Informe de Tesis de Doctorado en Depósito: *Topics in Programming Languages, a Philosophical Analysis through the case of Prolog*

[1]Universidad de Salamanca

{u131277@usal.es}

Resumen. *Se intenta con este documento exponer un informe de la tesis "Topics in Programming Languages, a Philosophical Analysis through the case of Prolog", integrando no sólo él título apuntado, como también el índice, la introducción, un resumen significativo y las conclusiones de la tesis doctoral en castellano, conforme Normativa de la USAL (ART.14.3).*
Palabras clave*: Recursividad, Lenguages de programación, Prolog, Cálculo lambda, Lisp, Algol 60/68, C/C++.*

Copia. *Traducción del Resumen presente en la tesis (pp. v-vii):*
"Los lenguajes de programación rara vez encuentran un anclaje adecuado en la filosofía de la lógica, del lenguaje y de la ciencia. Lo que es más, la filosofía del lenguaje parece estar restringida a las lenguas/lenguajes naturales y la lingüística, y incluso la filosofía de la lógica rara vez es encuadrada en los temas de los lenguaje de programación. La historia de los lenguajes naturales es intrínsecamente acústica-hacia-visualidad, de fonética-hacia-escritura, mientras que los lenguajes de programación, bajo la interacción hombre-máquina, aspiran mejor a una visualidad-hacia-acústica, a una escritura-hacia-fonética, a través de la idea y técnica de procesamiento de lenguajes naturales. Un lenguaje de programación como Prolog tiene el peculiar atributo de haberse desarrollado en el paso de los estudios de gramática, lingüística y gramática formal-simbólica hasta las empresas computacionales muy específicas e incipientes del procesamiento del lenguaje natural y la programación lógica. El recuento de los antecedentes filosóficos, mecánicos y algorítmicos de Prolog nos avala al trasfondo mecánico-computacional explorado por Pascal, Leibniz, Boole, Jacquard, Babbage, Zuse, hasta llegar al ACE (Alan Turing) y al EDVAC (von Neumann). La teoría de la recursión y la computabilidad (Turing, Church, Gödel, Kleene), la criptografía y la teoría de la información (Shannon), nos permiten interpretar el dominio evolutivo de los lenguajes de programación. La línea de Cálculo lambda a Lisp, y de Lisp a los lenguajes de programación de la familia Algol, la división de lenguajes de raíces procedural y declarativa con el lenguaje C y Prolog, y la sintaxis y la explosión de ramificación semántica subsiguientes se investigan aquí en relación con el élan *original de ambas lenguajes C y Prolog. Los diferentes enfoques de la filosofía de las matemáticas – estructuralista, formalista, logicista e intuicionista – se enfrentan así a la computabilidad y, además, a la programación lógica y a Prolog, estudiando de cerca autores como Frege, Gödel y Wittgenstein. La naturaleza logicista, completa y atomista de primer orden de la programación lógica, que se conforma*

con las cláusulas de Horn con Resolución SLD, se examinará posteriormente. Del mismo modo, el camino de la semiótica lingüística y el giro lingüístico del Saussuriano Course Générale a la jerarquía Chomsky-Schützenberger, con una mirada clara hacia la teoría de autómatas y teoría de la complejidad, en general con un fondo común en la antigua Gramática Sánscrita de Pāṇini, impulsa la investigación a un espectro mucho más amplio que el mero uso de la forma Backus-Naur en semiótica Saussuriana y lenguajes de programación, por sí mísmo ya frecuentemente olvidado. Esto permite describir Prolog como un producto típico de by-pass *del logicismo computacional y del estructuralismo lingüístico, con el diseño filosófico de la jerarquía Chomsky-Schützenberger, especialmente apto para la ilusión de la Inteligencia Artificial (fuerte), como se ve por su entrada en el Sistema de la Quinta Generación de Computadoras (FGCS) en Japón. También produce los medios para entender cómo la sintaxis-estructuralista y la destitución fonológica en la historia del procesamiento del lenguaje natural (de primera mano en la demostración automatizada de teoremas, pero también en el aprendizaje automático estadístico) y en la programación lógica y Prolog (a través de sistemas Q), son signos de una destitución más general de la filosofía de la música en la filosofía del lenguaje occidental, razón suficiente para convocar al medievalista Boecio, y la procedencia contemporánea de la teoría generativa de la música tonal. Las pisadas separadas e individuales de los diferentes artífices de Prolog – el trabajo de Alain Colmeraueur en Resolución SL y en el manual de Prolog, y el trabajo de Robert Kowalski sobre el desarrollo del paradigma de programación lógica Turing-completo – se enmarcan en la investigación, de igual modo que de manera más amplia la filosofía crítica kantiana, con el objetivo de dotar de una riqueza clásica la artificialidad, informática y filosofía de la información, estudios de tecnología y lenguajes de programación, con Prolog resaltado. Las antinomias de Kant son revisadas a la luz de estas premisas, y las nuevas antinomias informáticas interactúan con el concepto de virtualidad en la era de computus, en contraste con la del calculus. Más estrictamente, la doctrina kantiana de schemata se enfrenta al núcleo de la teoría de lenguajes de programación a través de un tema que el autor ha llamado, convencionalmente, $\bigcup$-mentalism, es decir, la posibilidad de igualar funciones computables con jerarquía de visión artificial o visión por computador y, de mismo modo que la inversión del relato de esquemas de Kant, proponemos la inversión adecuada de la arquitectura de la computadora de von Neumann o arquitectura de Princeton (la filosofía de la imagen y los conceptos ontológicos, como La Pensée et le Mouvant, Matière et mémoire de Bergson, y L'Image-Mouvement y L'image-Temps de Deleuze se revelan muy adecuados para una tal reflexion). Con esto también se presenta el encuentro con la geometría de paralelas Euclidiana, una tradición de larga duración hasta Bolyai-Lobachevsky, principalmente bajo la relajación del quinto postulado (de las paralelas), que tuvo un destino determinante en la filosofía de la física (desde Riemann hasta Einstein). De la misma manera, las pruebas de diagonalización de Cantor a Gödel (y el proceso de diagonalización contenido en computabilidad y criptografía en comunicación informacional) demuestran adecuadamente la historia de computación, con dimensiones topológicas fla-*

grantes incluidas. La programación lógica y Prolog son explorados a lo largo de esta línea, recurriendo al tema de los límites mínimos de restricción a la lógica e inherentes paradojas, inevitablemente constituidas también en lenguajes de programación y teoría de lenguajes de programación."

***Palabras clave**: Recursividad, Lenguages de programación, Prolog, Cálculo lambda, Lisp, Algol 60/68, C/C++,.*

1. Índice de la Tesis (en lengua castellana):

2. Resumen Significativo

La tesis comienza por la enumeración y análisis de los fundamentos básicos de Prolog y de la programación lógica, en el contexto de la filosofía de la ciencia. Así, la unificación de Robinson, la 'vuelta atrás' del algoritmo DPLL o 'backtracking', las cláusulas de Horn o el teorema de Herbrand son ejemplos de citas directas, dentro del fondo más profundo de filosofía de la ciencia y la lógica. Se ofrece una panorámica bajo la cual se guían conceptos como aquellos presentes en la obra de McCarthy, Church, Tarski, Gödel o Turing para que estén, desde el inicio de la refléxion, correctamente abarcados. Inevitablemente, los autores Alain Colmeraeur, Philippe Roussel y Robert Kowalski también se quedan conectados con sus especiales contribuiciones a este núcleo de la investigación. Se vuelve a la conclusión introductoria de que el procesamiento del lenguaje natural y la programación lógica equiparan, en forma encapsulada en la era computacional y respectivamente, el lugar de los estudios de gramática y de la lógica formal en el espectro más amplio de la historia del lenguaje y de la lógica. Una nota muy concisa acerca de 'comentarios históricos & filosóficos preliminares' es aquí introducida. Ella permite que se alcanze un *substratum* filosófico más profondo, en el sentido en que la filosofía original

de Leibniz en relación a la computación evoluciona para ser confrontada con la propia imbricación entre cosmologia y razón partiendo de Kant, pero también de la filosofía de la física contemporánea, hasta una construcción de hipótesis fabricada por el autor, con el intento de considerar líneas próximas entre la teoría de la computabilidad (la tesis de Turing-Church) (1936) (la máquina de Turing y cálculo lambda) y conceptos geométricos libres como es el caso del Programa de Erlangen de Felix Klein, hasta el punto donde se considera la propria vertiente geométrica-topológica de la máquina de Turing y, como consecuencia, es permitida la introducción de la idea de U-mentalism, partiendo de la inversión del esquema Kantiano y simultáneamente de la arquitectura de computadoras de von Neumann o de Princeton. Aquí la facultad de la imaginación Kantiana es central en la comprensión de la cuestión.

La tesis procede al estudio detallado de una forma cercana a los textos originales, de la arquitectura de computadoras de von Neumann, al mismo tiempo que del estudio asociado de neuronas artificiales (MacCulloch & Pitts). Sucede de la misma manera con la teoría de la información de Shannon, a partir de la cual se catapulta para temáticas más avanzadas, como sea la história moderna y contemporánea de la ciencia europea, con un enfoque en las diferentes perspectivas físicas y filosóficas radicadas desde la oposición entre Newton y Leibniz, y el razonamiento sobre el envolvimiento de lo precedente con temas avanzados en la filosofía crítica (Kant) y la filosofía natural (Darwin), sin olvidar importantes incursiones en la teoría de la complejidad y sus aspectos intrínsecamente más desafiantes como las entidades cosmológicas aleatorias del cerebro de Boltzmann, o entidad hipotética consciente de sí misma. Abundantes referencias a la historia de las computadoras (como las industriales máquina analítica de Babbage, o el telar de Jacquard, hasta las modernas mecánico-electrónicas y digitales ENIAC/EDVAC y Colossus) son también arraigadas en la argumentación filosófica, asomando cada vez más claro el principio (euclidiano) de computación paralela moderna con el funcionamento de argumento de diagonalización (y de paradojas) simultánea en la máquina de Turing y, consecuentemente, en el estado del arte de la computación, en el estado de ignoramibus acerca de problema de P vs NP.

Se trata después de nuevas antinomias informáticas, tratando directamente sobre la *Crítica de la Razón Pura* de Kant, bajo la premisa de que los juicios sintéticos a priori después de los teoremas de incompletitud y la estipulación de funciones computables en la teoría de la recursividad han obligado a una consideración renovada de los principios antinómicos de la analítica trascendental. Esto también es para mostrar más ejemplarmente el paso del paradigma de calculus hasta lo de computus. Una marca especial en estudios naturales en la construcción de la artificialidad es, por lo tanto, conocida, revelando a lo mejor de nuestro esfuerzo las consecuencias generales de conceptos como el del problema de decisión, en contraste con la metafísica transcedental.

Sigue un apéndice muy corto sobre virtualidad, donde la noción de extensionalidad en el dominio de funciones computables se soporta con una jerarquía ponderada de geometría a topología, con la máquina de Turing como el agente transformacional (asimismo ideal y sensible en términos kantianos).

El contenido del Capítulo 2 trata sobre procesamiento del lenguaje natural en Prolog. Se explora de manera práctica, propenso a ejemplos extraídos de las inmensas posibilidades de la jerarquía de Chomsky, antes del apartado de interludio musical, en el que se convoca la filosofía medieval de Agustín y Boecio, con el objetivo de construir con argumentos más sólidos un conjunto y estructura diagramática donde la jerarquía de Chomsky es su-

bordinada al tríptico de Boecio.
La Parte II es denominada 'Core Prolog' y desde aquí empienza toda la parte más técnica sobre el lenguaje de programción Prolog y, además, su idiosincrasia en lo que se refiere al paradigma de programación lógica, desde luego a partir del manejo de la consola SWI-Prolog. La máquina abstracta de Warren demonstra en su núcleo poseer en su núcleo poseer los requisitos para una discussión más avanzada sobre la célebre fórmula de Kowalski 'Algoritmo= Lógica + Control'. Recursivamente se evoluciona en términos abstractos en Prolog a partir de fragmentos de notaciones seleccionadas de Backus-Naur, sobre el fondo de la influencia original antigua de la Gramática Sánscrita de Pāṇini. En conjunto, de esa manera, y convocando términos como suposiciones abiertas y cerradas del mundo, al mismo tiempo que acompañando la ontología de las formas normales conjuntiva (CNF) y disyuntiva (DNF), se invocan contenidos emparentados con la disciplina científica de visión artificial y, claro, de la visión de múltiple geometría en traducción libre, o de un género de geometría epipolar tanto computacional cuanto ontológica, tanto digital cuanto fotónico, así preparando la consideración de la fundición de visión artificial con (teoria de) lenguajes de programación, para lo cual es esencial la mirada de las diferencias entre Wittgenstein I y Wittgesntein II. Inevitablemente, la filosofía de las matemáticas de Wittgenstein, junto con otros autores como Gödel, es minuciosamente tratada incluso a partir de textos originales. Después de este estudio, la sintaxis y la semántica de Prolog se toman paso a paso. El último párrafo, designado 'Fragmentos Convergentes Anteriores de Calculo Lambda, Lisp y Algol 60-68, y Fragmentos Divergentes Posteriores de C/C ++, Java y Phyton' es orientado a servir también de conclusión en cuanto, no sólo la filogenia de lenguajes de programación, en el caso particular de Prolog, como también cuanto al isomorfismo entre las subsecuentes familias, en lugar de una conclusión más discursiva.

3. Conclusiones

Por cuestiones de economía de texto, se ha optado por exponer directamente el isomorfismo en código, repasando el cálculo lambda, Lisp y Algol 60/68 dentro de una lógica anterior, y también C/C++, Java, y Python, dentro de una lógica posterior. Por razones históricas relacionadas con Prolog, el cálculo lambda en cuanto sistema formal equivalente a la definición de función y de la propia recursión, Lisp también en cuanto lenguaje de programación de alto nivel con notación de prefijo y, naturalmente, Algol 60/68 en cuanto lenguaje de programación con bloques de código, han merecido una mayor dinámica de análisis, comparando con las últimas tres. Tomando en consideración que C/C++ ya había sido desarrollado por comparación con Prolog, la idea ha sido precisamente dejar al final una muestra de código de alguns aspectos isomórficos de lenguajes de programación, como recordando la idea de que una jerarquía abstracta de ideas siempre está presente en la formación del lenguajes desde un punto de vista universal. El algoritmo escogido ha sido el de la criba de Eratóstenes.
Dentro de una lógica de una conclusión más alargada, y como aclaración técnica adicional, traducimos aquí los últimos párrafos antes de la exposición de código en la tésis:
'Algol 60, en detalle, es un producto de la filosofía típica de Wittgenstein I, que representa la ascendencia de la mayoría de los lenguajes de programación modernos (tipo Algol). Un ejemplo bastante sencillo y fácilmente visible de tal es la cita principal en el 'Revised Report on the Algorithmic Language Algol 60' (1962), reviviendo directamente Wittgenstein I del 'Tractatus Logico-Philosophicus' (1922):

'Lo que se puede decir, se puede decir claramente; De aquello de lo que no podemos hablar debemos guardar silencio.'
Algol 60 centra y garantiza la mayoría de las implementaciones de Algol indiscutiblemente. Algol 68 fue criticado por algunos miembros de su comité de diseño como CAR Hoare y E. Dijkstra, por abandonar la simplicidad de Alogl 60, acusando a la empresa de un gran riesgo de complejidad, mientras que el compilador exigía mucho trabajo, pero uno de esos vehículos de programación es obviamente pertinente para anticipar los años de Prolog, y no solo los de Prolog, sino también para comenzar el lenguaje C.
'Algol 60 fue particularmente influyente en el diseño de lenguajes posteriores ya que introdujo la estructura de bloques anidados, el alcance léxico y una sintaxis en la forma de Backus-Naur (BNF). Casi todos los lenguajes de programación subsiguientes han usado una variante de BNF para describir la sintaxis sin contexto ', pero lo superlativo en importancia es que llevó menos de una década estimular la siguiente gramática innovadora de 'comunicación hombre a hombre '– la gramática de Van Wijngaarden en Algol 68 – que establece Algol no solo como la marca de agua de una época de (programación en informática) filosofía del lenguaje, habiendo lanzado programación científica y estudios de compilación, como, en continuidad, además del hecho de que una definición formal completa existente no necesitaba de un compilador para tener éxito, la confluencia del diseño de lenguajes de programación con un lenguaje de programación en sí mismo fue realmente nuevo.
Con esto, estamos de acuerdo con HT de Beer, autor de The History of the Algol Efoort (2006), quien escribió: 'Yo sostengo que el esfuerzo de Algol fue un catalizador para la transformación del campo de la escritura de compiladores y lenguajes de programación en un campo científico. El *ALGOL 60 report* fue la clave de esta transformación y esto explica la influencia del *ALGOL 60 report*; su alcance era mucho más amplio que otro lenguaje de programación algorítmico, se trataba de la definición de lenguajes de programación en general.' Si recordamos que desde que Wegstein propuso tres niveles diferentes de representaación – referencia, publicación y hardware – sobreviviendo a Algol 60 y Algol 68, es comprensible qué tanto una teoría del significado y el estudio filosófico de la verdad, en el sentido de Tarski, invitando a la corrección formal y la adecuación material al reino de los lenguajes de programación –'estructura y contenido deben ser los mismos para todas las representaciones' – se propuso la explosión semántica de los lenguajes de programación en la medida exacta en que se construyó una sintaxis bien definida.
Por razones de economía, ahora ejemplificaremos los fragmentos de código tanto de Algol 60 como de Algol 68, y cómo se relacionan con fragmentos de C / C ++, Java y Phyton, con esta esperanza de mostrar, de forma simple y gráfica, el isomorfismo inherente de los lenguajes de programación – variables del programa en conjunción con células de almacenamiento informático (Wittgenstein I), instrucciones de almacenamiento en relación con instrucciones de control (Shannon), instrucciones de almacenamiento en relación con instrucciones de asignación (von Neumann) y expresiones en ligación con instrucciones aritméticas y referencia de memoria (Turing).
Observador como uno debería ser, Algol 60 habitualmente presenta la forma Backus-Naur (BNF), mientras que lo que se puede ver en Algol 68 es la aplicación del formalismo gramatical de dos niveles (libre de contexto) de la gramática van Wijngaarden, pavimentando el camino a la misma aplicación en la traducción automática de Q-systems (y luego en Prolog). Las características de las gramáticas-W y de Algol 68, las dos juntas, han permiti-

do la inclusión de una sintaxis basada en expresiones, con tipos declarados por el usuario y estructuras identificadas por su marca y su conjunto de nombres de campo. También han transportado un modelo de referencia de variables y parámetros de referencia. La extracción de cadenas y matrices, y la ejecución de órdenes parciales en concurrencia, son características mucho más plásticas – congruentes con la característica general de las gramáticas-W de poder codificar el universo potencialmente infinito de gramáticas libres de contexto en un conjunto finito de reglas –, que dan testimonio de una etapa fundamental en la evolución de los lenguajes de programación, es decir, una de especiación rápida en aparente contraste con una de 'equilibrio puntuado' o de stasis. Con esto también nos referimos a una especie de atavismo filogenético de lenguajes en informática, una vez que la influencia de los lenguajes tipo Algol fue integral y de larga duración, aunque sus rasgos solo se identificaron cuando reaparecieron en lenguajes de programación posteriores. Y con esto también nos referimos a algo muy cercano a un sentido informático–morfológico, no solo porque Algol – un esfuerzo de colaboración realizado por un comité internacional de la ACM (1958-1960) –, inauguró bloques estructurados de código, delimitaciones de inicio / fin y funciones anidadas con un alcance léxico, que permitía, como Lisp, procedimientos de programación recursiva, pero también porque pertenece a una época en la que, afortunadamente, los lenguajes de programación se diseñaron sin que hubiera una comprensión clara sobre si realmente eran un lenguaje de interpretación para el compilador, un informe del compilador, un lenguaje de programación real, o de hecho una meta-teoría de diseño de lenguajes de programación. En el caso de Algol 68, se estableció como un lenguaje de programación, como cuestión de hecho, antes incluso de su implementación real. A partir de ese momento, es historia, como a menudo escuchamos. Algol (1958-1968), uno de los primeros cuatro lenguajes de programación de alto nivel - junto con Fortran (1957), Lisp (1958) y Cobol (1959) - puede asumir una importancia superlativa en comparación, en virtud de su evolución gramatical formal e innovadora, una filosofía de la revolución interna del lenguaje en la informática (BNF a gramáticas-W), ya que, definitivamente, sentó las bases para el fenómeno apropiado de las características heredables de los lenguajes de programación (tipos informáticos) y paradigmas - en relación con expresiones aritméticas, enunciados, bloques y programas, relaciones y condicionales, etc., surgiendo adaptativamente, como un eje especial, tanto de la coevolución de Fortran (1957) como de Lisp (1958), como al extinguirse, influyendo en la especiación de C (1972) y Prolog (1972).

Concluyentemente, y en comparación, una síntesis de programación genética (informacional), impacta significativamente en la codificación (lenguajes de programación), decodificación (intérpretes y compiladores), y más raramente así como en el caso de Algol, en expresiones (gramática formal). A menudo no se ve, y incluso se descuida, lo que es más importante, la regulación (el mensaje del lenguaje de programación con el mensajero) que Algol ha inducido. Hasta el punto de que, en la medida en que los procedimientos recursivos del subprograma fueron parte de Lisp (1957) y Algol (1958), tanto C como Prolog (1972), como Phyton (1990) y Java (1995), comparten un isomorfismo natural-hacia-artificial.'