso base]; $a+(b+1) = (a+b)+1$ Def. [paso recursivo].

las generales, en las que se aplica sobre, al menos, dos variables simultáneamente y que incluye a las primeras (véase Gödel, 1934);[7] y las parciales que incluyen a las anteriores como aquellas funciones parciales para las cuales está definido todo el conjunto de sus argumentos (esto es, como funciones recursivas totales) y que reciben su nombre precisamente porque una función no necesita estar definida para todas las n-tuplas de números naturales que toma como argumentos (Kleene, 1938, 1943, 1952).[8] Tal clase de funciones recursivas identifica correctamente la clase de las funciones computables (Kleene, 1952).[9]

Otros formalismos propuestos dentro de la teoría de la computabilidad y la lógica matemática también hacen uso de la definición recursiva. Tal es el caso del formalismo de Post (1921, 1943, 1944). Post formalizó un sistema capaz de generar todas las proposiciones de la lógica de enunciados (alias "tautologías", *Cf.* Wittgenstein, 6.1). Así, sea g un enunciado de la lógica proposicional y P una variable operacional aplicada a dicho enunciado, el sistema produce un enunciado g' que sustituye a g. Esto se expresa en su sistema de producción normal como sigue: $gP \rightarrow Pg$' (Post, 1943, p. 199). De esta manera, el método de decisión formulado por Post, similar al proporcionado por Wittgenstein (1922), permite decidir en un número finito de pasos si una fórmula es o no una tautología (esto es, una proposición

[7] Un ejemplo de función recursiva general es: $\varphi(0,y) = \psi(y)$ Def., $\varphi(x+1, 0) = \chi(x)$ Def., $\varphi(x+1, y+1) = \varphi(x, \varphi(x+1, y))$ Def. Aunque en '$\varphi(x, \varphi(x+1, y))$' las funciones interna y externa son denotadas por 'φ' ambas son funciones diferentes, de primer orden la interna de y de segundo orden la externa (ver infra).

[8] Una función parcial es: $f(x) = f(x\text{-}1)/2$. Esta función sólo está definida para los números naturales impares tomados como argumentos (*Cf.* Boolos y Jeffrey, 1974; para una definición más técnica véase Kleene, 1938, 1943, 1952).

[9] Como Soare (2009) y otros autores señalan (véase Sieg, 1997), Church cometió un error al identificar las funciones recursivas generales con las computables. Este error, por un lado, y la identificación extensional de las funciones recursivas parciales con la máquina de Turing (esto es, que ambos formalismos computan la (misma) clase de las funciones computables o calculables; véase Epstein y Carnielli, 1989), me llevaron a definir explícitamente la Tesis de Kleene y la Tesis de Turing-Kleene, que se pueden condensar en la siguiente formulación: *una función es computable si y sólo si es una función recursiva parcial (o está definida por el formalismo de Kleene)* –que hace referencia a la Tesis de Kleene– *o, de forma equivalente, si es computable/calculable por una máquina de Turing* –que junto con lo anterior constituye la Tesis de Turing-Kleene (Mota, 2013, 2014). La *Tesis de Turing-Kleene* bien podría substituir a la *Tesis de Church-Turing* dado que, como señalo, fue Kleene, y no Church, quien identificó correctamente la clase de las funciones computables (Mota, 2014). Por otro lado, se usa la Tesis de Church para hacer alusión a la equivalencia extensional de las funciones recursivas generales definidas por Gödel, el cálculo-λ de Church y las máquinas de Turing, todas ellas propuestas para caracterizar la noción de algoritmo (*Cf.* Soare, 1996).

de la lógica (proposicional)).[10] Su tesis, conocida como la Tesis de Post, establece que un conjunto no vacío (como el de las proposiciones de las lógica de enunciados) es efectivamente numerable si y sólo si es derivado de un sistema de producción (normal) [derivado de su sistema canónico (normal)] (Soare, 2009).[11] Un sistema tal hace uso de la definición por recursión, tal como Post (1943, p. 201) indica (él hace uso de la expresión definición por inducción pero se pueden intercambiar), y como he definido en otro trabajo (Mota, 2013; 2014, p. 31):

$P^{(0)}{}'(g) = g$ Def.,
$P^{(n)}{}'(g) = P'P^{(n-1)}{}'(g)$ Def.

Como señalé al comienzo de este apartado, el concepto de recursión se ha aplicado con cierta ambigüedad dentro de la teoría de la computabilidad. Así, tal y como señala Soare (1996, 2009), dicha noción se comenzó a aplicar para significar 'computabilidad'. Sin embargo, el término 'computabilidad' (o 'efectivamente computable') se refiere a una función para la que existe un procedimiento (efectivo) mecánico finito (esto es, un procedimiento generativo/computacional o algoritmo) por medio del cual es computada o calculada; esto es, si se ha definido para ella una secuencia de reglas con las que, dado un *input*, se obtuviera un valor o *output*, a través de una secuencia finita de operaciones. Desde 1996, Soare ha propuesto volver a emplear el concepto de recursión en su sentido original, distinguiéndolo del de computabilidad, un concepto relacionado pero no coextensivo, dado que hay formalismos que no proceden recursivamente, como la máquina de Turing (véase Turing, 1937; para una definición esquemática ver infra).

[10] El llamado 'problema de la decisión' (*Entscheidungsproblem*), formulado por Hilbert a comienzos del Siglo XX, consistía en establecer un procedimiento de decisión (*Entscheidungsverfahren*) que permitiera determinar o decidir, en un número finito de operaciones, si una expresión o fórmula bien formada es o no válida. Obviamente, esto se relaciona con su *programa finitista*, por medio del cual pretendía probar la consistencia de la aritmética. Dicho problema fue mostrado irresoluble por Gödel (1931), Church (1936), Turing (1937) y Post (1943).

[11] Post (1944) también se centró en los conjuntos recursivamente numerables, distinguiendo entre éstos y los conjuntos recursivos. Un conjunto S es recursivo si se puede establecer un método efectivo –un procedimiento mecánico finito– para determinar si, por ejemplo, un número entero positivo n pertenece o no al conjunto S. Un conjunto S es recesivamente numerable si existe un procedimiento mecánico efectivo para numerar efectivamente los elementos de S (adelanto aquí que tales procedimientos pueden ser las funciones recursivas primitivas, generales y parciales; véase infra para más detalles). Tal y como señala Soare (1996, 1999, 2009) 'recursivamente numerable' significa 'efectivamente numerable' y 'conjunto recursivo' significa 'conjunto efectivo'.

En todo caso, aunque la máquina de Turing proceda iterativamente, a diferencia de las funciones recursivas, esto no quiere decir que no se pueda establecer una regla recursiva que defina lo que haga. Así, tal y como he indicado en otro trabajo (Mota, 2014), haciendo uso del siguiente diagrama de estados (simplificado):

… 0 0 0 0 … (esta es la cinta simplificada)

a. La primera instrucción es: E0, 0, E1, 1: la máquina, estando en E0 escaneando '0', pasa al estado E1 substituyendo '0' por '1': … 1 0 0 0 … [E1 se define como $\tilde{O}$ sobre 0]
b. En este segundo paso, las instrucciones son: E1, 1, E2, >: la máquina, estando en E1 escaneando '1', pasa a E2 moviéndose a la derecha: *[*E2 se define como $\tilde{O}$ sobre E1 o como $\tilde{O}$ sobre $\tilde{O}(0)$*]*
c. En el tercer paso, las instrucciones son: E2, 0, E3, 1: la máquina, estando en E2, escaneando '0', pasa a E3 substituyendo '0' por '1': … 1 1 0 0 …*[*E3 se define como $\tilde{O}$ sobre E2, o como $\tilde{O}$ sobre $\tilde{O}'\tilde{O}(0)$*]*

(y así sucesivamente hasta que la máquina se detiene; omito aquí ese paso)

Es posible establecer el siguiente sistema de ecuaciones recursivas (Mota, 2014, pp. 31-32):

$\tilde{O}^{(0)}{}'(0) = 0$ Def.,
$\tilde{O}^{(n)}{}'(0) = \tilde{O}'\tilde{O}^{(n-1)}{}'(0)$ Def.

La primera ecuación expresa el estado E0 de la máquina, mientras que la segunda ecuación expresa la definición recursiva de los distintos términos/estados. Es importante señalar que '$\tilde{O}$' es una variable operacional cuyos valores son las distintas operaciones que realiza la máquina de Turing. Esto sólo indica que la serie generada por una máquina de Turing puede definirse recursivamente en el nivel de análisis relacionado con *qué* hace tal procedimiento. Sin embargo, respecto a *cómo* procede u opera es claro que lo hace iterativamente, esto es, pasando en sucesión de un estado a otro.

Así, aunque, por ejemplo, la función suma se defina recursivamente, no tiene que proceder (operar o implementarse, tanto de manera abstracta como en tiempo real) necesariamente de manera recursiva (*Cf.* Abelson et al., 1996). Una implementación recursiva instancia una definición recursiva, y un ejemplo del primer caso sería el siguiente: si se quiere computar 2+2, el procedimiento invoca un valor previamente computado para un argumento menor, esto es, se computaría (2+1)+1; con lo que obtenemos '2+2 = (2+1)+1 = ((2+0)+1)+1'; en cambio, una implementación iterativa de '2+2' se puede definir del siguiente modo: primero se obtiene el sucesor de 1 (1+1) y al

(último) resultado (2) se le añade 1, y al resultado (3) se le añade 1, hasta que se alcanza el valor '4' (lo que se puede expresar como 2+2 = 1+1+1+1). Mediante tal implementación iterativa, una operación genera sucesivamente valores desde un argumento a otro sin usar valores previamente calculados para argumentos menores.

En cualquier caso, teniendo en cuenta que un sistema formal como los sistemas que acabo de mencionar es una serie de formas, esto es, cada uno de sus términos están ordenados por relaciones internas (Wittgenstein, 1975, § 154) y que una serie de formas queda definida por [*a, x, O'x*] (Wittgenstein, 1922, 5. 2522)[12] o por $O^{(0)}{}'(a) = a$ Def.; $O^{(n)}{}'(a) = O'O^{(n-1)}{}'(a)$ Def. (Mota, 2013, 2014) podemos definir la forma general de un procedimiento mecánico finito como [$\tilde{\sigma}$, $\tilde{\varepsilon}$, $\tilde{O}(\tilde{\varepsilon})$] (Mota, 2013, §2; 2014, p. 30) o como $\tilde{O}^{(0)}{}'(\tilde{\varepsilon}) = \tilde{\varepsilon}$ Def.; $\tilde{O}^{(n)}{}'(\tilde{\varepsilon}) = \tilde{O}'\tilde{O}^{(n-1)}{}'(\tilde{\varepsilon})$ Def. (Mota, *Ibid*.). Así, bajo esta formulación, que expresa una variable, caen los siguientes ejemplos:

a. En el caso de una función recursiva,
   $\tilde{\varepsilon}$, $\tilde{O}\tilde{\varepsilon}$, $\tilde{O}\tilde{O}\tilde{\varepsilon}$,…
   2+2, 2+2+1, 2+2+1+1,…
   (2+2), (2+3), (2+4),…
   se puede instanciar como '2+4 = (2+3)+1'.

b. En el caso del formalismo de Turing,[13]
   $\tilde{\varepsilon}$, $\tilde{O}\tilde{\varepsilon}$, $\tilde{O}\tilde{O}\tilde{\varepsilon}$,…
   $\tilde{\varepsilon}$, $\tilde{O}\tilde{\varepsilon}$, $\tilde{O}\tilde{O}\tilde{\varepsilon}$,…
   E0, $\tilde{O}(E_0)$, $\tilde{O}\tilde{O}(E_0)$,…

que en el caso de la generación de los números naturales se puede escribir:

[12] En 5.2522, Wittgenstein explica en qué consiste esta notación. Así, "[e]l primer término…es el comienzo de la serie de formas, el segundo es la forma de un término x cualquiera de la serie y el tercero la forma de aquel término de la serie que sigue inmediatamente a *x*".

[13] Bien es cierto que el formalismo de Turing es considerado como un modelo de computabilidad, pero esto no niega que se pueda concebir como una adecuada formulación de un sistema formal o, también, de un procedimiento mecánico finito (véase Gödel, 1964, pp. 71-72). Por otro lado, los sistemas formales considerados aquí pueden entenderse como objetos matemáticos formales (*Cf.* Soare, 1996), considerados, bajo la perspectiva del formalismo; esto es, bajo una concepción no-descriptivista de la matemática. En otro sitios (Mota, 2013, 2014), y siguiendo la concepción formalista y wittgensteiniana de la matemática (y de la lógica), he considerado que tales objetos matemáticos o sistemas formales no son sino construcciones o invenciones constituidas por reglas gramaticales, las cuales no son descripciones de ninguna realidad (empírica o platónica).

E_0; Sub0/1' Esc0' (E_0); >' Esc1' (Sub0/1' Esc0' (E_0)),[14] … que significa:

E_0, E1 (=Sub0/1' Esc0' (E_0)), E_2 (= >' Esc1' (Sub0/1' Esc0' (E_0))),…y así sucesivamente.[15]

Por tanto, cada estado/configuración o término se puede definir como sigue:

$\tilde{O}^{(n)\prime}(\tilde{\varepsilon}) = \tilde{O}\,'\tilde{O}^{(n-1)\prime}(\tilde{\varepsilon})$.

Así, la expresión 'Sub0/1' Esc0'>' Esc1' (Sub0/1' Esc0' (E_0))' muestra que, estando en E0, se aplican las operaciones 'Esc0' y 'Sub0/1', lo que permite calcular '1', pasando así al estado E1. Mediante la aplicación de las operaciones 'Esc1', '>', 'Esc0' y 'Sub0/1' la máquina calcula '2' y así sucesivamente. La recursión caracteriza la definición de los *n* estados/configuraciones, términos o formas (v.gr., '$\tilde{O}^{(n)\prime}(\tilde{\varepsilon})$') efectivamente computados por una máquina de Turing. Esto caracteriza *qué* hace una máquina de Turing (lo que se puede aplicar al resto de formalismos), esto es, generar recursivamente una serie de términos, pero *cómo* lo hace queda descrito por medio de la iteración, esto es, por medio de la repetición sucesiva de operaciones sobre el último resultado calculado (*Cf.* Mota, 2013, 2014).

El análisis que acabo de ofrecer me permite hacer la siguiente observación. Tal como indica Soare (2009), puede establecerse una diferencia intensional entre *sistemas generacionales*, que proporcionan un procedimiento o algoritmo para listar un conjunto de, digamos, números naturales, como los sistemas de producción de Post, y *sistemas computacionales*, los cuales computan funciones, como las funciones recursivas primitivas, generales, parciales o las funciones computables por una máquina de Turing. Es indudable que tales diferencias intensionales no son esenciales, lo cual no quiere decir que sean irrelevantes, para la formulación de la forma general de un procedimiento mecánico finito. De hecho, todos ellos caen bajo esa misma forma general.[16] No obstante, podemos encontrar *sistemas computacionales* en el sentido que acabo de señalar que listen un conjunto de números naturales; esto es, que sean *sistemas generacionales*. De tal manera que una función recursiva también proporciona un procedimiento generativo (o generacional), por medio del

[14] 'EscX' significa 'escanear X'; 'SubX/Y' significa 'sustituir X por Y'; '>' significa 'mover a la derecha'.

[15] Por ejemplo, E1 es el estado de la máquina obtenido estando previamente en E0, escaneando 0, y substituyendo 0 por 1.

[16] Tal forma general muestra lo que todos ellos tienen en común como procedimientos mecánicos finitos, algoritmos o sistemas matemáticos formales: un conjunto de términos que generan (u operan sobre) otros términos o secuencia de ellos mediante la aplicación de operaciones siguiendo un conjunto de reglas (*Cf.* Gödel, 1934).

cual se puede listar (o generar) un conjunto de números naturales (*Cf.* Post, 1944).[17] Pero también es posible encontrar el caso inverso, es decir, un sistema de producción *gP→Pg'* donde g puede estar por una cadena de '1s' y 'bs', que a su vez están por un número natural n, cadena a la que al aplicar una variable operacional *P* el sistema produce otra secuencia de '1s' y '*bs*' tal que genera g'; esto es, otra secuencia. Esto se puede representar como *b*1*bP→P*1*bb*1 (Post, 1944, p. 307). De este modo, cada número natural *n* se puede definir como una función de *n* aplicaciones de *P* sobre *g*: $P^{(n)}$'(*g*) = P'$P^{(n-1)}$'(*g*) Def.

Otro ejemplo de esto puede mostrarse con respecto al cálculo-λ definido por Church (1932, 1936). Así, Church, (1936, p. 91), define los números naturales de la siguiente manera:[18]

$1 \rightarrow \lambda ab.\ a(b)$
$2 \rightarrow \lambda ab.\ a(a(b))$
$3 \rightarrow \lambda ab.\ a(a(a(b)))$

Tal formulación puede convertirse en términos de funciones (*Cf.* Frascolla, 1994; Marion, 1995, 1998; Odifreddi, 2001). Así, como ya he señalado en otro lugares (Mota, 2013, 2014), donde también subrayé el parecido de esta formulación con la de Wittgenstein unos años antes (1922), podemos transformar ese sistema en el sistema de ecuaciones que ofrezco a continuación, en donde la definición recursiva para ambos procedimientos es como sigue: Ω^0'$x = x$ (en el caso del formalismo de Church: $F^{(0)}(X) = X$) [*el caso base*]; $\Omega^{(v+1)}$'$x = \Omega$'Ω^v'x (en el caso del formalismo de Church: $F^{(n+1)}(X) = F(F^{(n)}(X))$) [*el paso recursivo*].

En suma, la recursión, que se diferencia de la iteración conceptualmente aunque ambas puedan coexistir en diferentes dominios de análisis, es una propiedad de una definición o una regla que indica que para definir un valor se hace uso de un valor o valores previamente computado/s para un argumento (o varios) menor/es (*Cf.* Cutland, 1980; Odifreddi, 2001). Esto es así

[17] De hecho, el propio Post (1944, pp. 307-308) reconoce que el formalismo de las funciones recursivas parciales amplió el concepto de conjunto recursivamente numerable. En todo caso, en ese mismo trabajo (p. 306), indica que un conjunto de enteros positivos (los números naturales) es recursivamente numerable si hay una función recursiva $f(x)$ cuyos valores, para enteros positivos como valores de x, constituye tal conjunto (esto es, el rango de $f(x)$). Si tal función recursiva permite generar los números naturales en su orden natural, entonces también se ha establecido un procedimiento para determinar si un entero natural pertenece o no a tal conjunto; de ahí se desprende, de acuerdo con las definiciones dadas más arriba, que tal conjunto es, también, recursivo (*Cf.* Post, 1944, p. 311).

[18] Estas expresiones tienen la forma $\alpha \rightarrow A$, la cual ha de leerse como "α representa a/está por A" (Church, 1936, p.91). En este sentido α puede 'rescribirse' como A.

independientemente de que la función se defina para todos los argumentos, pues como muestra Kleene (1943), todas las clases de funciones recursivas hacen uso de la definición por recursión, expresada por medio del esquema V (*Cf.* Soare, 1996, p. 287). En este punto se equivoca Tomalin (2007, 2011), cuando trata de precisar la noción de recursión dentro de la teoría sintáctica, pues él apunta que la noción de recursión se define de manera distinta en cada clase de funciones recursivas. El principio de recursión, además de garantizar la buena definición de las funciones recursivas, permite que manejemos un único uso de 'recursión', congruente, por tanto, con la definición dada por Soare e indicada unas líneas más arriba.

A continuación mostraré cómo se emplea la noción de recursión en la Ciencia Cognitiva, para después mostrar los paralelismos y las diferencias en el tratamiento que se hace de este concepto en la Ciencia de la Computación.

Recursión Vs. auto-inclusión en la Ciencia Cognitiva

En este apartado me propongo analizar conceptualmente las relaciones entre 'recursión' y 'auto-inclusión', empezando por la noción de 'recursión' aplicada a las estructuras, para después analizar la relación con los mecanismos recursivos empleados para generar tales estructuras.

En términos generales, se entiende que una estructura recursiva es aquella que contiene un constituyente A (o estructura) dentro de otro constituyente B (o estructura) del mismo tipo (véase por ejemplo Pinker y Jackendoff, 2005; Jackendoff y Pinker, 2005; Everett, 2005, 2009; Corballis, 2007, 2011; Moro, 2008; Karlsson, 2010; Kinsella, 2010; Zwart, 2011; Jackendoff, 2011; Lobina, 2011, 2014a; 2014b; Arsenijević y Hinzen, 2010, 2012).

En esta definición entran diferentes tipos de estructuras, ya sea en un nivel de descripción concreto, como las que contienen dos sintagmas nominales o dos oraciones de relativo (Karlsson, 2010, p. 51) o en un nivel mayor de abstracción, como los esquemas del tipo [especificador-[núcleo-complemento] incluidos dentro de otros esquemas idénticos, dado que tanto el especificador como el complemento pueden incluir tales esquemas: […*Núcleo*…-[*Núcleo*-[…*Núcleo*…]]] (Moro, 2008).

Luuk y Luuk (2011) definen las estructuras con auto-inclusión como acabo de hacer, pero ellos se muestran más críticos con la aplicación de la noción de recursión a tales estructuras, como parece que Chomsky (1965) hiciera al definir las estructuras con auto-inclusión. Así, una estructura con auto-inclusión es aquella en la que el sintagma (o estructura) A esta auto-incluido en B si A está anidado en B y, además, A es un sintagma (o una estructura) del mismo tipo que B.

Desde una perspectiva histórica, tales estructuras se han venido relacionando con reglas de rescritura del tipo *S→aSb*. Sin embargo, como muy bien han señalado Luuk y Luuk (2011), una regla tal puede generar tanto secuencias que exhiben auto-inclusión como secuencias que no la exhiben. En concreto, estos autores han puesto de relieve que una regla como AB→AABB, que genera secuencias como *aabb*, *aaabbb*,…, no entraña necesariamente auto-inclusión, y a la inversa, secuencias como estas, incluso admitiendo que presentan auto-inclusión en su estructura interna, no tienen por qué haberse generado necesariamente mediante reglas recursivas. En esta misma línea, Lobina (2014a, 2014b) ha mostrado que una secuencia de reglas como *A*→aB, B→*aC*, *C*→*bD*, *D*→*b* genera secuencias como aabb, las cuales, como en el caso anterior, pueden o no exhibir auto-inclusión.

Lo que quiero subrayar aquí es que aunque AB→AABB sea una regla recursiva, de acuerdo con la definición original del término expuesta en el apartado anterior, la secuencia *aaabbb*, resultado de su aplicación, no puede juzgarse como recursiva o con auto-inclusión. En otras palabras, las estructuras subyacentes a las secuencias de este tipo pueden ser o no "recursivas" con independencia de la propiedad recursiva de la regla.

Esta falta de precisión conceptual se muestra de forma palmaria en Jackendoff (2011), quien distingue entre dos tipos de recursión, la *recursión formal* y la *recursión estructural*. Jackendoff aplica la primera a las reglas y afirma que un conjunto de reglas es recursivo si éstas pueden aplicarse a su propio resultado un número ilimitado de veces a partir de un conjunto de ítems primitivos o no definidos (*Op. cit.*, p. 591). Esta definición, sin embargo, es incorrecta, ya que no distingue la recursión de la iteración. Por otra parte, define la recursión estructural en los términos arriba señalados, afirmando que una estructura recursiva es aquella que incluye constituyentes dentro de otros (del mismo tipo) con una profundidad ilimitada (*Op. cit.*, p. 592). Después de establecer tal distinción, Jackendoff sostiene que de estos dos usos del término, el de recursión estructural es más útil para los propósitos de la Ciencia Cognitiva. Así, en el lenguaje natural, nos dice Jackendoff, tenemos que establecer primero su dominio de recursión estructural y, si podemos hacerlo, entonces podemos concluir que las reglas que definen el dominio son formalmente recursivas. En otras palabras, según Jackendoff, podemos inferir la recursión formal a partir de la recursión estructural. O dicho de una manera ligeramente diferente, que la *única* evidencia de la recursión formal es la recursión estructural.

Ciertamente, el análisis que presenta Lobina (2014b) en términos de reglas no recursivas (ver supra), las cuales pueden generar secuencias tanto sin como con auto-inclusión, es válido para poner en duda la aseveración

de Jackendoff. Sin embargo, también se puede desmontar la aseveración de Jackendoff sin acudir al argumento de las reglas no recursivas. De hecho, tal y como he mostrado en otro trabajo (Mota, 2014), se puede definir una regla recursiva sin necesidad de establecer primero un dominio de auto-inclusión. Así, una regla recursiva como '$f(x) = s(f(x\text{-}1))$', que puede perfectamente definir una función gramatical, expresa una definición recursiva en la cual no hay rastro de auto-inclusión. Esta regla define valores haciendo uso de valores previamente computados para argumentos menores, estableciendo una relación de intercambio o substitución entre expresiones, pero nunca de auto-inclusión (Mota, 2013, 2014). Lo recursivo es, por tanto, la definición, la regla que subyace a dicha ecuación matemática y la constituye, pero ninguna de las expresiones por separado lo es (Mota, 2014, p. 41).

Un asunto no menos importante es el hecho de que una regla recursiva no tiene por qué ser necesariamente una regla de rescritura. *Merge*, el actual procedimiento generativo/computacional que subyace a la facultad del lenguaje dentro del *Programa Minimista* (Chomsky, 1995; Hauser, Chomsky y Fitch, 2002), puede definirse en términos de un sistema de ecuaciones recursivas y no en términos de reglas de rescritura y no por ello deja de ser recursivo.

En definitiva, que una regla recursiva no pueda caracterizarse mediante la propiedad de la auto-inclusión dado que la relación entre expresiones es de substitución o intercambio y no de (auto-)inclusión (Mota, 2013, 2014), muestra claramente que la auto-inclusión no puede ser razón a partir de la cual inferir o deducir la propiedad de la recursión. Por ello, si usamos 'recursión' en su sentido original, la auto-inclusión no justifica la propiedad de la recursión, pues ambas propiedades son independientes, y la razón por la que se predica recursión (en tal sentido original) no es, en ningún caso, la auto-inclusión (Mota, 2014).

La misma línea argumental puede seguirse para Merge. Dentro del actual *Programa Minimista*, Chomsky (1995, 2007a) establece que un lenguaje-I es un procedimiento generativo/computacional que genera recursivamente una ordenación (o serie) infinita de expresiones jerárquicamente estructuradas mediante la operación denominada "*Merge* (o ensamble)".[19] Lo que hace un procedimiento mecánico finito de este tipo es construir recursivamente objetos sintácticos (Chomsky, 1995, p. 226). Como es bien sabido, las

[19] Para Chomsky (1959, 1995, 2005, 2007, 2008, 2010, 2011, 2012), el Lenguaje-I es concebido como un procedimiento generativo o sistema de cómputo capaz de generar una cantidad potencialmente infinita de expresiones jerárquicas internas. De este modo, la teoría de la gramática, que comprende el estudio del lenguaje así entendido, es el estudio de una clase especial de funciones recursivas dentro de la teoría de la computación (o computabilidad) (Chomsky, 1959, 2012).

expresiones u objetos sintácticos generados por *Merge* son interpretados por dos sistemas, a saber: el sensorio-motor y el conceptual-intencional (véase para más detalles Hauser Chomsky y Fitch, 2002; Chomsky, 2006, 2007a, 2007b, 2008, 2010, 2011).

Tal y como he defendido en diferentes lugares (Mota, 2013, 2014), Chomsky ha empleado correctamente la recursión como propiedad que define su procedimiento generativo/computacional. En esos mismos trabajos defino recursivamente de forma clara tal procedimiento; lo cual muestra qué hace *Merge*, a saber, generar recursivamente objetos sintácticos (Chomsky, 1995). Así, la forma general de *Merge* se puede establecer de la siguiente manera: [*N*, *Os*, $M(O_s)$]. *N* hace referencia a las piezas léxicas (o ítems léxicos) y objetos sintácticos que configuran la numeración (o repertorio seleccionado de objetos que entran en una derivación sintáctica), O_s hace referencia a un objeto sintáctico cualquiera de la serie generada (que puede tomar n-valores: si $n = 1$, entonces $Os = X$; si $n = 2$, entonces $Os = X, Y$, y así sucesivamente) y $M(O_s)$ hace referencia a un objeto sintáctico nuevo generado a partir de O_s mediante la aplicación de $M(\ )$ –esto es, *Merge*. Este procedimiento genera/define recursivamente objetos sintácticos, independientemente de cuál sea su organización interna a efectos de la auto-inclusión de constituyentes (o esquemas) (Mota, 2013, nota 2; 2014, p. 38).

Lo que hace *Merge* (entendida aquí como una operación binaria) es tomar dos ítems léxicos, X,Y, para crear con esa unión un nuevo objeto sintáctico Z (X,Y = {X,Y} = Z). Mediante otra aplicación, Merge genera un nuevo objeto sintáctico, formado a partir de un objeto previamente creado. Así, y de manera esquemática, vemos que lo que Merge hace queda expresado mediante la siguiente serie: O_s, $M'O_s$, $M'M'O_s$… Esto puede definirse recursivamente de manera formal como sigue (Mota, 2013, nota 2; 2014, p. 38):

$M^{(0)\prime}(O_s) = O_s$, Def,
$M^{(n)\prime}(O_s) = M'M^{(n-1)\prime}(O_s)$ Def.

Pero el procedimiento empleado por Chomsky y definido por medio de este sistema de ecuaciones recursivas, no solo genera expresiones sin auto-inclusión de constituyentes (como *Juan canta muchas canciones*), sino que también genera estructuras con auto-inclusión como *el ratón que el gato que el perro perseguía mordió corría*, la cual exhibe inclusión central y, además, auto-inclusión, dado que una cláusula de relativo se incluye dentro de otra del mismo tipo. Antes de continuar, conviene advertir que según lo que acabo de exponer, un dominio sin auto-inclusión también puede ser definido por recursión y, por tanto, como ya he mostrado, la

auto-inclusión no justifica la recursión y, por ello, ambos conceptos no deberían identificarse/equiparase.

Dicho esto, la descripción esquemática de cómo genera *Merge* ambas oraciones es como sigue:

Juan canta muchas canciones
M_{Juan}'M_{canta}'$Os_{\{muchas,\ canciones\}}$

el ratón que el gato que el perro perseguía mordió corría
M_{el}'$M_{ratón}$'M_{que}'M_{el}'M_{gato}'M_{que}'$M_{\{\{el,\ perro\},\ perseguía\}}$'$M_{mordió}$'$O_{s\ corría}$

Por otra parte, Chomsky (2007a, p. 6; 2008, p. 139) ha insistido en que el modo de operar (cómo procede, o puede implementarse de manera abstracta) de este procedimiento generativo consiste en aplicar iterativamente la operación *Merge*. Así, la aplicación iterativa queda descrita mediante la siguiente serie, que muestra, en cada paso, las n veces que Merge se aplica de forma sucesiva, lo cual puede generar todo tipo de expresiones (con auto-inclusión o no) del lenguaje natural: $M^{(0)}$, $M^{(1)}$, $M^{(2)}$,…, $M^{(n)}$. Vemos, una vez más, que las estructuras con auto-inclusión no justifican la propiedad de la recursión.

Así, en el caso de la oración *Juan canta muchas canciones* (una oración que no presenta constituyentes dentro de constituyentes del mismo tipo y, por tanto, sin auto-inclusión a tal nivel), la aplicación iterativa es como sigue:

Juan canta muchas canciones
M_{Juan}'M_{canta}'$Os_{\{muchas,\ canciones\}}$

En donde la secuencia completa es:
{A,B} = C; {C,D}= E; {E,F} = G
Que equivale a la regla $M'M'M'(O_s) = M^{(3)}'(Os)$

Esto indica que la operación *Merge* se ha aplicado tres veces sucesivamente (iterativamente) sobre su último resultado generado (Mota, 2013, §1; 2014, p. 27).

Por su parte, en el caso de la oración *el ratón que el gato que el perro perseguía mordió corría* (la cual presenta auto-inclusión) la generación iterativa sería como sigue:

El ratón que el gato que el perro perseguía mordió corría
M_{el}'$M_{ratón}$'M_{que}'M_{el}'M_{gato}'M_{que}'$M_{\{\{el,\ perro\},\ perseguía\}}$'$M_{mordió}$'$O_{s\ corría}$

Cuya secuencia completa es:
{A,B} = C; {C,D} = E; {E,F} = G; {G,H} = I; {I,J} = K; {K,L} = M; {M,N} = P; {P,Q} = R; {R,S} = T
Esto equivale a $M'M'M'M'M'M'M'M'M'(Os) = M^{(9)}'(O_s)$

En este caso, *Merge* se aplica nueve veces de manera sucesiva sobre el último resultado producido, generando iterativamente una expresión jerárquicamente estructurada como la empleada en el ejemplo.

En todos estos ejemplos, es importante tener presente que la recursión es una propiedad de la regla; pues no debemos obviar el hecho de que una definición recursiva como '$M(n)'(O_s) = M'M^{(n-1)}'(O_s)$' es una regla.

En todo este análisis me he ceñido al uso original de 'recursión', como una propiedad constituida por una regla y diferenciándola de la noción de auto-inclusión aplicada a estructuras. Tal y como se desprende de mi análisis, sólo la regla, pero no sus valores o resultados, es recursiva (*Cf.* Zwart, 2011 para otras interpretaciones).

Los resultados o las estructuras poseen sus propiedades independientemente de tales reglas. De hecho, las propias definiciones no dejan duda en este punto. Por repetirlo una vez más, una regla es recursiva si para computar un valor hace uso de valores previamente computados para argumentos menores (ver supra). Por el contrario, una estructura de datos con un componente X dentro de otro X del mismo tipo es lo que se conoce como estructura con auto-inclusión (Arsenijević y Hinzen, 2012), cuyas propiedades son independientes de las reglas recursivas (véase por ejemplo Luuk y Luuk, 2011; Mota, 2013, 2014; Lobina, 2014a, 2014b). A este respecto, parece que se ha establecido una convención según la cual se ha de llamar 'recursión' a la auto-inclusión.[20] Esta convención sobre la auto-inclusión implica (I) el uso de la auto-inclusión para caracterizar las reglas y (II) el uso de la recursión para caracterizar a las estructuras (advierto al lector de que este punto también abarcará el análisis efectuado en la sección siguiente).

En el primer caso, conviene señalar que no es una aseveración adecuada decir que lo que hacen las reglas de rescritura puede caracterizarse mediante la auto-inclusión; como parece indicar Fitch (2010, p. 79), cuando señala que una regla recursiva es una que tiene la propiedad de la auto-inclusión. Veamos varios argumentos que critican esta idea.

Primero, lo que sencillamente hacen es definir/generar una nueva descripción estructural desde una previamente generada (Post, 1921). En este

[20] Ya Soare (1996, 1999, 2009) advirtió que en la Teoría de la Computabilidad se había consolidado una convención que implicaba usar recursión para significar lo mismo que computabilidad (ver supra) y, por tanto, emergió un período de confusión conceptual. Tal Convención de la Recursión (Recursion Convention) implicaba lo siguiente: La 'Recursion Convention' hace referencia a: (1) el uso de términos del formalismo de las funciones recursivas generales para describir resultados (i.e., como 'recursivamente enumerable'), (2) el uso del término 'Tesis de Church' para referirse a un conjunto de diferentes tesis, (3) la denominación de la materia con el lenguaje de la recursión.

sentido, Post no parece especificar en sus obras las reglas con la propiedad de la auto-inclusión, por lo que no es adecuado relacionarlas con dicho término, tal como indico en otro trabajo (Mota, 2014). Así, para generar Σ, ε, εε, εεε, equivalente a Σ, $R(\Sigma)$, $R(R(\Sigma))$, $R(R(R(\Sigma)))$, se puede aplicar una regla como εΣ→εεΣ, lo cual sólo indica que lo que hace tal regla es rescribir las expresiones de lado izquierdo como las del lado derecho; lo que puede formularse como εεΣ = $R'(\varepsilon\Sigma)$ (Mota, 2014, p. 40). Esto último, al igual que las reglas del tipo '$a+b' = (a+b)+1$' sólo muestran que un valor se computa haciendo uso de valores previamente computados y que un valor se substituye/reemplaza por otro, por lo que la noción de auto-inclusión no es adecuada para caracterizar las reglas del cálculo. Esto es igualmente válido para una regla como AB→AABB.

Por su parte, Lobina (2014a, 2014b) señala que Fitch (2010) más que interesarse por reglas recursivas, lo hace por las estructuras con auto-inclusión, dado que entiende que un indicador empírico de la noción de recursión es la interpretación adecuada de una oración con auto-inclusión. Ciertamente, esta cuestión tiene que ver más con una noción puramente semántica y no tanto con una noción sintáctica (Lobina, 2014b, p. 66).[21]

El otro punto de lo que he llamado la *convención sobre la auto-inclusión* tiene que ver con *el uso de 'recursión' para caracterizar a las estructuras* en las que un X contiene o está dentro de otro X del mismo tipo (esto es, en términos generales, una estructura con auto-inclusión, *Cf.* Karlsson, 2010; Arsenijević y Hinzen, 2012; sea X un sintagma, un esquema o una estructura en árbol). La noción original de recursión, surgida en la Lógica Matemática y la Teoría de la Computabilidad hace referencia a una definición por medio de la cual se define un nuevo valor para un nuevo argumento usando un valor previamente definido/computado/generado para un argumento menor. En otros dominios, como en la Ciencia de la Computación, se definen procedimientos y algoritmos recursivamente (lo cual deriva de lo dicho anteriormente) pero también aplican la noción de recursión a una configuración de datos; esto es, a la configuración/organización interna de una expresión como, por ejemplo, '$(a+b)+c$' (ver infra y también Wirth, 1986 para otros ejemplos). En el próximo apartado examinaré este aspecto más detenidamente dentro de la Ciencia de la Computación, una vez analizado el fenómeno en el dominio de la Ciencia Cognitiva.

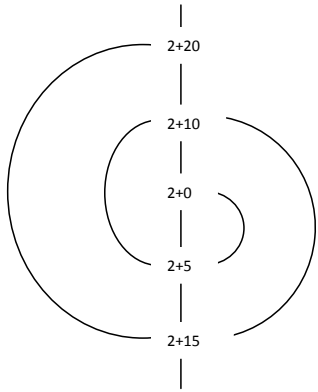
Recursión Vs. Auto-inclusión en la Ciencia de la Computación

[21] Por lo que Fitch (2010, pp. 80-81) parece interesarse es por la manera en la que se asigna un determinado significado a una señal o secuencia particular (y viceversa).

Como he señalado anteriormente, una estructura se considera recursiva si y sólo si un componente X contiene/incluye/está dentro de otro componente X del mismo tipo, siendo una instancia de tal definición las estructuras con auto-inclusión tratadas en el apartado precedente y las que se tratarán en este.

Quisiera advertir que una diferencia entre esta concepción de la recursión y la derivada de una *definición por recursión* (*o definición recursiva)* es que sólo en la primera pero no en la segunda se da una relación de 'estar dentro de', esto es, una estructura recursiva necesariamente tiene que tener un X que contiene o incluye a otro X del mismo tipo.

Creo que esta cuestión la expresó Wittgenstein de una manera meridianamente clara, contribuyendo de manera notable al desarrollo de la lógica matemática y la teoría de la computabilidad (véase, Wittgenstein, 1974, 1975, 1978; Wrigley, 1977; Shanker, 1987; Frascolla, 1994; Marion, 1995, 1998, 2009; Rodych, 1997, 1999a, 1999b, 2003; Oddifreddi, 2001; Mota, 2013, 2014). Como Wittgenstein señaló en las *Observaciones Filosóficas* (1975, §163) con absoluta precisión, una definición recursiva (por ejemplo 'a+0 = a'; 'a+(b+1) = (a+b)+1') es una regla fundamental del sistema que nos indica cómo proceder y, como tal, no se puede aseverar o negar, es decir, no son descripciones y, por ello, carece de valor de verdad. Abundando en esta misma idea, Wittgenstein nos indica en la *Gramática Filosófica* que la definición recursiva "es una regla para la construcción de reglas de sustitución, o también el término general de una serie de definiciones [formas]" (1974, 36, p. 851).[22] De forma muy elocuente, Wittgenstein (1975, §164) expresa gráficamente lo enunciado anteriormente mediante la siguiente figura:



[22] La noción de regla recursiva es un puente entre los primeros trabajos de Wittgenstein y sus posteriores escritos. Así, la forma general de una serie de formas, por ejemplo, '[0, ξ, ξ + 1]', expresa una regla (gramatical) del tipo '(a+ (ξ + 1)) = (a+ ξ)+1' (Mota, 2013, 2014, p. 35).

Esta figura muestra, por ejemplo, el paso (la regla) 'hacia delante' de un término a otro de la serie 2+0, 2+1, 2+2,…,2+5,… pero también muestra el paso (la regla) 'hacia atrás', por ejemplo 2+5 = (2+4)+1.

Por otra parte, una cosa es decir que el término 'estructura' se define inductivamente, lo cual no hace *ipso facto* a las estructuras recursivas, y otra muy diferente decir que una estructura es recursiva atendiendo a la organización interna de sus elementos.[23]

Podemos ver un ejemplo muy claro de esto en el trabajo de Lobina (2014a). Dicho autor, recoge la definición de Roberts (2006) que indica que una clase puede ser recursiva en el sentido de que la definición de esta clase puede hacer referencia a objetos de la misma clase (véase más abajo la definición de 'número natural').[24] Además, Lobina (2014b, p. 59) advierte que Rodgers y Black (2004) definen una estructura recursiva como aquella que está parcialmente compuesta por instancias más simples o pequeñas de la misma estructura de datos. Por otro lado, Wirth (1986, p. 109), nos indica que una estructura de datos es recursiva si un componente contiene a otro del mismo tipo. Así, en el caso de Roberts, la recursión es una propiedad de la definición, no de la clase, mientras que en el caso de Rodgers y Black, lo mismo que en el de Wirth, la recursión es una propiedad de la organización interna de la estructura. Así, se habla en el primer caso de una definición inductiva, mientras que en el segundo se habla de una estructura recursiva; dos cosas muy diferentes que *a priori* no deben confundirse. Veamos unos ejemplos.

Se puede definir inductivamente el término 'número natural' como sigue: (a) 0 es un número natural, (b) el sucesor de un número natural es un número natural. Asimismo, se puede definir el término 'estructura en árbol' de la siguiente manera (véase Wirth, 1986): (a) *O* es una estructura en árbol, llamada vacía, (b) si t_1 y t_2 son estructuras en árbol, entonces las estructuras consistentes en un nodo con dos descendientes $t1$ y t_2 son también estructuras en árbol binarias. En estos casos, la recursión es una propiedad que se constituye en la definición, pero no es una propiedad de los números naturales o las estructuras. Por tanto, la recursión es una propiedad constituida por la regla (Mota, 2013, 2014).

Nótese, por tanto la siguiente diferencia: (I) el término 'estructura' es definido inductivamente y la recursión es una propiedad de la regla, no de la estructura definida y (II) una estructura de datos es recursiva si un

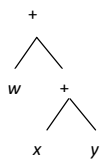
[23] Por 'estructura' se significa, en un sentido general, una organización determinada entre uno o varios elementos.

[24] Obsérvese que una serie "recursiva" no es más que una expresión de la regla/definición recursiva. En este caso, la recursión es una propiedad de (o constituida por) la regla.

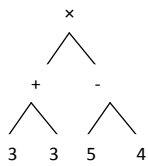
componente X contiene/incluye o está dentro de otro X del mismo tipo que él. Según esto, la definición recursiva, propia de la lógica matemática y de la teoría de la computabilidad cae fuera de (II) dado que: (a) 'recursión' en (II) se predica de estructuras, algo que no sucede en las mencionadas disciplinas (aunque sí en la Ciencia Cognitiva y de la Ciencia de la Computación) y (b) hace referencia a la configuración/organización interna de una estructura; lo cual no tiene sentido atribuir a una regla que sólo indica cómo proceder en la definición. Es decir, la relación entre los elementos que configuran la regla recursiva y los elementos que configuran la estructura de datos es diferente. Veámoslo con ejemplos.

La ecuación '2+3 = (2+2)+1' es una instancia de una regla recursiva (o una instancia del paso recursivo de una definición recursiva, ver supra), mientras que la organización interna de la expresión '(x+y)+w'

+		
T	**w**	
F	+	
	T	**y**
	T	**z**



o de la expresión '(3+3)×(5-4)':



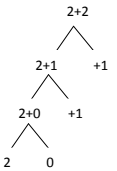
son sendos ejemplos de lo que se entiende en la Ciencia de la Computación como estructuras recursivas de datos: X dentro de X del mismo tipo (algo diferente a una definición recursiva).

Vuelvo a insistir entre la diferencia de una estructura recursiva y un procedimiento recursivo. En éste último caso, y como he señalado arriba, un procedimiento puede aplicarse recursivamente para computar '2+2', invocando un valor previamente computado para un argumento menor, esto es, '(2+1)+1' hasta que alcanza el caso base '(((2+0)+1)+1'. Esta implementación recursiva deriva directamente de la definición recursiva de la función suma (*Cf.* Abelson et al., 1996, entre otros):[25]

[25] Tanto Wirth (1986) como Abelson et al. (1996) indican que aunque una función se defina recursivamente ésta no tiene que implementarse necesariamente de manera recursiva (ver supra).

1. 2+2
2. (2+1)+1
3. (((2+0)+1)+1
4. (2+1)+1
5. 3+1
6. 4

Nótese que éste uso de recursión es diferente al que se predica de la forma de un proceso de cómputo, siendo tal una estructura recursiva de datos (un componente de la estructura en árbol incluye/contiene otro del mismo tipo):

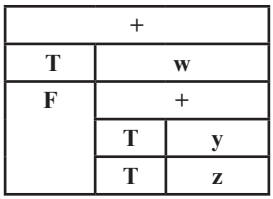


pero son 'recursivos' por razones diferentes; a saber, una regla permite substituir o reemplazar una expresión (2+2) por otra ((2+1)+1) mientras que una estructura tal exhibe la ya mencionada organización interna referente a un X (un componente) dentro de/contiene a otro X (componente) del mismo tipo. De hecho, Wirth (1986, p. 110) indica que una expresión como 2+2 exhibe una estructura recursiva como la siguiente:

+	
T	**y**
T	**z**

Esto es así de manera independiente de la regla recursiva, por lo que una vez más se concluye que la organización interna de una expresión como '2+2' o '(*a*+*b*)+*c*' no justifica la propiedad de la recursión entendida en su sentido original, como propiedad constituida por una regla (Mota, 2013, 2014). Dicho de otro modo, una regla no es recursiva porque una expresión que aparece en la misma exhiba una organización interna como las mostradas aquí.

Por otro lado, aunque en el caso de la regla no esté justificado hablar de auto-inclusión, sí lo está en el caso de las estructuras recursivas, pues no son las funciones las que se caracterizarían mediante esa propiedad, sino el esquema, la estructura que contiene un X dentro de otro X del mismo tipo; esto es, esquemas/componentes dentro de esquemas/componentes del mismo tipo, como en:



Esto es análogo a lo que se puede predicar de los constituyentes o de la configuración de E-N-C definidas por Moro (ver supra), siendo una instancia de un esquema dentro de otro de exactamente del mismo tipo. Si, como vengo mostrando, la recursión en su sentido original no hace referencia a la organización interna de un X dentro de otro X del mismo tipo, propongo reservar la noción de recursión a su sentido original y aplicar en estos otros casos la denominación más apropiada de 'auto-inclusión', estando, así, de acuerdo con Luuk y Luuk (2011, p. 9) en este punto. Nótese bien que esta propuesta es mucho más que una simple cuestión terminológica.

Addendum sobre recursión y auto-referencia

No es infrecuente encontrar en la literatura sobre recursión que tal propiedad es equiparable o se puede identificar con la auto-referencia (Tomalin, 2006, Lobina, 2011, 2014a, 2014b). La auto-referencia se puede definir, siguiendo a Odifreddi (2001), en términos de notación de conjuntos, como '$x \in x$', que puede traducirse como *'x* pertenece a sí mismo'. Un conocido ejemplo de esta formulación general es la Paradoja de Russell sobre la clase que contiene a todas las clases que no pertenecen a sí misma ($R = \{x_1, x_2, x_3, \ldots x_n\}$; pudiendo estar el conjunto vacío). Si por ejemplo la clase x_1 fuera miembro de sí misma (R), entonces no sería miembro de sí misma ($\neg(x \in x)$), pero si no es miembro de sí misma ($\neg(R \in R)$) entonces sería miembro de sí misma ($(R \in R)$): contradicción.

Como he venido señalando, en el caso de las estructuras que exhiben un X dentro de otro X del mismo tipo también cabría hablar de auto-referencia (véase también Lobina, 2011, 2014a, 2014b). Así, una estructura con un X dentro de un X del mismo tipo, esto es, una estructura con auto-inclusión (Arsenijević y Hinzen, 2012), es una estructura que exhibe auto-referencia. Si esto es así, entonces se sigue de la premisa expresada más arriba (esto es, que *x* es recursivo porque *x* es auto-referente) que tal estructura es recursiva. De esto se sigue, por ende, que la auto-inclusión es un tipo de auto-referencia y por tanto un tipo de recursión. Mi intención es hacer una reflexión conceptual entre las nociones de 'recursión' y 'auto-referencia' y ver en qué sentido podemos decir que x es recursivo porque x es auto-referente.

En el caso de las funciones parece que la relación no es exactamente igual que en el caso anterior. Así, no se puede decir que una función *f* contiene otra función f del mismo tipo. Siguiendo a Wittgenstein (1922), en el *Tractatus Logico-Philosophicus* argumentó en respuesta a la Paradoja de Russell que una función no puede contenerse a sí misma; esto es, no puede ser su propio argumento, dado que el signo para función '*F*( )' contiene la protofigura de su argumento, y no puede contenerse a sí misma (3.333). Así, en '*F*(*F*(*x*))' la función externa (de segundo orden o nivel) e interna (de primer orden), aun usando ambas la misma letra, tienen significados distintos. Es evidente pues que son funciones de diferente tipo. Así, la función interna, nos dice Wittgenstein, tiene la forma $\phi(x)$ y la externa la forma $\psi(\phi(x))$ (Mota, 2013, 2014).

Esta relación de (auto-)inclusión diferencia claramente la recursión que se predica como propiedad constituida por una definición (esto es, por una regla) de la que se predica de la organización interna de una estructura gramatical o de datos (ver supra). Así, en el caso de las estructuras, la recursión se predica de la inclusión de X dentro de X del mismo tipo, sean X sintagmas, esquemas, estructuras en árbol,…, mientras que en el caso de las reglas o las definiciones (por ejemplo de una función o de un proceso) no existe tal relación de inclusión, sino que una expresión se substituye por otra que está constituida por un valor definido con anterioridad para un argumento menor (Mota, 2013, 2014).

Me gustaría terminar este apartado indicando dos cosas. La primera es llamar la atención sobre la extensión de 'recursión' y 'auto-referencia'. Así, si equiparamos 'recursión' y 'auto-referencia', entonces nos encontramos con que expresiones como 'Yo miento', 'Yo soy indecidible' o 'esta expresión tiene cinco palabras' serían recursivas puesto que son auto-referentes. Es cierto que la auto-referencia en estos casos es semántica más que sintáctica pero no se ha especificado en la literatura que tenga que restringirse necesariamente a esto último. Si decimos que todo aquello a lo que subyace la auto-referencia es recursivo entonces la noción de recursión se usará en un sentido muy diferente al original. Esto es, no sólo se predicará de objetos de los que no se predicaba en origen sino que adoptará otros significados distintos (por ejemplo el que se refiere a la auto-referencia semántica).

La segunda cuestión que quería indicar es que aunque se pueda decir que la recursión usada en su sentido original y la recursión entendida como X dentro de X del mismo tipo (esto es, auto-inclusión) presentan algún tipo de auto-referencia, yo reservaría, por lo dicho anteriormente, el término 'recursión' para expresar su sentido original y emplearía el de 'auto-inclusión' en el segundo caso. Esto lo indico por dos razones: (I) no todo lo auto-referente

es recursivo, entendiendo por 'recursivo' su significado primario y original: como acabo de indicar, 'recursión' adquiriría un significado diferente cuando se usara para indicar, por ejemplo, la auto-referencia propia de la Paradoja de Russell; y (II) cuando Soare propuso usar 'recursión' en su sentido original y no como sinónimo de computabilidad (ver supra) lo hizo, entre otras cosas, para no confundir tal sentido original. Así, por ejemplo, la auto-referencia, por lo demás legítima, de una expresión como 'esta expresión tiene cinco palabras' no indica, a la luz de lo expuesto anteriormente, recursión en su sentido original.[26] Por ello, llamo la atención sobre la relación entre 'recursión' y 'auto-referencia', dado que quizá, no todo lo que presenta auto-referencia sea recursivo si queremos preservar el sentido original del término.

Conclusiones

En este trabajo he analizado el uso que del término 'recursión' se hace en los dominios de la Lógica Matemática y la Teoría de la Computabilidad, de la Ciencia Cognitiva y, finalmente, de la Ciencia de la Computación. Se pueden establecer dos usos derivados, por una parte, de su aplicación a la caracterización de un modo de definición, esto es, de una regla (el cual hace referencia a su uso original y primario), y por otra, de su aplicación a la configuración interna de una estructura. De forma similar, he analizado la noción de recursión y su relación con la auto-referencia, indicando que, quizá, no todo lo que presenta auto-referencia sea recursivo, al menos si se entiende 'recursión' en su sentido original y primario. Derivado del análisis, concluyo, de forma similar a como hicieran Luuk y Luuk (2011), que se podría reservar, sin menoscabo conceptual alguno, la noción de recursión a su sentido original y usando, por ejemplo, auto-inclusión, para referirnos a una configuración interna de estructuras con X dentro de X del mismo tipo.

Referencias

Abelson, H. & Sussman, G. J. with Sussman, J. (1996). *Structure and interpretation of computer programs*. Cambridge, MA: MIT Press.

Arsenijevic, B. & Hinzen, W. (2010). Recursion as a human universal and as a primitive. Biolinguistics, 4(2-3), 165-173.

Arsenijević, B. & Hinzen, W. (2012). On the abscence of X-within-X recursion in human grammar, *Linguistic Inquiry*, 43, 423-440.

[26] Para Corballis (2011) "Pienso luego existo" es un ejemplo de recursión, entendida como un X (un pensamiento nos dice él) dentro de sí mismo. Siguiendo mi análisis, este ejemplo exhibe recursión entendida como X dentro de X del mismo tipo (esto es, auto-inclusión) y auto-referencia. Sin embargo, este sentido estructural no es el mismo que el de la definición recursiva. Este es un ejemplo de, en mi opinión, una generalización de la noción de recursión.

Boolos, G. & Jeffrey, R. (1974). *Computability and logic*. Cambridge: Cambridge University Press.
Boolos, G. (1971). The iterative conception of set, *The Journal of Philosophy*, 68, 215-231.
Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control*, 2, 137-167.
Chomsky, N. (1965). *Aspects of theory of syntax*. Cambridge. MA: MIT Press
Chomsky, N. (1995). *The minimalist program*. Cambridge. MA: MIT Press.
Chomsky, N. (2006). *Language and mind*. Cambridge: Cambridge University Press.
Chomsky, N. (2007a). Of minds and language, *Biolinguistics*, 1, 9-27.
Chomsky, N. (2007b). Approaching UG from below. In U. Sauerland & H. M. Gärtner (Eds.), *Interfaces + Recursion = Language?* (pp. 1-30). Berlin: Mouton.
Chomsky, N. (2008). On phases. In R. Freidin, C. Otero, & M. L. Zubizarreta (Eds.), *Foundational issues in linguistic theory* (pp. 133-166). Cambridge. MA: MIT Press.
Chomsky, N. (2010). Some simple evo devo theses: How true might they be for language? In R. Larson, V. Déprez & H. Yamakido (Eds.), *The evolution of human language* (pp. 45-62). Cambridge: Cambridge University Press.
Chomsky, N. (2011). Language and other cognitive systems. What is special about language?. *Language Learning and Development*, 7, 263-278.

Chomsky, N. (2012). Some core contested concepts. *Proceedings of the CUNY 2012*, 1-18.
Church, A. (1932). A set of postulates for the foundation of logic, The Annals of Mathematics, 33, 346-366.
Church, A. (1936). An unsolvable problem of elementary number theory. In M. Davis (Ed.), *The undecidable* (pp. 88-107). New York: Raven Press.
Corballis, M. C. (2007). Recursion, language and starlings, *Cognitive Science, 31*, 69-704.
Corballis, M. C. (2011). *The recursive mind: The origins of human language, thought, and civilization*. Princeton, NJ: Princeton University Press.
Cutland, N. (1980). *Computability: an introduction to recursive function theory*. Cambridge: Cambridge University Press.
Epstein, R. & Carnielli, W. (1989). *Computability: computable functions, logic, and the foundations of mathematics*. Pacific Grove, CA: Wadsworth & Brooks/Cole.
Everett, D. (2005). Cultural constraints on grammar and cognition in Pirahã, *Current Anthropology, 46*(4), 621-646.
Everett, D. (2009). Pirahã culture and grammar: a response to some criticisms, *Language, 85*(2), 405-442.
Fitch, T. (2010). Three meanings of recursion: key distinctions for biolinguistics. In R. Larson, V. Déprez & H. Yamakido (*Eds.*), *The evolution of human language* (pp. 73-90). Cambridge: Cambridge University Press.
Frascolla, P. (1994). *Wittgenstein's philosophy of mathematics*. London: Routledge.

Gödel, K. (1931). On formally undecidable propositions of the Principia Mathematica and related systems. I. In M. Davis (*Ed*.), *The undecidable* (pp. 4-38). New York: Raven Press.
Gödel, K. (1934). On undecidable propositions of formal mathematical systems. In M. Davis (*Ed*.), *The undecidable* (pp. 39-74). New York: Raven Press.
Gödel, K. (1964). Postscriptum to Gödel 1931. In M. Davis (*Ed*.), *The undecidable* (pp. 71-73). New York: Raven Press.
Hauser, M., Chomsky, N. & Fitch, T. (2002). The faculty of language: What is, who has it, and how did it evolve?, *Science, 298*, 1569-1579.
Hrbacek, K. & Jech, T. (1999). *Introduction to Set Theory*, New York: Marcel Dekker, Inc.
Jackendoff, R. & Pinker, S. (2005). The nature of the language faculty and its implications for evolution of language (reply to Fitch, Hauser, and Chomsky), *Cognition, 97*, 211-225.
Jackendoff, R. (2011). What is the human language faculty? Two views, *Language, 87*, 586-624.
Karlsson, F. (2010). Syntactic recursion and iteration. In H. van der Hulst (*Ed*.), *Recursion and human language* (pp. 43-67).
Kinsella, A. (2010). Was recursion the key step in the evolution of the human language faculty? In H. van der Hulst (Ed.), *Recursion and human language* (pp. 179-191).
Kleene, S. C. (1938). On notation for ordinal numbers, *The Journal of Symbolic Logic, 3*, 150-5.
Kleene, S. C. (1943). Recursive predicates and quantifiers, *Transactions of the American Mathematical Society, 53*, 41-73.
Kleene, S. C. (1952). *Introduction to metamathematics*. Amsterdam: North-Holland Publishing.
Lobina, D. J. (2011). Recursion and the competence/performance distinction in AGL tasks, *Language and Cognitive Processes*, 26, 1563-1586.
Lobina, D. J. (2014a). Probing recursion. *Cognitive Processing*, DOI 10.1007/s10339-014-0619-z
Lobina, D. J. (2014b). What linguists are talking about when talking about..., *Language Sciences, 45*, 56-70.
Luuk, E. & Luuk, H. (2011). The redundancy of recursion and infinity for natural language, *Cognitive Processing, 12*, 1-11.
Marion, M. (1995). Wittgenstein and finitism, *Synthese, 105*, 141-176.
Marion, M. (1998). *Wittgenstein, finitism, and the foundations of mathematics*. Oxford: Oxford University Press.
Marion, M. (2009). Radical anti-realism, *Wittgenstein and the length of proofs, Synthese, 171*, 419-432.
Moro, A. (2008). *The boundaries of Babel*. Cambridge, MA: MIT Press.
Mota, S. (2013). La propiedad de la recursión en el "Tractatus Logico-Philosophicus" de Wittgenstein y su relación con la Teoría de la Computabilidad y la Lógica

Matemática, 17, *Observaciones Filosóficas*. http://www.obervacionesfilosoficas.net/lapropiedaddelarecursion.htm
Mota, S. (2014). La historia y la gramática de la recursión: una precisión desde la obra de Wittgenstein, *Pensamiento y Cultura, 17*, 20-48.
Odifreddi, P. (2001). Recursive functions: an archeological look. In C.S. Claude, M.J. Dinneen & S. Sburlan (*Eds*.), *Combinatorics, Computability and Logic* (pp. 13-31). London: Springer-Verlag.
Pinker, S. & Jackendoff, R. (2005). The faculty of language: What's special about it?, *Cognition, 95*, 201-236.
Post, E. (1921). Introduction to a general theory of elementary propositions, *American Journal of Mathematics, 43*, 163-185.
Post, E. (1943). Formal reductions of the general combinatorial decision problem, *American Journal of Mathematics, 65*, 197-215.
Post, E. (1944). Recursively enumerable sets of positive integers and their decision problems. In M. Davis (*Ed*.), *The undecidable* (pp. 305-337). New York: Raven Press.
Roberts, E. (2006). *Thinking Recursively with Java*. Hoboken, NJ: John Wiley and Sons, Inc.
Roddych, V. (1999a). Wittgenstein on irrationals and algorithmic decidability, *Synthese, 118*, 279-304.

Rodgers, P., Black, P.E. (2004). Recursive data structure. In: Pieterse, V., Black, P.E. (Eds.), Dictionary of Algorithms and Data Structures. Online at: http://www.nist.gov/dads/HTML/recursivstrc.html.
Rodych, V. (1997). Wittgenstein on mathematical meaningfulness, decidability, and application, *Notre Dame Journal of Formal Logic, 38*, 195-225.
Rodych, V. (1999b). Wittgenstein's inversion of Gödel's Theorem, *Erkenntnis, 51*, 173, 206.
Rodych, V. (2002). Wittgenstein on Gödel: the newly published remarks, *Erkenntnis, 56*, 379-397.
Rodych, V. (2003). Misunderstanding Gödel: new arguments about Wittgenstein and new remarks by Wittgenstein, *Dialectica, 57*, 279-313.
Shanker, S.G. (1987). Wittgenstein versus Turing on nature of Church's thesis, *Notre Dame Journal of Formal Logic, 28*, 615-649.
Sieg, W. (1997). Step by recursive step: Church´s analysis of effective calculability, *The Bulletin of Symbolic Logic, 3*, 154-180.
Skolem, T. (1923). The foundations of elementary arithmetic established by means of the recursive mode of thought, without the use of apparent variables ranging over infinite domains. In J. Van Heijenoort (*Ed*.), *From Frege to Gödel. A source book in mathematical logic, 1879-1931* (pp. 302-333). Cambridge, MA: Harvard University Press.
Soare, R. (1996). Computability and recursion, *The Bulletin of Symbolic Logic, 2*, 284-321.
Soare, R. (2009). Turing oracles machines, online computing, and three displacements in computability theory, *Annals of Pure and Applied Logic*, 160, 368-399.

Tomalin, M. (2006). *Linguistics and the Formal Sciences*. Cambridge: Cambridge University Press.
Tomalin, M. (2007). Reconsidering recursion in syntactic theory, *Lingua, 117*, 1784-1800.
Tomalin, M. (2011). Syntactic structures and recursive devices: A legacy of imprecision, Journal of Logic, *Language and Information, 20*, 297-315.
Turing, A. (1937). On computable numbers, with an application to the Entscheidungsproblem. In M. Davis (*Ed.*), *The undecidable* (pp. 116-151). New York: Raven Press.
Wirth, N. (1986). *Algorithms and data structures*. Hemel Hempstead, UK: Prentice Hall. © N. Wirth 1985 (Oberon version: August 2004).
Wittgenstein, L. (1922). *Tractatus logico-philosophicus*. London: Routledge.
Wittgenstein, L. (1974). *Philosophical grammar*. Traducción de Luis F. Segura, ed. 1992. México, D.F.: UNAM.
Wittgenstein, L. (1975). *Philosophical remarks*. Traducción de Alejandro Tomasini Bassols, ed. 1997. México, D.F.: UNAM.
Wittgenstein, L. (1978). *Remarks on the foundations of mathematics*. Oxford: Blackwell.
Wrigley, M. (1977). Wittgenstein's philosophy of mathematics, *Philosophical Quarterly, 27*, 50-59.
Zwart, J.W. (2011). Recursion in language: A layered-derivation approach, *Biolinguistics, 5*, 43-56.