GÉNESIS Y EVOLUCIÓN DE LA INFORMÁTICA TEÓRICA

RT 84/05

Josep DÍAZ CORT

GENESIS Y EVOLUCION DE LA INFORMATICA TEORICA

1. El Cálculo y la Algorítmica en la antigüedad.

Desde los tiempos antiguos de la humanidad, el desarrollo de los sistemas de numeración ha corrido parejo al de los procedimientos de cálculo. Los algoritmos más antiguos que se conocen son varios procedimientos descritos por los babilonios para resolver ciertas clases de ecuaciones cuadráticas de dos variables y algunos problemas de superficies de triángulos /NE.52/. Estos procedimientos datan aproximadamente del año 2.000 a.C. y se pueden considerar algoritmos genuinos, en el sentido de que en ellos se describen procedimientos generales para resolver determinados problemas específicos. Un ameno estudio de estos algoritmos se puede encontrar en el artículo de **D. Knuth:** "Ancien Babilonian Algorithms" /KN.72/.

También en el antiguo Egipto tenemos ejemplos de toscos procedimientos para resolver algunos problemas específicos, como los que vienen descritos en el papiro de Moscú (aprox. 1850 a.C.), y en el papiro de **Rhind** (aprox. 1650 a.C.). El primero es una colección de problemas geométricos con sus correspondientes soluciones y el segundo contiene un conjunto de reglas para realizar operaciones, que van desde dividir números o solucionar ecuaciones de primer grado, hasta el cálculo de gran variedad de áreas y volúmenes de figuras geométricas. (/MI.68/ cap. 2 y 3), (/NE.79/ vol. 1, cap.2). El inconveniente más grande para considerar como algoritmos los procedimientos descritos en los papiros es que en éstos sólo se dan ejemplos, sin ningún tipo de sistematización, siendo por consiguiente de aplicación muy restringida.

La matemática griega también utilizó procedimientos para resolver problemas, tanto geométricos como aritméticos. Sin duda el algoritmo más conocido de la época y que todavía actualmente es de uso extendido, es el llamado algoritmo de **Euclides** para encontrar el máximo común divisor de dos números. Curiosamente, el algoritmo no es original de **Euclides**, aunque venga descrito en el libro 7, proposiciones 1 y 2 de los Elementos (/MI.68/ cap. 26), puesto que el matemático **Eudoxus**, anterior a **Euclides**, ya lo utilizaba /FA.79/. En la actualidad, dicho algoritmo forma la base de los algoritmos rápidos que existen para el cálculo de m.c.d. (/KN.81/ sección 4.5.2),

funciones recursivas generales. El teorema incluyó un resultado que establece el hecho de que cada función recursiva se puede obtener utilizando únicamente recursión primitiva y una sola aplicación del operador minimización.

En 1936, las dos comunicaciones a que nos hemos referido, de Church y de Kleene, salieron en forma de artículos (16) y (18). El de Church salió antes que el de Kleene y contenía un apartado demostrando la equivalencia de la recursividad vía cálculo-λ y la recursividad de Kleene, referenciando el artículo de Kleene como "de **próxima aparición**". ((16) pp. 353-355). En resumen, Church y Kleene dieron en estos artículos caracterizaciones matemáticas de la noción de funciones efectivamente calculables (calculables mediante algoritmos) y demostraron la existencia de problemas indecidibles, (problemas para los que no existían algoritmos que los resolvieran), aunque sin exhibir ninguno en particular. Esto es lo que hizo **A. Turing**, entrando de lleno en los temas de la Informática Teórica.

En 1936 el británico Alan Turing (1912-1954) /TU.59/ dió de manera independiente de la de Church y Kleene, una nueva formalización del concepto de procedimiento efectivo, utilizando un modelo de máquina general. En su artículo (20) definió los modelos que hoy se conocen con los nombres de **Máquina de Turing** y **Máquina de Turing Universal.** También demostró la indecidibilidad de varios problemas, entre ellos el **problema de parada:** dada una máquina y una entrada, ver si se parará o no. Como Church y Kleene, llegó a la conclusión de la indecidibilidad del **Entscheidungsproblem**, pero esta vez utilizando métodos constructivos. Precisamente en la demostración de la indecidibilidad del problema de la parada, Turing cometió algunos errores, lo que motivó una posterior nota rectificando dichos errores (21). Cuando Turing estaba a punto de enviar el artículo a publicar, se enteró del trabajo realizado por Church y Kleene en Princeton, por lo cual añadió a toda prisa un esbozo de la equivalencia de su computabilidad y los modelos de calculabilidad via cálculo-λ. En un artículo posterior, Turing demostró formalmente la equivalencia entre ambos modelos (22). Según cuenta Randell, la idea del modelo de máquina universal le vino a Turing cuando era un "Fellow" en Cambridge. Un día que asistía a una clase de M.H. Newman, éste presentó la conjetura de Hilbert sobre el **Entscheidungsproblem**, haciendo notar que, como consecuencia, cualquier problema matemático podría ser resuelto utilizando un "procedimiento mecánico". Turing interpretó esta última

Otros pueblos utilizaron ábacos en la antigüedad. Los más conocidos, fueron los de las civilizaciones orientales, especialmente China y Japón. La utilización del ábaco en China viene de muy antiguo. Algunos autores sugieren la posibilidad de que el ábaco pasase de Oriente a Europa, pero no hay suficiente base científica para aceptar sin discusión esta hipótesis. Lo que sí es cierto es que hay constancia escrita de la utilización del ábaco en China, que data de la misma época en que se desarrolló el ábaco entre los griegos. /IF.81/. En el capítulo 8 de una obra anónima, escrita durante la dinastía de los Han (200 a.C. - 100 d.C.) y titulada **"Kien Tchany Souan Chou"**,(El arte del cálculo en nueve capítulos), se explican los detalles sobre algunos cálculos algebraicos realizados mediante el ábaco. En particular, se describen algoritmos para resolver un sistema de n ecuaciones con n incógnitas, detallándose varios ejemplos prácticos, (/IF.81/ pp.130-131). Aquí tenemos una temprana utilización de algoritmos algebraicos, expresados en relación a un instrumento de cálculo.

Volviendo al desarrollo del concepto de algoritmo, en el siglo X durante el califato de **al-Ma'mûn** y bajo su patrocinio, floreció en Bagdad un movimiento científico agrupado en torno a una especie de academia de las ciencias de su tiempo, denominada **"La casa de la sabiduría"**. En particular nos ocuparemos de dos miembros de dicha academia, parte de cuyo trabajo nos concierne. El primero de ellos es **Abû Rayhan al-Bîrunî** (973-1048), filósofo, historiador, geógrafo, lingüista, matemático e informático teórico, ya que uno de sus intereses fue la eficiencia en el cálculo (disminuir la complejidad de los procedimientos de cálculo). Al-Bîrunî mostró, por ejemplo, cómo evaluar de manera rápida el número de granos de trigo colocados sobre un tablero de ajedrez, cuando se ha situado un grano sobre la primera casilla, dos sobre la segunda etc., es decir, cómo evaluar $\sum_{i=0}^{63} 2^i$ de forma rápida. Utilizó para ello una técnica hoy clásica en informática, la técnica de "divide y vencerás" /KN.80/. Al-Bîrunî escribió también un libro sobre las matemáticas conocidas hasta aquel momento, titulado **"Cronología de las naciones antiguas"**, en donde expone un algoritmo para realizar de manera eficiente el cálculo de 2^n, basado en la representación binaria de n, aunque dicha representación como tal no era todavía conocida. El método forma el núcleo de los algoritmos rápidos para la evaluación de potencias. (/KN.81/ pp.441-442). Consignemos que Datta y Singh, en su historia sobre la matemática hindú, mencionan que dicho método ya era conocido por el matemático hindú Pingala en el 200 a.C. /DS.76/.

De todos modos la persona más importante de esta época, desde el punto de vista algorítmico, fue sin duda el también miembro de la "casa de la sabiduría", **Abu Jafar Muhammad ibn Mûsâ al-Khwârizmî** (780-850), (Muhammad, padre de Jafar, hijo de Moses el de Khwârizmî). Polifacético como sus compañeros, tiene contribuciones substanciales en las áreas de astronomía (**"Las tablas astronómicas"**, **"Diseño y utilización del astrolabio"**), geografía (**"Atlas geográfico"**) e historia, ya que también escribió una crónica sobre su época. Pero su contribución más importante se dió en el campo de la algorítmica, y a ésa nos referiremos. La mayor parte del material sobre al-Khwârizmî está sacado del magnífico opúsculo de H.Zemanek /ZE.80/ y a él remitimos al lector interesado en profundizar más en la obra y vida de al-Khwârizmî.

En primer lugar, hay que resaltar dos contribuciones lingüísticas de al-Khwârizmî a la matemática y la algorítmica: la palabra **"algoritmo"**, que deriva de su propio nombre, y la palabra **"algebra"**, proveniente del título de una de sus obras: **"Kitâb al-jabr wa'l-muqâbala"**, (el libro de Aljabr y almuqâbala), (/MI.68/ cap.17). Los filólogos no se ponen de acuerdo sobre la traducción y el significado del título, como, de paso sea dicho, tampoco se ponen de acuerdo sobre la manera de escribir el propio nombre de al-Khwârizmî en caracteres latinos y así /KN.80/, /KN.81/, /ZE.80/ y /MI.68/, entre otros, escriben el nombre de manera diferente. En la presente memoria hemos adoptado la de /KN.80/, que parece ser la más aceptada actualmente. La finalidad del Algebra de al-Khwârizmî no era resumir todo el conocimiento sobre el tema, sino presentar los métodos más útiles y generales para resolver ciertos problemas algebraicos. Descubrió que métodos complicados podían ser substituidos por métodos más simples y sistemáticos, expresando de manera muy clara el concepto y metodología del algoritmo.

Los algoritmos que da para la suma, resta, multiplicación y división de decimales son interesantes, aunque son complicados de llevar a término con lápiz y papel, ya que están llenos de tablas en donde se tiene que borrar y tachar. Asemejan, y probablemente lo sean, adaptaciones de procedimientos utilizados para calcular con ábacos. Citando a Zemanek, se puede decir que:

"el trabajo de al-Khwârizmî es el comienzo de un tipo de abstracción matemática práctica, un tipo de abstracción muy diferente de la griega, que estaba conectada a lo real, mientras que la abstracción de al-Khwârizmî es operacional y orientada hacia un fin concreto, como las abstracciones que se utilizan para hacer funcionar las computadoras de nuestro siglo" /ZE.80/

Dos siglos después de fundarse la casa de la sabiduría, al-Uqlîdisî, un matemático de Damasco, adaptó muchos de los algoritmos de al-Khwârizmî a métodos más apropiados para realizar con papel y lápiz. También expuso de manera más clara el método descrito por al-Bîrunî para calcular de manera rápida 2^n. La obra de al-Uqlîdisî ha sido recientemente traducida al inglés /SA.75/.

Mucho se ha escrito sobre la transmisión del conocimiento matemático del mundo árabe a Europa (/BA.53/ cap.3) /IF.81/. (/KN.81/ sec. 4.1); no obstante, al estar nuestro interés centrado en los fundamentos y desarrollo de los procedimientos de cálculo, restringiremos nuestra exposición únicamente a este aspecto . En el siglo XII, Ramón Llull (1235-?), impregnado de buena dosis de misticismo, fue en peregrinación a Tierra Santa; a la vuelta de dicho viaje decidió dedicar su vida a predicar la doctrina cristiana entre los árabes. Para ello pasó 9 años estudiando la lengua árabe y en algún momento debió entrar en contacto con la obra de los matemáticos árabes, especialmente el lógico Al-Grazzali, al que tradujo al catalán, y probablemente con el Algebra de al-Khwârizmî. Los métodos algorítmicos utilizados por los persas le sugirieron la posibilidad de crear un procedimiento universal capaz de resolver cualquier problema. Tal procedimiento fue descrito en su **"Ars Magna"** (1) /CO.83/. El libro tiene la pretensión de dar un procedimiento general, de base combinatoria, para encontrar todas las verdades. La motivación del libro era religiosa: la búsqueda del algoritmo universal que resolvería todos los problemas y por tanto acercaría a Dios. El transfondo del **Ars Magna**, queda perfectamente reflejado en la siguiente frase de M. Cantor sobre el libro de Llull:

"...una mezcla de lógica, locura cabalística y de su propia locura, en donde han caído, no me imagino cómo, algunos granos de saludable sentido común..." (/HE.69/, pg.27).

Lo magnífico del libro de Llull es la idea de encontrar procedimientos universales para resolver problemas. Esta idea tuvo gran influencia sobre los matemáticos europeos de los siglos siguientes. René Descartes (1598-1650) introdujo la geometría analítica, en donde los problemas geométricos podían ser reducidos a problemas algebraicos y, por tanto, los algoritmos desarrollados para resolver dichos problemas algebraicos podían ser aplicados a la resolución de los problemas geométricos.

Durante toda la Edad Media, los ábacos continuaron siendo utilizados en Europa, pero a medida que la metodología algorítmica se fue introduciendo, comenzó una polémica entre abacistas y algoritmistas. Los que defendían el uso de papel y lápiz contra el uso del ábaco argumentaban que, aunque éste facilitaba mucho la realización de operaciones simples (adición, substracción e incluso multiplicación), era en cambio muy poco conveniente para llevar a cabo operaciones más complejas, que necesitaban de mucho tiempo para su resolución y de gran pericia en el manejo del ábaco, aun cuando se utilizaran procedimientos de una cierta sofisticación, que como hemos visto existían. Reminiscencias de este tiempo son los términos guarismo y algoritmo. Guarismo era la palabra utilizada para denotar los números y algoritmo, los procedimientos efectivos sobre papel y lápiz; ambas palabras derivan del nombre de al-Khwârizmî, como ya se ha apuntado anteriormente. En Europa el ábaco recibió su puntilla con el desarrollo de las máquinas mecánicas de cálculo. La primera de estas máquinas, inventada en 1623 por el alemán **Wilhelm Schickard**, permitía efectuar raíces cuadradas y las cuatro operaciones aritméticas. Todo lo que nos resta de ella es su descripción en una carta de Schickard a Kepler /TF.63/. Poco después, en 1639, **Blaise Pascal** ideó otra máquina mecánica que aún se conserva y que mediante un sistema de engranajes realiza sumas y substracciones. También realiza multiplicaciones, aunque por un lento procedimiento de sumas sucesivas. La historia narra que Pascal ideó esta máquina viendo realizar cuentas con el ábaco a su padre, que trabajaba de contable en el ayuntamiento de Rouen /TF.63/. En síntesis la paulatina substitución del ábaco por algoritmos que permitían la más fácil manipulación simbólica y la introducción de las máquinas mecánicas son las características

definitorias de esta época. Los procedimientos de cálculo basados en instrumentos materiales, originados por el ábaco y desaparecidos a raíz de la caída en desuso de éste, no reaparecieron hasta el siglo XIX con la introducción de las calculadoras mecánicas programadas, como veremos más adelante.

2. De Leibnitz a Frege: dos preguntas y una respuesta.

Quizás una de las figuras más destacadas de la algorítmica y también de la matemática haya sido el alemán **Gottfried Leibnitz** (1646-1716). La importancia de su contribución al desarrollo del análisis matemático es bien conocida. Aquí nos ocuparemos de su contribución a la algorítmica. Como él mismo reconoció, la obra de Llull ejerció gran influencia sobre Leibnitz, especialmente el **Ars Magna.** La lectura de este libro le impulsó a crear en su ciudad -Maguncia- un círculo para estudiar la obra de Llull.

Leibnitz extrajo del **Ars Magna** dos aspectos diferenciados: **el ars iuvendi,** el de un lenguaje simbólico universal; y **el ars indicandi,** el descubrir un procedimiento general que permita deducir si los enunciados, propuestos en el lenguaje universal son o no ciertos. Este planteamiento clarifica el problema fundamental que determinó la génesis y objetivos de la lógica y posteriormente de la informática teórica: (a) Construir un lenguaje formal tan amplio como sea posible, que por supuesto incluya la aritmética; y (b) encontrar algoritmos capaces de decidir si un enunciado cualquiera es cierto o no. Como veremos más adelante, el primer lenguaje formal fue construido en 1878 por **G. Frege** y la parte (b) fue contestada negativamente por los trabajos de **Gödel** y **Turing** en los años 1931-37. Leibnitz también previó la posibilidad de que los algoritmos fueran aplicados de manera "automática", por medio de las máquinas de calcular, de manera que éstas pudieran resolver directamente sin intervención intermedia de operador externo problemas de cierta complejidad. Por otra parte, Leibnitz mejoró la máquina de calcular de Pascal, añadiéndole ruedas dentadas que posibilitaban la realización de productos y divisiones de una manera rápida. (/HT.67/pp.37-44).

/MO.77/ y /TF.63/. A la condesa de Lovelace se la considera la primera programadora, ya que diseñó el lenguaje que había de ser utilizado en la máquina analítica. El programa más complicado que llegaron a diseñar, y del que aún se conservan referencias, fue una rutina para calcular los números de Bernouilli (3).

Los trabajos de Babbage fueron la semilla de los modernos medios de cálculo, que han posibilitado el desarrollo de la informática. La evolución de estos medios de cálculo afecta a la Informática Teórica sólo en la medida en que es su marco técnico de referencia. Sin el aumento de la velocidad de cálculo, que la tecnología de las máquinas ha aportado, no habría tenido ningún sentido práctico el estudio de la eficiencia de los algoritmos, tema sobre el que se centra el objetivo de la Informática Teórica.

Pero cuestiones tales como qué clase de problemas pueden ser resueltos algorítmicamente, y en el caso de los que pueden serlo, cuál es la dificultad inherente a su resolución, no podían ser consideradas en la época de Babbage, ya que hacía falta desarrollar más la lógica para poder plantearlas adecuadamente. Se tenía que comenzar por desarrollar el **ars invendi** de Leibnitz. Fue precisamente en esa época cuando el matemático inglés **George Boole** (1815-1869) dió los primeros pasos hacia una formalización de la lógica. En su trabajo (4) y (5), Boole realiza lo que acostumbra a ser el primer paso de todo proceso de formalización: establece un sistema simbólico que permite operar con una mayor facilidad. En este sentido, prodríamos decir que Boole representa para la lógica lo que 250 años antes Vieta había representado para las matemáticas; intenta dar una expresión matemática a la lógica a base de considerar aspectos algebraicos de la lógica de enunciados. Con todo, Boole realizó un papel de pionero, asentando las bases de lo que después sería la lógica simbólica. Pero no fue hasta fines de siglo que la lógica entró de la mano de **Gottlob Frege** (1848-1925) en el campo de la formalización. Sin olvidar que el período que va de Boole a Frege es rico en trabajos sobre el tema (Jevons, Venn, Mac Coll, Peirce), es Frege quien construye un sistema suficientemente amplio como para proceder a una formalización que se sitúa en la base de todo el desarrollo posterior.

La contribución de Frege a la lógica tiene una significación transcendental desde el punto de vista de la Infomática Teórica. A diferencia de Boole, que toma la

expresión como algo que puede ser realizado por una máquina automática e ideó un modelo abstracto de máquina para poder demostrar que Hilbert estaba equivocado (/MHR.80/ pg.52). En un artículo posterior, Turing definió la **máquina con oráculo**, así como la **reducidibilidad entre problemas** mediante su modelo de procedimiento efectivo y utilizó este concepto para comparar las dificultades de resolución de problemas diferentes (24).

La innovación en este modelo de computabilidad es que define perfectamente lo que es un **paso de computación**, cosa que en los otros modelos, cálculo-λ y funciones recursivas, no está bien definido. El concepto de paso de computación será clave cuando comience a estudiarse la complejidad de las computaciones.

El mismo año que Turing publicaba su modelo de computabilidad, un polaco nacionalizado en América, **Emil Post** (1847-1954), publicaba un artículo con un modelo de computación muy similar al descrito por Turing. El artículo de Post salía con la siguiente nota del editor:

> "El lector deberá comparar este artículo con el de A.Turing "On computable numbers", que saldrá en un próximo número de los **Proc. of London Mat. Soc.** El presente artículo, aunque fechado posteriormente, fue escrito independientemente del de Turing". ((19), pg.1).

En su artículo, Post describe de manera muy clara y concisa (lo que constituye una característica típica de sus escritos) un modelo de máquina muy similar al de Turing y da una caracterización del concepto de procedimiento efectivo también muy similar a la dada por Turing /US.79/.

En un artículo publicado en 1943 (26), Kleene profundizaba sobre los conceptos definidos en su artículo anterior; utilizando la idea de relativización definida por Turing, relativizaba su modelo de funciones recursivas. Una contribución fundamental de este artículo fue expresar cualquier conjunto como una secuencia de cuantificadores seguidos de un predicado recursivo, así si lo que Post definía como conjunto enumerablemente recursivo, se puede definir de la forma $\exists x\ R\ (a, x)$, en donde

deducir, a partir de un sistema arbritrario de axiomas, todas las inferencias que deseemos, siempre que nos mantengamos dentro del lenguje del cálculo de predicados de primer orden. Pero desde el punto de vista informático, el resultado realmente importante de Gödel es su **teorema de incompletitud**, aparecido en 1931 (13). El teorema establece que si el sistema formal desarrollado por Russell y Whitehead en su **Principia** /RW.10/, es consistente existen en él teoremas indecidibles, es decir, enunciados para los que ni ellos ni su negación son demostrables en el sistema. El artículo está escrito con bastante claridad y comienza con una introducción en lenguaje matemático, que explica perfectamente, incluso para el profano, la significación del teorema. De dicha introducción entresacamos las siguientes palabras que expresan con claridad el teorema:

> "Los sistemas formales más extensos que se han constituido hasta el momento son, por una parte, el sistema de **Principia Matemática** y, por la otra, el sistema axiomático de Zermelo-Fraenkel para la teoría de conjuntos. Ambos sistemas son tan extensos que todos los métodos de demostración utilizados en las matemáticas actuales pueden ser formalizados en ellos. Es razonable, por tanto, conjeturar que esos axiomas y reglas de inferencia bastan para decidir todas las cuestiones matemáticas que puedan ser expresadas en dichos sistemas. En el resto del presente artículo, demostraremos que esto no sucede así, sino que en los dos sistemas citados existen problemas relativamente sencillos de la teoría de números que no pueden ser decididos dentro del sistema" /(DA.70) pp.5-6.

La demostración de Gödel está basada en una sabia utilización del clásico argumento de contradicción. El argumento de Gödel se conoce hoy como la paradoja del mentiroso: Se especifica una fórmula F que depende del sistema de reglas y axiomas dados, y tal que la semántica de F puede ser interpretada como que "F no es deducible en el sistema". F no contiene variables libres, por lo que o F es válida o¬F no es válida. Si F es válida, entonces F no es deducible y por tanto el sistema es incompleto. Si ¬F válida y el sistema es completo, entonces¬F debe de ser deducible lo que implica que F no es deducible, y por definición de F también implica que F es válida. Contradicción.

En el mismo artículo, Gödel definió el concepto de funciones **primitivas recursivas** y sugirió pero no demostró soluciones negativas a dos problemas propuestos por Hilbert: el **Entscheidungsproblem** y el de la resolubilidad de las ecuaciones diofánticas (también conocido como el problema décimo de Hilbert) que consiste en determinar si dada un ecuación polinómica con coeficientes enteros, ésta tiene una solución de enteros.

Los resultados de Gödel sobre la incompletitud no fueron establecidos en la forma más general, porque a Gödel le faltaba el concepto explícito de **procedimiento efectivo.** En otoño de 1933, Gödel fue como investigador al "Institut for Advanced Studies" en Princeton. En aquellos días Alonzo Church trabajaba en Princeton sobre el cálculo-λ y su aplicación para definir el concepto de procedimento efectivo. Cuando le comunicó sus planes a Gödel éste opinó que le parecía poco deseable esa formalización, a lo que Church respondió que si Gödel proponía una definición de procedimiento efectivo que fuese medianamente aceptable, él mismo demostraría que dicha definición estaba incluida dentro de la definición via cálculo-λ (/KL.81/pg.59); y así fue como sucedió. Durante la primavera de 1934, Gödel dió un curso en Princeton, del cual Kleene y Rosser, que eran asistentes, publicaron unas notas que habían tomado, bajo el título **"On undecidable proposition of formal mathematical systems"** (15). El curso estaba basado en los resultados previos de Gödel, pero a diferencia de éstos, en vez de utilizar el sistema formal específico de **Principia Matemática**, los generalizó a cualquier sistema formal. Como ya hemos mencionado, para esto era necesario explicitar lo que se entiende por procedimiento efectivo, caracterización que no tenía en 1931, pero que en 1934, utilizando una idea de Herbrand, definió como equivalente a la noción de función recursiva. De hecho, en la edición de las notas de clase Gödel indicó su convencimiento heurístico de que la clase de funciones que se pueden obtener por recursión es la misma que la clase de funciones computables por procedimientos efectivos. (/DA.70/ pg. 44). La vaguedad, escasa formalización , y sobre todo el hecho de que las funciones que Gödel llamaba recursivas correspondan únicamente a las recursivas primitivas, hace que la mencionada frase sólo fuese un preámbulo de lo que después sería la tesis de Church, y no un enunciado equivalente.

Mucho se ha escrito sobre la importancia del teorema de Gödel, sobre todo de cara a la informática. **Nagel** y **Newman**, en su libro sobre el teorema de la

incompletitud, presentaban a éste como la prueba definitiva de la limitación matemática de los computadores. Según los mencionados autores, el teorema de la incompletitud nos indica que en cualquier sistema formal de una cierta complejidad siempre habrá enunciados indemostrables, mientras que el cerebro humano puede llegar a demostrar dichos enunciados, de donde ellos deducen que la estructura y potencia de la mente humana son mucho más complejas que la de cualquier computadora (/NN.70/ pp. 116-121). La primera parte del argumento es válida, el teorema de Gödel demuestra las limitaciones de cualquier sistema formal, y esto incluye todos los lenguajes de programación. Pero de aquí no se puede inferir tan alegremente como lo hacen Nagel y Newman que no se llegará nunca a crear otros tipos de sistemas, que utilicen procedimientos más afines al comportamiento de la mente humana y que por tanto no entren dentro de las hipótesis del teorema de la incompletitud. El área que estudia esta importante cuestión es la Inteligencia Artificial, que queda fuera de la temática histórica que actualmente tratamos. El teorema de la incompletitud dió al traste con el programa de Hilbert, al demostrar la existencia de enunciados dentro de la formalización de la lógica de primer orden, que no eran decidibles. Este resultado fue fundamental en el campo de la lógica, si bien desde el punto de vista de la Informática Teórica lo realmente importante es el **problema de la indecidibilidad:** ver que para todo algoritmo de demostración existe al menos una fórmula cuya verdad no puede ser decidida por dicho algoritmo. Esta sería la respuesta al **ars indicandi** de Leibnitz. En este sentido, la contribución real de Gödel fue el sentar las bases que permitieron la resolución del problema.

El 19 de abril 1935, **Alonzo Church** presentó una comunicación en la reunión del AMS, en donde dió una formalización de las funciones recursivas en términos del cálculo-λ y enunció un resultado fundamental que se conoce con el nombre de la **tesis de Church** y que ya hemos mencionado anteriormente. La tesis de Church identifica el conjunto de funciones recursivas con el conjunto de funciones calculables por medio de procedimientos efectivos (16). En un artículo posterior, Church demostró la indecidibilidad del **Entscheindungsproblem** (17), basándose en el teorema de la completitud de Gödel. Dicha demostración es de naturaleza no constructiva, por lo que en algunos círculos de lógicos de la época -los constructivistas- se cuestionó su validez.

Otra importante contribución teórica de Leibnitz a la informática fue el desarrollo de la numeración binaria. Ya hemos visto como Pingala conocía un algoritmo para calcular 2^n, que implícitamente utiliza la expresión de n en binario, sin que ello quiera decir que fuese consciente de esta utilización. Según **Knuth**, la primera discusión escrita sobre sistema binario de numeración aparece en 1670, en el libro **"Mathesis Biceps I"**, del obispo español **Juan de Caramuel.** En dicha obra, también se discuten otros sitemas de representación de números, pero no se da ningún ejemplo de operaciones matemáticas con sistemas no decimales (/KN,82/ pg.183). Leibnitz, en un artículo para la Academia de Ciencias de París (2), introdujo la adición, substracción, multiplicación y división en binario. En mi opinión, la aportación fundamental de este trabajo de Leibnitz no es el cálculo en binario, que el mismo Leibnitz recomendaba no utilizar para realizar cálculos prácticos, sino su indicación de que la notación binaria permite descubrir propiedades de los números que ayudan a diseñar algoritmos más eficaces (más rápidos y más sencillos).

A principios del siglo XIX, gracias al avance tecnológico que comportó la disponibilidad de materiales y piezas mecánicas cada vez más precisas, el desarrollo de las máquinas de calcular fue progresando. El matemático británico **Charles Babbage** (1792-1871) concibió la idea de construir un instrumento mecánico para calcular e imprimir tablas de funciones matemáticas, al que denominó **"máquina de diferencias"**. Dicha máquina, que quedó como proyecto inacabado, tenía que haber calculado funciones de hasta grado 6, con una precisión de 18 cifras (3). Después de algunos roces con el gobierno británico (/RA.75/ pp.53-65), Babbage perdió interés en este proyecto y comenzó a idear un nuevo ingenio automático: la **"máquina analítica"**, claro precursor de nuestras computadoras actuales. Esta máquina fue concebida como un **"computador universal"**, totalmente automático, es decir, capaz de realizar de manera autónoma cualquier cálculo. Estructuralmente, la máquina correspondía al modelo que hoy se conoce con el nombre de **"arquitectura Von Newman"**. Constaba de una memoria (formada por columnas de ruedas dentadas, con capacidad para 1.000 números con cincuenta posiciones decimales cada uno, una unidad de control (que corresponde a nuestra concepción actual de programa) y elementos de entrada y salida. (/RA.75/ pp 65-86). Mucha de la información que tenemos del trabajo de Babbage nos ha llegado gracias a los escritos de su ayudante, Augusta Ada, condesa de Lovelace (1.815-1.852). Su vida es tan interesante como sus escritos científicos

siendo uno de los procedimientos de naturaleza iterativa más antiguos que se conocen. Es interesante notar que la demostración que dió **Euclides** no es propiamente una demostración, puesto que simplemente verifica que el algoritmo funciona para 3 iteraciones. De hecho es natural que así fuese, ya que el método de inducción no fue descubierto hasta el siglo XVII y por tanto la única manera que tenían los antiguos de demostrar la validez de un algoritmo era el comprobarlo para casos finitos.

Otros aspectos históricamente conectados con el desarrollo de la informática son los sistemas de numeración y los procedimientos de cómputo. Desde tiempos antiguos, el hombre ha utilizado la mano como elemento de representación numérica y también a modo de ábaco para realizar cálculos. **Aristófanes**, hacia el 400 a.C., menciona un curioso sistema para multiplicar y sumar módulo 5 utilizando las manos, que fue desarrollado por los griegos (/IF.81/ cap.3); de aquí deriva el nombre de **cálculo digital**. Sistemas similares fueron desarrollados por otros pueblos primitivos del Oriente y Oriente Medio. Hacia el siglo IV a.C., los griegos pasaron de utilizar los dedos, los nudillos de las manos, a sistemas de cómputo que utilizaban como soporte superficies cuadriculadas; estos fueron los primeros ábacos europeos. Curiosamente, el ábaco de los griegos permitía realizar las operaciones en un sistema muy similar a nuestro sistema decimal (/IF.81/, cap.8), pero el sistema nunca fue adaptado en forma de números escritos, puesto que dada la gran facilidad de realizar operaciones utilizando únicamente el ábaco, o la mano en su defecto, nunca sintieron la necesidad de utilizar papel y lápiz para realizar sus sumas y multiplicaciones.

Los romanos continuaron utilizando el ábaco de piedras: ("abacus calculi"), de donde deriva la palabra cálculo. Inventaron el primer ábaco de bolsillo: una pequeña placa de metal provista de ranuras paralelas por donde se deslizaban unas cuentas del mismo tamaño (/IF.81/ pg.121). Que yo sepa, ésta fue la primera calculadora de bolsillo. Destaquemos que entre los romanos la palabra "calculator" designaba por una parte a los maestros del cálculo, cuya tarea principal consistía en enseñar a la gente a calcular por medio del ábaco (precursores de los profesores de informática), y por otra parte se utilizaba para designar a los contables que existían en las casas de los grandes patricios. Como veremos más adelante, la utilización del ábaco en Europa continuó hasta bien entrada la Edad Media.

Cuando Church expuso su tesis, el americano **Stephen C. Kleene** se propuso demostrar su falsedad por medio de una diagonalización sobre toda la clase de funciones definibles por medio del cálculo-λ, pero al apercibirse posteriormente de que dicha diagonalización no podía ser llevada a término de manera efectiva, cambió de opinión, para pasar a ser un firme defensor de la afirmación de Church. Curiosamente, el nombre de **Tesis de Church** fue utilizado por vez primera en un libro de Kleene ((39) cap.XII).

En septiembre de 1935, Kleene presentó una comunicación (18), en donde definía formalmente el concepto de función recursiva, utilizando un número finito de ecuaciones. El mismo Kleene explica la génesis de su formalismo:

> "Mi estudio sobre las funciones definidas por medio del cálculo-λ comienza a partir del curso que A.Church dió en Princeton en enero de 1932. En este curso, Church presentó las ideas básicas de su sistema formal lógico, mediante la utilización del cálculo-λ. A partir de dicho curso yo comencé un proyecto de tesis doctoral consistente en desarrollar la teoría de la aritmética utilizando el sistema lógico de Church. El punto central del proyecto era demostrar los axiomas de Peano, utilizando el formalismo de Church". (/KL.81/ pg.56)

Kleene llevó a término con éxito su tesis doctoral y parte de ella fue expuesta en la mencionada comunicación. Uno de los puntos que más le costó fue el definir la función predecesor en términos del cálculo-λ. (/KL.81/ pg.56). Kleene diferenció entre la clase de **funciones primitivas recursivas**, (obtenidas a partir de las funciones básicas cero, sucesor e identidad, por aplicación de los esquemas de composición y recursión primitiva) y la clase de **funciones recursivas generales** (obtenida a partir de la anterior, mediante la introducción del operador minimización).

También en esta comunicación, Kleene demostraba la existencia de conjuntos indecidibles, (conjuntos que no eran reconocidos mediante procedimientos efectivos) y definía la clase de los conjuntos **enumerables recursivamente** (conjuntos que se pueden enumerar, no necesariamente en orden, mediante procedimientos efectivos). Finalmente, Kleene demostró en el artículo el **Teorema de la Forma Normal** para las

/RE.70/. Hacia 1910, Hilbert enunció lo que hoy se conoce con el nombre de **programa de Hilbert**, que fundamentalmente consiste en formalizar las teorías axiomatizadas mediante un lenguaje de primer orden y tratar de demostrar su consistencia. En particular, partiendo de cualquier axiomatización de la aritmética, como la de Peano, hay que determinar que las matemáticas están libres de contradicción, es decir, que si un enunciado es un teorema entonces su negación no puede serlo. Otra parte del programa amplía el problema a determinar si un enunciado matemático es una consecuencia lógica de un conjunto de premisas, que a su vez pueden ser deducidas de axiomas, utilizando únicamente los símbolos de la lógica. Hilbert y sus seguidores consideraron este problema de decidir si un enunciado es o no es un teorema en la lógica y lo denotaron con el nombre de **Entscheidungsproblem.**

3. De Gödel a Kleene: La toma de identidad de la Informática Teórica.

Hemos visto como en el período anterior se daba respuesta al lenguaje simbólico universal de Leibnitz. Una vez creado este lenguaje formal y utilizado conjuntamente con un sistema axiomático para definir teorías, surgen inmediatemante dos preguntas: la primera hace referencia a la completitud de una teoría, es decir si todo enunciado de la teoría es o cierto o falso. La segunda pregunta alude a la decidibilidad de una teoría, esto es, a la existencia de un algoritmo capaz de decidir para cada enunciado de la teoría si éste es cierto o falso. Esta segunda cuestión coincide con los propósitos del **ars indicandi** de Leibnitz y conecta con el núcleo conceptual de la Informática Teórica. Aunque los trabajos realizados por Frege, Boole, etc. pertenecen a la historia de la lógica, también fueron seminales, por las razones ya vistas, para el desarrollo de la Informática Teórica, y por este motivo los hemos mencionado. Los trabajos posteriores que formaron lo que se conoce con el nombre de **teoría de la recursividad** y que tiene como punto de partida la obra de **K. Gödel**, dieron identidad propia a la Informática Teórica.

En el año 1930, el austríaco **Kurt Gödel** (1906-1978) publicó su tesis doctoral en la Universidad de Viena: **"Die Vollstän digkeit des Logikkalkūls"** (12). En el mencionado trabajo, Gödel demostraba la completitud del cálculo de predicados de primer orden. Este resultado viene a decir que las reglas de la lógica son suficientes para

formalización del álgebra (en aquella época todavía en un estado rudimentario), como punto de partida para su simbolización, Frege se propone la construcción de una formalización total de la lógica precisamente para poder fundamentar correctamente las matemáticas en este sistema. Esta formalización viene expresada fundamentalmente en su artículo de 1879 **Begriffschrift** (6) y se puede considerar como el primer **lenguaje formal.** Es la respuesta al lenguaje universal de Leibnitz. Con el trabajo de Frege se abre por vez primera una problemática que más adelante afectará de pleno a la Informática Teórica. Problemas como los de la **completitud** (si todos los enunciados son demostrables) o de la **decidibilidad** (si existen algoritmos para demostrar los enunciados), que son punteros en teoría de la calculabilidad, solamente pueden ser planteados en sistemas totalmente formalizados. El tratamiento dado a estos problemas en la lógica se extiende de manera inmediata a los lenguajes formales estudiados en programación y teoría de lenguajes. Las insuficiencias e incorrecciones de tipo formal que padece la obra pionera de Frege no quitan a éste su debido reconocimiento, aunque durante algunos años fuera totalmente ignorado por sus contemporáneos. **B. Russell** fue quien divulgó y dió la consideración que merece a la obra de Frege. Su libro **Principia Matemática** (9) está basado en ella, afirmación ratificada por el propio Russell, que fue la primera persona en leer el **Begriffschrift** veinte años después de su publicación (/RU.20/ pg.25).

Paralelamente a Frege, el matemático italiano **Giuseppe Peano** (1858-1932) escribió cinco volúmenes titulados **Formulaire des Mathèmatiques,** en donde axiomatizó la artimética de primer orden, es decir, enunció tres conceptos y cinco axiomas básicos de donde se puede deducir toda la teoría de los números naturales (7) y (/RU.20/ cap.1). Aunque aparentemente el trabajo de Peano pertenece más a la lógica que a la Informática Teórica, éste fue el modelo que **Kleene** eligió de cara a su formalización de la teoría de la recursividad.

El período de la historia de la lógica que estamos describiendo se caracteriza por la creación de otras axiomatizaciones de la aritmética, como por ejemplo la de **Zermelo - Fraenkel.** Una escuela de pensamiento lógico, la de los **formalistas,** se dedicó a formalizar las teorías axiomatizadas para así poder demostrar la **consistencia** (ausencia de contradicciones) de los enunciados de cada teoría. El máximo impulsor de dicha escuela fue el lógico alemán **David Hilbert** (1826-1943)

R (a, x) es un predicado recursivo. Utilizando esta formalización, Kleene definió una jerarquía de predicados, que se conoce con el nombre de **"Jerarquía de Kleene o Jerarquía Aritmética"**, puesto que contiene todos los enunciados de la aritmética. Esta jerarquía comienza con los predicados recursivos; y dado un conjunto, éste se clasifica en el nivel n de la jerarquía, si se puede expresar como una secuencia de n cuantificadores alternados delante de un predicado recursivo. En cierta manera, la jerarquía clasifica los conjuntos respecto a su nivel de indecidibilidad.

Al año siguiente Post, en un magnífico artículo (29), profundizó en el estudio de los conjuntos enumerablemente recursivos. En este artículo, Post define el concepto de **reducibilidad - m** y el de **problemas completos**, es decir, los problemas "más difíciles" de una clase (en cierta manera, Kleene ya había intuido el concepto, ((18) teorema XIII, pg.740). También define Post el concepto de **reducibilidad - tt** (tabla de la verdad) y compara las reducibilidades m, tt, y la T que es la introducida originalmente por Turing. Post demuestra la existencia de conjuntos simples, es decir, conjuntos enumerables que no son ni recursivos ni m-completos. Finalmente, en dicho artículo, Post propone un problema que se conoce con el nombre de **Problema de Post:** la existencia o no existencia de conjuntos que sean enumerables recursivamente, pero no sean ni recursivos ni T - completos. El problema fue resuelto en sentido positivo por el soviético **A. Muchnik** en 1956 (46). Independientemente, al año siguiente el americano **R. Friedberg** también llegó al mismo resultado (47). Tanto Muchnik como Friedberg utilizaron básicamente la misma técnica, que hoy se conoce con el nombre de **"método de prioridad"** y ha pasado a ser una técnica clásica en teoría de la recursividad y teoría de la complejidad abstracta.

En un artículo posterior (32), Post demostró por vez primera la indecidibilidad de un problema clásico de las matemáticas, propuesto por **Axel Thue** en 1906: el **problema de las palabras.** Dadas dos palabras de un alfabeto y una lista de ecuaciones entre palabras del alfabeto, el problema consiste en determinar si se puede derivar una palabra de la otra mediante las substituciones dadas por las ecuaciones (8). Church ya había conjeturado la indecidibilidad de dicho problema. El método utilizado por Post es una reducción del problema de la parada a dicho problema, en el sentido de que existe una solución para el problema de Thue si y solo si existe una solución para el problema de la parada. Recordemos que Turing ya había demostrado la

irresolubilidad de dicho problema. En un apéndice, Post dirije una fuerte crítica al artículo "On computable numbers" de Turing, señalando que el modelo de máquina allí propuesto es incompleto y revelando la existencia de errores en algunas demostraciones. Post completa la definición de Turing y rectifica las definiciones. Es necesario hacer notar que, mientras los artículos escritos por Turing son de difícil lectura, los de Post como ya hemos mencionado son muy fáciles de leer. En el mismo año 1947, el soviético A. Markov, de forma independiente de Post, también demostró la indecidibilidad del problema de Thue, utilizando un nuevo modelo de computación equivalente a los vistos anteriormente (31).

Al año siguiente, Post definió los grados de irresolubilidad de los conjuntos. Expresando éstos como predicados, estableció que un predicado P es del mismo grado que Q si y solo si P es T-reducible a Q. Un **grado de irresolubilidad** se define como la clase de todos los conjuntos que tienen el mismo grado (34). Los grados de Post dan una estructura más refinada que la jerarquía aritmética, en el sentido de que cada nivel de la jerarquía se puede subdividir en subclases de conjuntos con igual grado. (42).

Quisiera mencionar aquí una curiosa historia sobre Post, que indica hasta qué punto se llegó de forma independiente a las mismas conclusiones y cómo la máquina de Turing o la tesis de Church no son conocidas por otros nombres por cuestión de meses. En 1941 Post escribió un artículo, cuya publicación fue realizada y no apareció hasta 1970, en el libro antológico de M. Davis /DA.70/. El artículo relata los resultados conseguidos por Post en la década de los veinte, cuando en EE.UU existía un desinterés total hacia la lógica. Durante esos años, Post anticipó y demostró los resultados que más tarde demostrarían Gödel, Church y Turing. Post basó sus resultados sobre una hipótesis equivalente a la que después sería la tesis de Church y durante largo tiempo concentró sus esfuerzos en la verificación de dicha hipótesis, que para él consistía en un análisis del proceso mental que se realiza al resolver problemas combinatorios. Tenía el propósito de publicar su trabajo cuando estuviera totalmente finalizado, pero en el proceso, los artículos de Gödel y Church salieron a la luz. El estado de ánimo de Post en el año 41 queda puesto de manifiesto en las primeras líneas del mencionado artículo:

"Existiría poco interés en publicar los resultados anticipados del autor sobre la existencia de problemas indecidibles en el sentido de Church y, como corolario, el resultado de Gödel sobre la incompletitud de la lógica simbólica, si únicamente se hiciese como reivindicación de un reconocimiento no oficial de prioridad,.." (/DA.70/ pg.340)

Su vida personal estuvo plagada de desgracias. La falta de reconocimiento científico que sufrió su trabajo los primeros años, le obligó por ejemplo a ganarse la vida como profesor en una escuela secundaria de Nueva York. También sufrió a lo largo de su vida constantes ataques nerviosos, que le llevaron repetidas veces a ser internado en un hospital psiquiátrico.

Durante el mismo período de tiempo, paralelamente al desarrollo ya descrito de los conceptos teóricos de computación y de algoritmo, tuvo lugar un desarrollo de los instrumentos de cálculo que fue desde la máquina de cálculo mecánico tipo Babbage hasta el primer ordenador electrónico. Aunque desde el punto de vista de la Informática Teórica, el desarrollo de los ordenadores es en cierto modo marginal, daremos una breve referencia histórica del mismo.

A partir de la máquina de Babbage, se dió una rápida evolución de las calculadoras mecánicas hasta llegar a la utilización de relés y mecanismos electromecánicos que substituyeron a los engranajes de las máquinas mecánicas. Una persona que mencionaremos aquí un poco a título anecdótico es **Leonardo Torres Quevedo** (1852-1936), quien llegó a construir algunos modelos de calculadores electromecánicos que fueron innovadores en su tiempo (11). Diseñó además un "autómata" con la pretensión no sólo de imitar los movimientos del hombre, sino también todas sus acciones, pudiendo llegar a reemplazarle ((10), pg.87). Torres Quevedo pensaba utilizar el lenguaje natural para programar su hipotético autómata, que nunca llegó a realizarse. Lo que sí construyó fue una serie de autómatas particulares que realizaban funciones específicas, el más conocido ejemplo de los cuales es uno para jugar al ajedrez, que aún se conserva.

En 1937 **G. Stibitz**, un matemático que trabajaba en los laboratorios Bell, contruyó un sumador decimal utilizando relés, que dió pié al diseño y construcción del

"Complex Calculator", una máquina que trabajaba en binario y realizaba aritmética de números complejos, con una biblioteca de programas almacenada en cintas magnéticas. Poco después la universidad de Harvard y la compañía I.B.M. colaboraban en un nuevo modelo de calculador electromecánico: el Mark I. El padre intelectual de dicho calculador fue el profesor de Harvard **Howard Aiken,** que como él mismo admite en sus escritos, recibió una cierta influencia de Torres Quevedo. Tanto la máquina de Stibitz como la de Aiken utilizaron los trabajos de **Claude Shannon** sobre la implementación de cálculo binario mediante circuitos de conmutación. En 1937, mientras estudiaba en el MIT, C. Shannon escribió una tesis sobre la aplicación de la lógica simbólica al diseño y análisis de circuitos de conmutación (23). En subsiguientes trabajos Shannon introdujo una idea que, como veremos más adelante, dió paso a una de las medidas de complejidad más útiles: el **coste Booleano.** La idea de Shannon consistió en implementar, mediante circuitos de conmutación, los conectivos booleanos Λ, V, y, a partir de aquí, implementar ciertas funciones expresándolas como combinaciones de estos conectivos (36).

Por otra parte, en Europa, el alemán **Konrad Zuse,** sin el apoyo monetario e institucional de que disfrutaban Stibitz o Aiken, fabricó en 1941 un calculador programable universal, el Z3, que se puede considerar equivalente al Mark I, pero más pequeño y rápido. Zuse se propuso construir esta máquina en versión electrónica, pero el veto de la Alemania Nazi se lo impidió; de no ser así, su máquina hubiese sido el primer calculador electrónico. También Zuse amplió las ideas de Babbage sobre la programación en Europa durante la década de los cincuenta. El primer ordenador electrónico fue el COLOSUS, desarrollado en Gran Bretaña durante los años (1940-1944). Hasta 1973 su existencia fue desconocida excepto para un grupo muy pequeño de gente, al estar considerado secreto militar. Uno de los científicos que trabajó en dicho proyecto fue A. Turing. Durante ese tiempo, Turing también desarrolló sus ideas sobre máquimas inteligentes y algoritmos heurísticos, como la regla **minimax** para exploraciones de árboles de búsqueda. En 1947 Turing escribió todos los conceptos sobre "máquinas inteligentes" desarrollados durante dicho período, pero no fueron publicados hasta 1969 (33). Como ya hemos indicado no es nuestro propósito profundizar sobre el tema de la evolución de los ordenadores, tema sobre el que por otra parte existe una amplia literatura. Abundante información sobre el desarrollo de los ordenadores electrónicos se encuentra en los libros de Randell /RA.75/ y de

Metropolos /MHR.80/; éstos incluyen artículos originales y reflexiones a posteriori sobre el tema, muchas veces escritos por los propios protagonistas. Los artículos de **A. Sales**, sobre el desarrollo de las máquinas de cálculo y sobre el desarrollo de los ordenadores electrónicos /SA.80/, contituyen una introducción muy amena al tema. Una revisión técnica excelente del desarrollo de los ordenadores electrónicos, especialmente en lo que refiere a la evolución de los diseños de las arquitecturas de ordenadores, es la expuesta por M. Valero /VA.83/. Respecto al desarrollo de los lenguajes de programación, el artículo de J.Backus. "Programming in America in the 1950's" (/MHR.80/ pp.125-136), el artículo de Ershov y Shura-Bura: "The early development of programming in the USSR" (/MHR.80/ pp.137-196), y el mencionado de Knuth y Trabb /KY.80/, constituyen magníficas revisiones. También los artículos de F. Saltor /SA.75/ dan una visión global de la evolución de algunos lenguajes de programanción.

Otro campo que en los años cuarenta cobró cierta importancia, y que más tarde tendría una relevancia fundamental para la Informática Teórica, es el de la lingüística computacional. En esos años la escuela americana de lingüistas: Bloch, Harrir, Trager y otros, utilizaron la noción de análisis por medio de "**immediate constituents**" (IC) como medio de analizar la morfología y la sintaxis del lenguaje natural. Para analizar una frase, se aislan los IC que contiene, después cada IC se rompe a su vez en IC y así sucesivamente. A partir de este punto, parte de los esfuerzos se dirigieron hacia la formalización matemática de la noción de IC. En 1953 **Bar-Hillel**, utilizando los trabajos previos del lógico polaco **Kasimir Ajdukiewicz**, ideó como formalización del análisis IC las "**gramáticas categóricas**" (cathegorial grammars), las palabras y cadenas de palabras son asignadas a categorías, las cuales pueden ser de dos tipos: categorías básicas y categorías operacionales. A partir de ellas define unas reglas de derivación, para la formación de frases (39).

Como dice Greibach:

> "Las gramáticas categóricas de Bar-Hillel son el primer ejemplo de un sistema matemático capaz de generar lenguajes formales" /GR.81/.

Este trabajo también fue el comienzo de la traducción automática, tan de moda en la década de los 60.

Otra nueva área informática que comenzó a expandirse en esta década fue la cibernética. Descartes ya había estudiado los movimientos del cuerpo humano simulándolos por medio de ecuaciones diferenciales. Desde entonces, a lo largo de los años, se había intentado, conseguir mejores simulaciones; pero es con la colaboración en 1943 entre el matemático **N. Wiener** y el neurocirujano **A. Rosenblueth**, que esta nueva área recibe un impulso real. La elección del término "**cibernética**" es, según palabras del propio Wiener, un reconocimiento explícito al primer artículo serio sobre mecanismos de autorealimentación, escrito en 1868 por **J. C. Maxwell** sobre timoneles automáticos (en griego la palabra cibernos significa autotimonel) (35). La cibernética estudia el control y la comunicación en el animal y la máquina, poniendo énfasis en la retroalimentración y es esta característica la que la diferencia fundamentalmente de los estudios previos. La cibernética en general no tiene relación directa con el tema que nos ocupa, ni en sus fines ni en sus medios (series temporales, ...). Pero hay un aspecto que sí tuvo gran relevancia para la Informática Teórica. En 1942 el neurofisiólogo **Warren Mc Culloch**, interesado en problemas de simulación del cerebro humano, se puso en contacto con el grupo de Wiener en el MIT. Poco después, conjuntamente con el lógico **Walter Pitts**, proponía un modelo discreto de red neuronal para simular la actividad cerebral (27). Desde un punto de vista neurofisiológico, el modelo de Mc Culloch y Pitts es muy simplista. Pero para nosotros, lo verdaderamente importante de dicho modelo es que fue el primer **autómata finito**, (autómata en el sentido informático teórico, es decir, como reconocedor de lenguajes). En 1951 S. Kleene pasó el verano en la RAND Corporation, en donde le dieron el artículo de Mc Culloch y Pitts para que lo estudiase y viese qué podía sacar de él. Kleene formalizó el concepto de autómata finito y definió los **conjuntos regulares**, como la clase más pequeña que contiene los conjuntos finitos y es cerrada para concatenación y estrella. En el artículo original, en lugar de considerar cadenas sobre un alfabeto, Kleene considera tablas de patrones etiquetados como iniciales o no iniciales. Una tabla $a_1 \ldots a_n$, representa $a_n \ldots a_1$, con un a_n en tiempo 1 (primera posición); si a_n es inicial; en el caso de que sea no inicial, la tabla representa $\ldots a_n \ldots a_1$, sin especificar la colocación de a_n respecto al tiempo. Un "suceso" viene representado por un conjunto de tablas. La concatenación de sucesos E.F es

representada por el conjunto de tablas (X, Y), donde X es una tabla no inicial en E. e Y está en F. La clase de sucesos regulares se define como la familia más pequeña que contiene los sucesos unidad (el conjunto vacío) y que es cerrada respecto a la unión, la concatenación y una operación que definió Kleene E * F y que a partir de entonces se denominó **estrella de Kleene** (37). En el mismo artículo, Kleene establece la equivalencia entre "sucesos regulares" y autómatas finitos.

A partir de este punto, las ideas de la teoría de funciones recursivas, de la lingüística y del reconocedor de Kleene, se entrelazarían constituyendo el campo de la Informática Teórica, como veremos a continuación.

2.4. De Chomsky a Hartmanis : Los años de formación

Al principio de los años 50, un joven estudiante llamado **Noam Chomsky** repartía su interés entre la lingüística y la defensa del sionismo. Tras estar dudando entre irse a Palestina o continuar con su tesis doctoral, decidió esto último, presentándola en el MIT en 1955 (43). Parece ser que la influencia de Bar-Hillel, muy integrado también en el movimiento sionista, fue decisiva para que Chomsky tomase la decisión de escribir la tesis, incluso le indicó que leyese los trabajos de E. Post e intentase aplicar sus resultados al campo de la lingüística computacional (NW.80). En su tesis, Chomsky ofrecía un primer esbozo de su jerarquía de lenguajes, dando tres teorías diferentes de modelos lingüísticos, con énfasis en los aspectos generativos de las gramáticas. Esta parte de la tesis salió publicada como artículo al año siguiente (44). En artículos posteriores (51) y especialmente (57), definió la jerarquía en su forma actual, donde una gramática sobre un alfabeto finito Σ se expresa como un cuádruple (V_T, V_N, P, S), en que V_N representa el conjunto de variables no-terminales, V_T el de variable terminales, S la variable no - terminal inicial y P es el conjunto de reglas de producción de la gramática. Con esta nomenclatura, Chomsky distingue los siguientes tipos de lenguajes:

Tipo 0: Lenguajes generados por gramáticas sin restricciones.

Tipo 1 : Lenguajes generados por gramáticas con producciones del tipo:

$$u\ A\ v \rightarrow u\ y\ v,$$

donde A no es vacío y pertenece a V_N.

Tipo 2: Lenguajes generados por gramáticas con producciones del tipo $A \rightarrow y$, donde y no es vacío.

Tipo 3: Lenguajes generados por gramáticas con producciones del tipo $A \rightarrow a\ B$ o $A \rightarrow a$.

En un artículo de 1958, Chomsky y Miller identificaron los lenguajes tipo 3 con los lenguajes regulares definidos por Kleene (52).

La generación de lenguajes por medio de sistemas de reescritura había sido formalizada por Post (29), quien posteriormente estableció la equivalencia de estos lenguajes con los conjuntos enumerables recursivamente (32). Chomsky, en el mencionado artículo (57), demostró las inclusiones estrictas de las cuatro clases de la jerarquía. La inclusión estricta de los lenguajes tipo 1 en los tipo 0; se deduce a partir de los resultados previos de Kleene y Post; la inclusión estricta de los de tipo 2 en los de tipo 1 la demostró utilizando el lenguaje $\{a^n\ b^m\ a^n\ b^m \mid m, n > 1\}$; y para demostrar que los de tipo 3 están incluidos estrictamente en los del tipo 2, utilizó el lenguaje $\{a^n\ b^n \mid n > 1\}$. También demostró que todos los lenguajes del tipo 2 pueden ser generados por gramáticas que únicamente contengan reglas de producción del tipo $A \rightarrow b$ (b un terminal) o del tipo $A \rightarrow BC$ (B y C no terminales), lo que implica que las gramáticas tipo 2 no son más potentes que las gramáticas categóricas utilizadas por Bar-Hillel en el análisis IC. En 1961 se demostraría que ambos tipos de gramáticas generan la misma familia de lenguajes (61). A la anteriomente mencionada forma canónica de expresar las gramáticas tipo 2 se la conoce como la **Forma Canónica de Chomsky.**

En la sección anterior vimos que Kleene había definido los conceptos de autómata finito y expresiones regulares, basándose en los trabajos de Mc Culloch y Pitts, los cuales a su vez habían recibido una gran influencia de los trabajos de Shannon y su grupo sobre el análisis de circuitos de conmutación. La influencia de la teoría y diseño de los circuitos de conmutación sobre el desarrollo de la teoría de autómatas continuó a lo largo de la década de los 50. **Huffman,** en su tesis doctoral presentada en el MIT en 1954, sobre el tema de la síntesis de circuitos de conmutación, definió un modelo de **máquina secuencial** muy similar al modelo de

autómata de Kleene; ni los trabajos de este autor ni las redes neuronales vienen referenciadas, lo que induce a pensar que Huffman diseñó su modelo sin prestar atención a los modelos previos. En el mismo artículo se expone un algoritmo para la minimización de circuitos secuenciales, que es la base de los algoritmos utilizados en la actualidad para la minimización de autómatas. Dos años más tarde **E. Moore**, también de MIT, definía otro tipo de autómata, independiente pero equivalente al de Huffman; este nuevo tipo de autómata tenía la particularidad de dar el resultado en forma de ceros y unos (46). En el mismo artículo, Moore refinaba el algoritmo de minimización de Huffman. El artículo de Moore tiene la particularidad de utilizar diagramas de transición para representar los autómatas, con círculos representando los estados y flechas representando las transiciones, tal y como se utiliza en la actualidad. En 1957, **Myhill** caracterizó los **lenguajes regulares**, definidos por Kleene, utilizando relaciones de congruencia de índice finito (49) y, al año siguiente, **Nerode** dió una nueva caracterización de los lenguajes regulares en términos de relaciones de equivalencia invariantes por la derecha (56).

La caracterización de Nerode es utilizada en la actualidad para presentar de manera elegante el teorema de minimización de Moore. En el mismo año, **Copi, Elgot y Wright** daban una versión clara del artículo de Kleene sobre autómatas, redefiniendo las expresiones regulares, sin utilizar el concepto de tablas etiquetadas, como la clase mínima de conjuntos que es cerrada para la unión, concatenación y estrella de Kleene (53). En este artículo se formaliza la noción de la estrella de Kleene como operación unaria sobre expresiones regulares. En 1957, **M. Rabin** introdujo el modelo de autómata bidireccional (50), en el que el cabezal puede leer hacia la derecha y hacia la izquierda. En 1959 Shepherdson (60) demostraba la equivalencia entre los autómatas bidireccionales y los unidireccionales, equivalencia en el sentido de que ambas reconocen la misma clase de lenguajes. La idea de la demostración es una ingeniosa reducción de los bidireccionales a los unidireccionales. También utilizaba por vez primera el **problema de la correspondencia de Post** que ya se sabía que era indecidible (30), para demostrar la indecidibilidad del problema de determinar si es vacía o no la intersección de dos lenguajes generados por dos autómatas finitos de dos cintas. En ese mismo año se publicó un artículo de **Rabin** y **Scott** (59), que en mi opinión es uno de los más importantes que se han publicado en Informática Teórica y probablemente también sea uno de los más referenciados. En este artículo se introduce la noción de

no determinismo, via el modelo de autómata no determinista. La idea ya había sido intuida por **Chomsky** y **Miller** (52). La noción de no determinismo, como iremos viendo, es fundamental en Informática Teórica, ya que, contrastando con el modo hasta entonces vigente, y que hoy denominamos determinista, de concebir el funcionamiento de los autómatas, consistente en utilizarlos para **averiguar** el cumplimiento de alguna propiedad, **Rabin** y **Scott** utilizan autómatas para **comprobar** que la propiedad se satisface. Así, para averiguar si la n-ésima cifra del final de una palabra binaria es un cero, se requiere un autómata de 2^n estados (los necesarios para memorizar los 2^n sufijos posibles), en cambio para comprobar meramente este hecho, sólo se requiere un autómata de $n + 1$ estados (los necesarios para contar, a partir de una cierta posición previamente elegida, las posiciones restantes). En cierta manera, es una idea semejante al oráculo de la máquina de Turing. En el artículo, también se demuestra la equivalencia entre las clases de lenguajes aceptados por los modelos determinista y no determinista de autómata finito.

En 1960, **Mc Naughton** y **Yamada** (62) dieron sendos algoritmos rápidos para transformar cualquier expresión regular en un autómata finito, y viceversa.

Volviendo a la jerarquía de **Chomsky**, ya hemos visto el desarrollo de los lenguajes tipo 3 (regulares) y tipo 0 (enumerables recursivamente).

En 1960, **Scheinberg** (65) demuestra que los lenguajes de tipo 2 son cerrados para la unión, pero no bajo intersección o complementación. También demuestra, utilizando por vez primera argumentos tipo bombeo, que $\{a^n b^n c^n \mid n > o\}$ no es tipo 2. En este artículo, Scheinberg denota a los lenguajes tipo 2 como **Context-free** (CF). **S. Greibach** afirma que este es el primer artículo en donde aparece la denominación context-free (GR.81). Al año siguiente, en un artículo de **Bar-Hillel, Perles** y **Shamir** se demuestra una serie de propiedades de los lenguajes CF, que los autores denominaban **"simple phrase structure"**; entre ellas, que la clase de CFL eran indecidibles, via una reducción del problema de la correspondencia de Post (66). En este artículo también enunciaban de manera generalizada el **teorema "uvwxy"**, también conocido como **lema del bombeo**, que establece una condición necesaria, pero no suficiente, para que un lenguaje sea CF y, por lo tanto, es una potente herramienta para demostrar que un lenguaje no es CF.

En el mismo año, un manuscrito de **Parikh**, que no sería publicado hasta 1966, establecía la existencia de CFL inherentemente ambiguos, demostrando que: $\{a^n b^m a^k b^r \mid n = k \text{ o } m = r, m, n, k, r > 1\}$ no puede ser generado por gramáticas CF no ambiguas (67).

Otra manera de estudiar los CFL fue la utilización de series de potencias, **Schützenberger** (68) y (74) estableció una formulación algebraica para expresar las familias de CFL y los lenguajes regulares.

Este punto de vista, ha dado lugar a los trabajos más matemáticos dentro de la teoría de lenguajes formales.

Al mismo tiempo que se desarrollaba la teoría de las gramáticas incontextuales, también se desarrollaban otros sistemas de potencia equivalentes utilizados para la especificación y el análisis sintáctico de lenguajes naturales (orientados a la traducción automática) y de lenguajes de programación. Como ejemplo, cabe citar los trabajos de **Backus** (57) y **Naur** (64), que utilizan un conjunto de fórmulas para especificar la sintaxis del Algol 60; a partir de entonces, este tipo de especificación se conoce como **Forma de Backus-Naur.** No es el propósito de esta exposición realizar una recopilación detallada de los lenguajes de programación sino únicamente dar el adecuado marco de referencia al desarrollo de la jerarquía de Chomsky. Para una descripción más detallada de la interacción entre el desarrollo de las gramáticas incontextuales y el de los sistemas de traducción automática, tanto de lenguajes naturales como de programación, consúltese (/ER.81/ pg.19-22). A modo de síntesis podemos decir que, en el período de tiempo que nos ocupa, los lingüistas e informáticos propusieron toda una gama de sistemas de generación y análisis sintácticos:

- Análisis de los constituyentes inmediatos.
- Gramáticas categóricas.
- Lenguajes tipo 2 (CFL).
- BNF (Sintaxis del Algol).
- Análisis Predicativo.
- Autómatas de pila.

Como hemos visto, ya en 1962 se conocía la equivalencia formal de algunos de estos sistemas de generación y **Gross** en un artículo escrito en ese año, pero no publicado hasta dos años más tarde, conjeturaba la equivalencia de todos los sistemas conocidos en el campo de traducción mecánica (73). La equivalencia de BNF y de los lenguajes tipo 2 la demostraron en 1962 **Ginsburg** y **Rice** (72).

La idea del autómata de pila aparece en un trabajo de **Yngve** de 1961 sobre traducción mecánica de programas (70). La equivalencia entre lenguajes aceptados por **autómatas de pila no deterministas** y los lenguajes incontextuales se debe a **Chomsky** y **Schützenberger** (75), (71) y (74). En 1963 **R. J. Evey**, da una demostración de la equivalencia mucho más fácil de entender, (76). En este trabajo se demuestra que todo lenguaje contextual, puede ser generado como la imagen de un **transductor determinista a pila.** En el mismo año **Schüntzenberger** había comenzado el estudio de las propiedades de los lenguajes generados por autómatas deterministas a pila (81). **Chomsky** y **Schützenberger** caracterizaban los lenguajes incontextuales como la imagen bajo homomorfismo de la intersección de un lenguaje de paréntesis con un lenguaje regular (75). Este teorema se conoce con el nombre del **Teorema de Chomsky - Schützenberger**, y ha sido una pieza clave en el desarrollo de la moderna escuela de lenguajes formales en Francia.

En su tesis doctoral presentada en 1963, **S. Greibach** demostró la equivalencia entre las gramáticas tipo 2 y el sistema generador del análisis predicativo (77). Demostró que toda gramática del tipo 2 se puede expresar con reglas que únicamente tienen la forma $P \rightarrow c\ P_1....P_k$ o $P \rightarrow c$, que hoy se conoce como **Forma Normal de Greibach** y coincide con la utilizada en análisis predicativos. Como se recordará, este método de demostración es similar al utilizado por Chomsky para demostrar la equivalencia entre análisis IC y gramáticas tipo 2.

En un artículo de 1960, **Myhill** clasificaba los lenguajes de acuerdo con su complejidad espacial, es decir, en términos de la cantidad de recursos (generalmente cinta) que utilizan para ser reconocidos y definía un nuevo modelo de máquina reconocedora: el **autómata linealmente acotado** (LBA), un modelo similar a la máquina de Turing, pero con la cinta acotada entre dos marcas y el cabezal trabajando entre esas dos marcas sin poder sobrepasarlas (63). Dos años después,

Landweber demostró que la clase de lenguajes generados por las gramáticas de tipo 1 (**lenguajes contextuales** (CS)) incluye la clase de lenguajes reconocidos por los LBA deterministas (78). **Kuroda**, al año siguiente demostraba la equivalencia entre los lenguajes contextuales y los lenguajes reconocidos por LBA indeterministas (86). En el mismo artículo, también se demostraba que la clase de lenguajes reconocidos por LBA deterministas forman un álgebra de Boole, que contiene los lenguajes incontextuales. Finalmente el artículo planteaba dos preguntas que todavía no han obtenido respuesta: ¿Es la clase CS cerrada para la complementación? y ¿La clase de lenguajes reconocidos por LBA deterministas está estrictamente contenida en la clase de lenguajes reconocidos por LBA no deterministas? ¿O es igual a ella?. Este último problema se conoce como el **problema LBA**. Finalmente, en su tesis doctoral, **Cole** (83) demostró que el lenguaje de los palíndromes, $L = \{ \alpha \alpha^R \mid \alpha \in (a, b)^* \}$, pertenece a la clase de los lenguajes CF no deterministas, pero no pertenece a la clase de los CF deterministas, con lo que queda demostrada la inclusión **propia** de los CF deterministas en los CF indeterministas.

Por tanto, en 1964 quedaron caracterizados, mediante los modelos de máquinas que los reconocen, los lenguajes tipo 0, tipo 1, tipo 2 y tipo 3 de la jerarquía de Chomsky en sus versiones deterministas y no deterministas; habiéndose demostrado para la mayoría de sus inclusiones, si son estrictas o no lo son, con la única salvedad de la correspondiente a los lenguajes contextuales deterministas y no deterministas.

Hasta ese momento, las consideraciones de la Informática Teórica habían girado principalmente en torno a lo que podía y no podía ser computado. La teoría de la recursividad, y las diferentes clases de la jerarquía de Chomsky, clasificaban los lenguajes o funciones por algún modelo específico de máquina. La salvedad había sido el mencionado trabajo de Myhill (63) y algunas clasificaciones previas de las funciones recursivas. El polaco **Grzegorczyk**, en 1953 (40), definió una jerarquía de las funciones recursivas, utilizando como "medida de complejidad" el mínimo número de operaciones necesario para computar una función dada. Esta jerarquía se conoce como la **jerarquía de Grzegorczyk**. **Yamada**, en 1961, estudió las funciones computables en **tiempo real**, es decir, computables en un número de pasos igual a la longitud de la entrada (69). No hay que confundir tiempo real con **tiempo lineal** ya que en este

último el número de pasos es proporcional a la longitud de la entrada. **Ritchie**, en 1963 (80) definió otra jerarquía de funciones similar a la definida por Myhill. En octubre de 1964, **J. Hartmanis** y **R. Stearns** presentaron una comunicación al **Symposium on Switching Theory and Logical Design**, celebrado en Princeton, donde clasificaban las funciones por la "**complejidad temporal**": el número mínimo de pasos necesarios para computar cada función particular (85). Este concepto es clave en la Informática Teórica: no basta saber si se puede o no computar, sino también cuán rápidamente se puede computar. Como el mismo Hartmanis comenta /HA.81/, hacia 1962, parcialmente influenciados por el artículo de Yamada, pero desconociendo la jerarquía de Grzegorczyk comenzaron el trabajo sobre complejidad de funciones computadas por máquinas de Turing. Cuando en Marzo de 1963 tuvieron el manuscrito "**On the computational Complexity of algorithms**" listo para publicar, lo enviaron al **Journal of ACM**, pero posteriormente, convencidos de que los contenidos del artículo eran demasiado matemáticos para una revista del ACM, decidieron renunciar al JACM y enviarlo al **Transactions of the American Mathematical Society**. Esta revista les aceptó el artículo, pero les intentó convencer de que un artículo en una revista matemática no debería contener dibujos de máquinas de Turing. El artículo fue finalmente publicado tal cual dos años más tarde (92), pero el incidente pone de manifiesto la dual identidad en que se mueve la Informática Teórica. La comunicación de Princeton era una visión recortada del mencionado artículo.

En 1964, también apareció un artículo de **Rabin** sobre el Tema de la Complejidad Temporal de cálculos en modelos de Máquinas de Turing (87). El resultado principal establece que en tiempo real las máquinas de Turing deterministas con 2 cintas reconocen más lenguajes que las máquinas de Turing deterministas con una cinta. Es éste un magnífico artículo y contiene varias técnicas que después se han convertido en clásicas. Sobre la génesis de dicho artículo, **Hartmanis** relata lo sucedido cuando él y **Stearns** visitaron Harvard en 1963 y mantuvieron varias discusiones con Rabin:

> "...tengo la impresión de que en estas discusiones le sugerimos a Rabin que se debería intentar demostrar que una máquina de Turing con dos cintas funcionaba más deprisa que una máquina de Turing de una cinta, lo que eventualmente se materializó en el artículo de Rabin "Real time computation". (/HA.81/ pg.229).

/IF.81/ IFRAH, G.
"Histoire universelle des chiffres".
Seghers 1981.

/KL.81/ KLEENE, S.C.
"Origins of recursive function theory".
Annals of the History of computing, 3,1. 1981, 52-67.

/KN.72/ KNUTH, D.
"Ancien Babylonian algorithms".
Comm. ACM, 15, 1972, 671-677.

/KN.80/ KNUTH, D.
"Algorithms in Modern Mathematics and Computer Science". En el libro del mismo nombre, editores A. P. Ershov y D. Knuth. Lectures Notes in Computer Science 122, Springer-Verlag 1980, 83-99.

/KT.80/ KNUTH, D.; TRABB PARDO, L.
"The early development of programming languages".
pp. 197-275 en /MHR.80/

/KN.81/ KNUTH, D.
"The art of computer programming: Seminumerical Algorithms".
Addison-Wesley, 2da. edición 1981.

/LL.69/ LEWIS, C; LANGFORD, C.H.
"Historia de la lógica simbólica".
En Newman /NW.79/ vol.5 pp. 248-266

/MHR.80/ METROPOLIS, N.; HOWLETT, J; ROTA, G.C.
"A history of computing in the twentieth century".
Academic Press 1980

/MI.68/ MIDONICK, H.
"The treasure of mathematics: 1"
Pelican Books 1968.

/MO.64/ MOORE, E .F. (ed.).
"Sequential machines: Selected papers"
Addison-Wesly 1964.

/MO.77/ MOORE, D. L.
"Ada, countess of Lovelace"
John Murray, 1977

/MM.61/ MORRISON, P.; MORRISON, E. (ed.).
"Charles Babbage and his calculating enginees"
Dover 1961

/NN.70/ NAGEL, E.; NEWMAN, J. R.:
"El teorema de Gödel"
Ed. Tecnos 1970.
(versión inglesa de 1958).

/NE.57/ NEUGEBAUER, O.
"The exact sciences in antiquity".
Princeton U. Press 1957.

/NW.59/ NEWMAN, R. (ed.).:
"Σ el mundo de las matemáticas"
4 ed. Grijalbo 1979.

/NW.80/ NEWMEYER, F .J.
"El primer cuarto de siglo de la gramática generativo-transformatoria 1955-1980".
Alianza Ed. 1980.

/RA.75/ RANDELL, B. (ed.).
"The origens of digital computers" (2da.ed.)
Springer-Verlag 1975.

/RA.80/ RANDELL B.
"The Colosus"
en /MHR.80/ pp. 47-93.

/RE.70/ REID, C.
"Hilbert".
Springer-Verlag 1970.

/RU.20/ RUSSELL, B.
"Introduction to mathematical philosophy".
Simon Schuster, 1920.

/SA.75/ SAIDAN, A. S.
"The arithmetic of al-Uqlidisi".
Reidel 1975.

/SA.80/ SALES, A.
"La prehistòria de la informàtica: Antecedents històrics de L'ENIAC" y "La primera generació als USA: DE L'ENIAC al transistor". Novatica, 34 1980, 5-37.

/SA.75/ SALTOR, F.
"Evolución de los lenguajes de programación":
Parte I: Novatica, 0 1974; Parte II: Novatica 1 1975.

/SM.56/ SHANNON, C. E. ; Mc CARTHY, J. (ed.).:
"Automata Studies".
Annals of Mathematics Studies.
Princeton Univ. Press, 1956. 5ta. edición 1972.

/TF.63/ TATON R. J.; FLAD, J. P.
"Le calcul mécanique".
Colección Que sais-je, 367
Presses Univers. de France 1963.

/TU.59/ TURING, S.
"Alan S. Turing".
Heffer 1959.

/US.79/ USPENSKI, V. A.
"Máquina de Post".
Ed. MIR, 1979.

/VA.83/ VALERO, M.
"Memoria de Cátedra. Cátedra Arquitectura de Computadores".
Facultad de Informática Barcelona, pp. 9-140 1983.

/ZE.80/ ZEMANEK, H.
"Dixit Algorithmi".
En "Algorithms in Modern Mathematics and Computer Science".
Ed. A.P. Ershov y D. Knuth.
Lectures Notes in Computer Science 122. Springer-Verlag 1980, 1-81.

BIBLIOGRAFIA ESPECIFICA

Comenzaremos dando los acrónimos utilizados en la bibliografía que sigue a continuación:

ACM-STOC	Symposium on the Theory of Computing (de la ACM).
BSTJ	Bell Systems Technical Journal.
CACM	Communications of the ACM.
ICALP	International Colloquium on Automata, Languages and Programming.
IEEE-FOCS	Foundations of Computer Science (de la IEEE).
IEEE Symp Sw. Cir. and Log. Desig.	Conference record on Switching Circuit Theory and Logical Design (de la IEEE. Posteriormente esta conferencia se transformó en FOCS).
JACM	Journal of the ACM.
JCSS	Journal of Computer and Systems Sciences.
MIT QR	MIT Research Electronics Lab. Quarterly Progress Report.
TCS	Theoretical Computer Science.

1 .308 (1) LLULL, R.
"Ars generalis ultima"
1308.

1703 (2) LEIBNITZ, G.W.
"Dissertatio de Arte Combinatoria: In qua ex arithmeticae Fundamentis complicationum ac transpositionum novis praeceptis exstruitur, et usus ambarum per universum scientiarum orbem ostenditur"
Publicado dentro de Philosophische Schrifen, editor C.I. Gerbardt,
Leipzig, 1849.

1837 (3) BABBAGE, C.
"On the mathematical powers of the calculating engine"
(manuscrito con fecha 26-12-1837, reproducido en /RA,75/ pp.(7-17)

1847 (4) BOOLE, G.
"The Mathematical Analysis of Logic".
Cambridge 1847.
(Reeditado por Blackwell 1965).

1854 (5) BOOLE, G.
"An investigation of the laws of though".
Londres 1854
(Reeditado por Blackwell 1967).

1879 (6) FREGE, G.
"Begriffschrift, eine der arithmetischen nachebildete formelsprache des reinen denken"
Halle, 1879 (Reeditado en /HE.77/ pp.1-82).

1895 (7) PEANO, G.
"Formulaire de mathématiques"
5 vol. Turín 1895-1903.

1906 (8) THUE, A.
"Über unendliche zeichenreihen".
Videnskapsselskapets skrifter.
1 math.-naturv. Klasse Kristiania, 1-22.
1906.

1910 (9) RUSSELL, B.; WHITEHEAD, J.
"Principia matemática"
3 vol., 1910-1913. Cambridge U.Press.

1914 (10) TORRES QUEVEDO, L.
"Essays on automatics"
(/RA.75/ pp.87-106).

1920 (11) TORRES QUEVEDO, L.:
"Electromechanical calculating machines".
(en /RA.75/ pp.107-118)

1930 (12) GODEL, K.
"Die völlständigkeit des logikkalküls".
Tesis Doctoral Universidad de Viena.
(También en /HE.77/ pp.582-591)

1931 (13) GODEL, K.
"On formaly undecidable propositions of Principia Matematica and related systems"
Monatshefte für Math. und Physik, 38,173-198
(También en /HE.77/ pp.592-617, y en /DA.65/ pp.5-39)

1933 (14) CHURCH, A.
"A set of postulates for the foundations of logic".
Annals of Mathematics 25. 33, 839-864,

1934 (15) GODEL, K.
"On undecidable propositions of formal mathematical systems"
Apuntes del curso dado por Godel en el Institut for Advance Studies, tomados por Kleene y Rosser.
(También en /DA.65/pp.39-83)

1936 (16) CHURCH, A.
"An unsolvable problem of elementary number theory"
The American Journal of Math. 58, 345-363.
(También en /DA.70/ 110-115)

(17) CHURCH, A.
"A note on the entscheidungsproblem"
Jorunal of Symbolic Logic 1, 40-41. Corrección 101-102.
(También en /DA-70/ 237-253)

(18) KLEENE, S. C.
"General recursive functions of natural numbers"
Mathematische Annalen 112, 727-742.
(También /DA-70/ 237-253).

(19) POST, E. L.
"Finite combinatory processes"
Jour. Symbolic Logic 1, 103-105.
(También en /DA.70/ 289-291).

(20) TURING, A. M.
"On computable numbers with an application to the entscheidungsproblem"
Proc. of London Math. Society 2, 42, 230-265.
(También en /DA.70/, 116-154).

1937 (21) TURING, A. M.
Rectificación a "On computable number"
Proc. of London Math. Society 43, 544-546.

(22) TURING, A. M.:
"Computability and λ-difinability".
J. Symbolic Logic 2, 153-163.

1938 (23) SHANNON, C.
"A Symbolic analysis of relay and switching circuits"
Trans. AIEE, 57, 713-723.

1939 (24) TURING, A. M.
"Systems of logic based on ordinals"
Proceedings of the London Math. Society 2, 45, 161-228 (También /DA.70/, 155-222).

1941 (25) POST, E. L.
"Absolutely unsolvable problems and relatively undecidable propositions"
Manuscrito sin publicar.
Publicado en 1970. /DA.70/ 340-433.

1943 (26) KLEENE, S. C.
"Recursive predicates and quantifiers"
Trans. Amer. Math. Soc. 53, 41-73.

(27) Mc CULLOCH, W. S.; PITTS, E.
"A logical calculus of the ideas immanent in nervous activity"
Bulletin Mathematical Biophysics 5, 115-133.

(28) POST, E. L.
"Formal reductions of the general combinatorial decision problem"
Amer. J. Mathematics 65, 197-215.

1944 (29) POST, E. L.
"Recursively enumerable set of positive integers and their decision problems"
Bulletin Amer. Math. Soc. 50, 284-316.

1946 (30) POST, E. L.
"A variant of a recursively unsolvable problem"
Bulletin Amer. Math. Soc. 52, 264-268.

1947 (31) MARKOV. A. A.
"On the representation of recursive functions"
Doklady Akademii Nauk. SSSR, 58, 1891-1892.

(32) POST, E. L.
"Recursive unsolvability of a problem of Thue"
Journal of Symbolic Logic, 12, 1-11.
(También /DA.70/, 293-303)

(33) TURING, A. M.
"Intelligent Machinery"
Monografía con fecha sept. 1947
Publicada en Machine Intelligence, vol.5, 3-23. 1969

1948 (34) POST, E. L.
"Degrees of recursive insolvabily"
Bulletin Ame. Math. Soc. 54, 641-642.

(35) WIENER, N.
"Cybernetics: Or control and communication in the animal and the machine"
MIT Press.

1949 (36) SHANNON, C.
"The synthesis of two-terminal switching circuits"
BSTJ, 28, 59-98.

1951 (37) KLEENE, S. C.
"Representation of events in nerve nets and finite automata"
Rand Corporation Project RM-704,
Dec, 15, 1951
Publicado en /SM.56/ 3-42

1952 (38) KLEENE, S. C.
"Introduction to methamatematics"
North-Holland.

1953 (39) BART-HILLEL, A.
"A quasi-arithmetical notation for syntactic description"
Language 29, 47-58.

(40) GRZEGORCZYK, A.
"Some classes of recursive functions"
Rosprawy Matematyczne 4,
Instituto Matemático de Varsovia.

1954 (41) HUFFMAN, D.
"The synthesis of sequential switching circuits"
Journal of Franklin Institute, vol.257. 3-4.
161-190 y 275-303
(También en /MO.643, 3-62).

(42) KLEENE, S. C.; POST, E. L.
"The upper semi-lattice of degrees of recursive unsolvability"
Annals of Math. 59, 379-407.

1955 (43) CHOMSKY, N.
"The logical structure of linguistic theory"
Ph. D. D. Dissertation, MIT.

1956 (44) CHOMSKY, N.
"Three models for the description of language"
Proc. Symposium on Information Theory.
MIT sept. 1956.

(45) DE LEEUW, K, ; MOORE E. F.; SHANNON, C. E.; SHAPIRO, N.
"Computability by Probabilistic Machines"
En /SM.56/.

(46) MOORE, E. F.
"Gedanken experiments on sequential machines"
publicado en /SM.56/ 129-156.

(47) MUCHNIK, A.
"On the separability of recursively enumerable sets"
Doklady Akademiĭ Nauk SSSR, 108. 194-197.

1957 (48) FRIEDBERG, R.
"Two recursively enumerable sets of incomparable degrees of unsolvability"
Proc. National Academy of Sciences, 43, 236-238.

(49) MYHILL, J.
"Finite autómata and the representation of events"
WADD, TR-57-624, 112-137, Wright AFB.

(50) RABIN, M.O.
"Two way finite automata"
Proc. Summer Institute of Symbolic Logic at Cornell.

1958 (51) CHOMSKY, N.
"Some properties of phrase structure grammars"
MIT QR, 49, 108-111

(52) CHOMSKY, N.; MILLER, G. A.
"Finite state languages"
Information and Control 1, 91-112.

(53) COPI, I. M.; ELGOT, C. WRIGHT, J. B.
"Realization of events by logical nets"
JACM; 5, 181-196.
(También /MO.64/, 175-192).

(54) DAVIS, M.
"Computability and Unsolvability"
Mc Graw-Hill.

(55) LUPANOV, O. B.
"A method of Circuit Synthesis"
Izv. V.U.Z. Radiofiz 1, 1, 120-140.

(56) NERODE, A
"Linear automaton transformation"
Proc. Amer. Math. Soc. 9, 541-544.

1959 (57) BACKUS, J. W.
"The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM conference"
Proc. Int. Conf. on Information Processing. UNESCO, Paris 1959, 125-132.

(58) CHOMSKY, N.
"On certain Formal Properties of Grammars"
Information and Control, 2, 137-167.

(59) RABIN, M. O.; SCOTT, D.
"Finite antómata and their decision problems"
IBM Journ. of Reseach and Developmet 3, 2, 114-125.
(También en /MO.64/ 63-91).

(60) SHEPHERDSON, J. C.
"The reduction of two-way automata to one-way automata"
IBM Journ. Research and Development 3, 2, 198-200.
(También en /MO.64/ 92-97.

1960 (61) BAR-HILLEL, Y.; GAIFMAN, H. ;SHAMIR, E.
"On categorical and phase structure grammars"
Bulletin Research Council Israel 9, 155-166.

(62) Mc NAUGHTON, R. ; YAMADA, H.
"Regular expressions and state graphs for automata"
IRE-PGEC p, 39-47
(También /MO.64/157-174)

(63) MYHILL, J.
"Linear bounded automata"
WADC Tech.Note 6-165.

(64) NAUR, P.
"Report on the algorithmic languaje ALGOL 60"
CACM 3, 76-83.

(65) SCHEINBERG, S.
"A note on the boolean properties of context-free languages"
Information and Control 3, 372-375

1961 (66) BAR-HILLEL, Y; PERLES M.; SHAMIR, E.
On formal properties of simple phrase structure grammars"
Z. Phonetik Sprach. Komm. 14, 143-179.

(67) PARIKH, R. J.
"Language-generating devices"
MIT QR 60, 199-222.
Publicado posteriormente:
"On context-free languages"
JACM 13, 570-581 1966.

(68) SCHUTZENBERGER, M. P.
"On the definition of a family of automata"
Information and Control 4, 245-270.

(69) YAMADA, H.
"Real-time computation and recursive functions no real-time computable"
IRE Trans. Compt. EC-D, 753-760.

(70) YNGVE, V.A.
"A model and a hypothesis for language structure"
Proc. Amer. Phil. Soc. 104, 444-466.

1962 (71) CHOMSKY, N.
"Contex-free grammars and pushdown storage"
MIT QR 65. 187-194.

(72) GINSBURG, S. ; RICE, H. G..
"Two families of languages related to ALGOL"
JACM 9, 350-371

(73) GROSS, M.
"On the equivalence of models of language used in the fields of mechanical translation and information retrival."
Versión preliminaria 1962.
Publicado en Inform. Storage Retrival 2, 1964 43-57.

(74) SCHUTZENBERGER, M. P.
"Finite counting automata "
Information and Control 5, 91-107.

1963 (75) CHOMSKY, N.; SCHUTZENBERGER, M. P.
"The algebraic theory of context-free languages"
En: Computer Programming and Formal Systems.
Ed. Braffort and D.Hirschberg, North-Holland 1963.

(76) EVERY, R. J.
"The theory and applications of pushdown store machines "
Tesis Doctoral en la Universidad de Harvard.

(77) GREIBACH, S.
"Inverses of Pharse Structure Generators".
Tesis doctoral en la Universidad de Harvad, 1963.
Publicado como: "The undecidability of ambiguity problem for minimal linear grammars"
Inform. and Control. 6 1963 119-125; "Formal parsing systems". CACM, 7 1964, 499-504.

(78) LANDWEBER, P. S.
"Three theorems on phrase structure grammars of type 1"
Information and Control, 6, 137-146.

(79) RABIN, M. O.
"Probabilistic automata"
Information and Control 6, 230-245.

(80) RITCHIE, R. W.;
"Classes of predictably computable functions"
Trans. Amer. Math. Society 106, 139-173.

(81) SCHUTZENBERGER, M. P.
"On context-free languages and push-down automata"
Information and Control 6, 3, 246-264

1964 (82) COBHAM, A.
"The intrinsic computational difficulty of functions"
Proc. 1964 Congress for Logic Mathematics and Philosophy of Science 24-30. North-Holland.

(83) COLE, S. N.
"Real-time computation by iteractive arrays of finite state machines"
Ph.D. Harvard.

(84) SHEPHERDSON, J. C.; STURGIS, H. E.
"Computability of recursive functions"
JACM 10: 2, 217-255

(85) HARTMANIS, J.; STEARNS, R. E.
"Computational complexity of recursive sequences"
Proc. IEEE Symp. Sw. Circ. Th. and Log. Des.
Princeton 82-90.

(86) KURODA, S. Y.
"Classes of languages and linear-bounded automata"
Information and Control 7, 207-223.

(87) RABIN, M. O.
"Realtime Computation"
Israel Journal Mathematics, 1, 203-211

(88) ROZENBERG, A.
"On multi head finite automata"
IEEE Symp. Sw. and Aut. 221-228

(89) TRAKHTENBROT, B. A.
"Turing computations with logarithmic delay"
Algebra i Logika 3, 4, 33-48

1965 (90) EDMONDS, J.
"Paths trees and flowers"
Canada Journal Mathematics, 17, 449-467.

(91) GINSBURG, S.; GREIBACH, S.
"Deterministic context-free languages"
Proc. IEEE Symp. Sw. Circ. Th. and log. Des.
Ann Arbor 203-220
Publicado en Inforormation and Control 8, 607-639.

(92) HARTMANIS, J.; STEARNS, R. E..
"On the computational complexity of algorithms"
Trans. Amer. Math. Soc. 117, 285-306

(93) HENNIE, F. C.
"One-tape off-line Turing machine computations"
Information and Control 8:6, 553-578.

(94) KASAMI, T.
"An efficient recognition and syntax-analysis algorithm for CFL"
Report 2/65, University of Hawai

(95) KNUTH, D. E.
"On the translation of languages from left to right"
Information and Control 8, 607-639

(96) KOLGOMOROV A. N.
"Three approches for defining the concept of information quantity"
Prob. Inform. Trans. 1, 1-7.

(97) LEWIS, P. M; STEARNS, R. E.; HARTMANIS, J.
"Memory bounds for recognition of context-free and context-sensitive languages"
Proc. IEEE Symp. Sw. Circuit Th. and Log. Des.
Ann Arbor, 191-202

(98) STEARNAS, R. S.;HARTMANIS, J.; LEWIS, PM.
"Hierarchies of memory limited computations"
Proc. IEEE Symp. Sw. Circ. and Log. Des.
Ann Arbor 179-190

(99) WINOGRAD, S.
"On time required to perform addition"
JACM, 12, 2, 277-285

1966 (100) CHAITIN, G. J.
"On the length of programs for computing binary sequences"
JACM 13, 547-569

(101) GINSBURG, S.; GREIBACH, S. HARRISON; M. A.
"One way stack automata"
Proc, IEEE Symp. Sw. and. Aut. Th.
Berkeley, 47-62

(102) HENNIE, F. C.; STEARNS, R. E.
"Two-tape simulation of multitape Turing machines"
JACM, 13; 4, 533-546

(103) YOUNGER, D. H.
"Contex-free language processing in time n^3"
Proc. IEEE Symp. Sw. and Aut. Th.
Berkeley 7-20

1967 (104) BLUM, M.
"A machine-independent theory of the complexity of recursive functions"
JACM 14; 2, 322-336.

(105) GINSBURG, S. ;GREIBACH, S.
"Abstract families of languages"
Proc. IEEE Symp. Sw.and Aut. Th. 228-245.
Posteriormente publicado en"Studies in AFL"
Mem. AMS, 87, 1969, 1-32

(106) GREIBACH, S.
"An infinite hierarchy of context-free languages"
Proc, IEEE Symp. Sw. Cir. and Aut. Th., 32-66.

(107) HOPCROFT, J. ; ULLMAN, J. D.
"An approach to a unified theory of autómata"
IEE. Symp. on Sw. and Aut. Th. 140-147.
Publicado en Bell Syst. Technical Journal 46 (1967) 1793-1829

(108) HOPCROFT, J.; ULLMAN, J. D.
"Nonerasing stack automata"
JCSS, 166-186

(109) NIVAT, M.
"Transductions des lenguages de Chomsky"
Tesis doctoral Univ. Grenoble 1967.
Publicado en Annals Inst. Fourier, Grenoble, 18 1968, 339-455.

1968 (110) GREIBACH. S.
"Checking automata and one-way stack languages"
IEEE Symp. on Sw. Circ. Des. and Aut. Th. 287-291

(111) LINDENMEYER, A.
"Mathematical models for cellular interactions in development; I and II"
J.Theoretical Biology 18, 280-315.

1969 (112) BOOK, R. V. ; GREIBACH, G.
"Quasi-realtime languages"
ACM-STOC, 15-18.

(113) SAVITCH, W. J.
"Deterministic simulation of non-deterministic Turing machines"
ACM-STOC, 247-248.

1970 (114) BOOK, R. V. ; GREIBACH, S. A.; WEGBREIT, B.
"Tape and time bounded Turing acceptors and AFL"
ACM-STOC,
92-99. Publicado como: BOOK, GREIBACH, IBARRA, WEGBREIT;
"Tape bonnded Turing acceptors and principal AFLs"
JCSS, 5, 1971 622-625.

(115) GINSBURG, S. ; GREIBACH, S.
"Principal AFLs"
JCSS, 4, 308-338

(116) NECHIPORUK, E. I.
"Realizations of disjunctions and conjunctions in monotone bases"
Probl. Kibernetiks 23, 291-293.

(117) WINOGRAD, S.
"On the number of multiplications necessary to computer certain functions"
Communications Pure Applied Mathematics 23, 165-179

1971 (118) BOASSON, L.
"Cônes rationnels et familles agréables de langages. Application au langage à compteur."
Tesis de estado. Univ. París.
Publicado como
"An iteration theorem for one-counter languages"
ACM-STOC, 116-120.

(119) BOOK, R. V. ; WAGBREIT, B.
"A note on AFLs and bounded erasing"
Information and Control 19, 18-29.

(120) COOK, S. A.
"The complexity of theorem-proving procedures"
ACM-STOC, 151-158

(121) KRAPCHENKO, V. M.
"A method of obtaining lower bounds for the complexity of P-Schemes".
Mat. Zamet 10, 1, 83-92

1972 (122) BORODIN, A.
"Computational complexity and the existence of complexity gaps"
JACM 19, 1, 158-174

(123) BOOK, R. V.
"On languages accepted in polynomial time"
SIAM J. Comput. 1, 281-287

(124) GILL, J. T.
"Probabilistic Turing machines and complexity of computation"
Tesis Ph. D., U C Berkeley.
Publicado como: "Computational complexity of probabilistic Turing machines"
SIAM J. Comput. 6, 4, (1977) 675-695

(125) GOLDSTINE, J.
"Substitution and bounded languages"
JCSS 6, 30-76

(126) KARP, R. M.
"Reducibility among combinatorial problems"
En Complexity of Computations, ed. MILLER y THATCHER.
Plenum Press, 85-104

(127) KLEENE, V.;MINTY, G.
"How good is the simplex algorithm"
Boeing Technical Report, 1972

(128) MEYER, A. R. ; STOCKMEYER, L. J.
"The equivalence problem for regular expressions with squaring requires exponential space"
IEEE- FOCS, 125-129.

(129) SAVAGE, J.
"Computational work and time on finite machines"
JACM, 19, 4, 660-674

1973 (130) COOK, S. A. ; RECKHOW, R. A.
"Time bounded random access machines"
JCSS 7, 4, 354-375.

(131) LEVIN, L. A.
"Universal problems of combinatorial search"
Problemi Peredachy Informachii, 9, 115-116

1974 (132) AANDERAA, S. O.
"On K-tape versus (K-1)-tape real time computation"
En: Complexity of Computation
(Proc. SIAM-AMS, vol. 7), 75-96.

(133) FISCHER, M. J. ; RABIN, M. O.
"Super-exponential complexity of Presburger arithmetic"
En Complexity of Computation.
(Proc. SIAM-AMS, vol. 7) AMS.

(134) ROZENBERG, G.; SALOMA, A.
" L Systems"
Lectures Notes Computer Sciences, 15.
Springer-Verlag

1975 (135) BAKER, T.; GILL, J.; SOLOVAL, R.
"Relativizations of the P = NP? question"
SIAM J. Computing 4, 4, 431-442.

(136) LADNER, R. E,
"On the structure of polynomial time reducibility"
JACM 22, 1, 155-171

(137) HARPER,L. H.; HSIEH, W.; SAVAGE, J.
"A classe of boolean functions with linear combinational complexity"
TCS 1, 2, 161-183.

(138) PRATT, V. R.
"Every prime has a succinct certificate"
SIAM J. Computing 4:3, 214-220

1976 (139) CHANDRA, A. K.; STOCKMEYER, L. J.
"Alternation"
IEEE-FOCS, 98-108

(140) KOZEN, D.
"On parallelism in Turing machines"
IEEE-FOCS, 89-97.

(141) MILLER, G. L.
"Riemann's hypothesis and test for primality"
JCSS, 13:3, 300-317.

(142) SCHNOR, C. P.
"The network complexity and the Turing machine complexity of finite functions"
Acta Informatica 7, 95-107.

(143) YAO, A. ; RIVEST, R. L.
"K+1 heads are better than k"
IEEE-FOCS, 67-70

1977 (144) ADLEMAN, L. ; MANDERS, K.
"Reducibility, randomness and intractability"
ACM-STOC, 81-88.

(145) BERMAN L.; HARTMANIS, J.
"On isomorphisms and density of NP and other complete sets"
SIAM J. Computing 6:2, 305-322.

(146) RABIN, M. O.
"Probabilistic algorithms"
En: "Algorithms and Complexity: New Directions and Recent Results".
Ed. TRAUB, J.
Academic Press, 21-40.

(147) SAVAGE, J.
"The complexity of computing"
Wiley-Interscience.

(148) SOLOVAY, R.; STRASSEN, V.
"A Fast Monte Carlo test for primality"
SIAM J. Comput. 6:1, 84-85.

(149) STOCKMEYER, L. J.
"The polynomial time hierarchy"
TCS 3:1, 1-22.

1978 (150) BERMAN, P.
"Relationship between density and deterministic complexity of NP-Complete languages"
ICALP, Udine.

(151) KOZEN, D.
"Indexing of subrecursive classes"
ACM-STOC. 287-295.

1979 (152) COOK, S. A.
"Deterministic CFL's are accepted simultaneously in poly. time and log. square space".
ACM-STOC, 338-345.

1980 (153) KARP, R.; LIPTON, R. J.
"Some connections between non-uniform and uniform complexities classes"
ACM STOC, 302-309.

(154) MAHANEY, S.
"Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis",
IEEE-FOCS, 42-49.

(155) PIPPENGER, N.
"Algebraic complexity theory"
IBM Jour. Res. Develop. 25, 5, 825-832.

1982 (156) BALCAZAR, J. L.; DIAZ, J.
"A note on a theorem by Ladner"
Information Processing Letters 15, 84-86.

1983 (157) SIPSER, M.
"A complexity theoretic approach to randomnes"
ACM-STOC, 330-335.

1984 (158) BALCAZAR, J. L.
"En torno a oráculos que ciertas máquinas consultan y a las espantables consecuencias a que ello da lugar"
Tesis Doctoral, FIB-UPC, Julio 1984.

(159) HARTMANIS, M.
"Generalized Kolmogorov Complexity"
IEEE-FOCS.

Puede considerarse que en esta época, la Informática Teórica alcanzó su mayoría de edad. Dos temáticas principales, la complejidad y la teoría de lenguajes, formarán a partir de aquí una malla entrelazada, que configura lo que hoy se denomina Informática Teórica.

5. De 1965 hasta nuestros días: la plenitud de la Informática Teórica.

Llegados a este punto, para mayor claridad, desglosaremos la historia de la Informática Teórica en dos apartados: La **Teoría de los Lenguajes Formales**, en donde expondremos las principales líneas de investigación sobre las clases de lenguajes definidos por modelos de máquinas más restrictivas que las máquinas de Turing, es decir, los lenguajes situados entre las clases tipo 3 y tipo 1 de la jerarquía de Chomsky y la **Teoría de la Complejidad**, cuyas líneas de investigación versan sobre la complejidad del modelo de máquina de Turing y otros modelos equivalentes, con aplicación al diseño y análisis de los algoritmos.

TEORIA DE LOS LENGUAJES FORMALES

A partir de 1965, el estudio de los lenguajes **deterministas incontextuales** (DCFL) cobró una gran relevancia, principalmente a causa de su aplicación en la construcción de compiladores para lenguajes de programación. En 1965 **Ginsburg** y **Greibach** (91) estudiaron aquellos lenguajes, demostrando toda una serie de propiedades de cierre, así como el hecho de que todo autómata determinista de pila puede ser convertido de manera efectiva en una gramática CF inambigua. En el mismo artículo también daban una demostración alternativa de la inclusión estricta de los CFL deterministas en los indeterministas. Un artículo de **Knuth** de ese mismo año (95) introdujo las gramáticas LR(K), que constituyen una restricción de las gramáticas incontextuales, siendo sin embargo suficientemente amplias como para expresar la sintaxis de la mayoría de los lenguajes de programación, y que, a diferencia de aquéllas, tienen analizadores eficientes, lo que las hace idóneas para la

construcción de compiladores. De hecho, en el artículo, Knuth discutía la utilización de las gramáticas LR(K) en la construcción de un compilador para el lenguaje ALGOL. También demostró que el lenguaje generado por una LR(K) es DCFL y que para todo DCFL existe una gramática LR(K). Finalmente, el americanado artículo demostraba la indecidibilidad del problema de ver si para una K fija y una gramática incontextual determinada, ésta es o no es LR(K).

A finales de la década de los sesenta, se definieron y estudiaron las propiedades de una serie de lenguajes CF: los **lineales, metalineales** y **acotados.** En general, su importancia no pasa de constituir subclasificaciones de los incontextuales. Ver por ejemplo /GR.81/.

Se conocía ya la existencia de un algoritmo polinómico para reconocer si un lenguaje era incontextual, (algoritmo atribuido a **Cocke** y nunca publicado) cuando en 1965 **Kasami** (94) y en 1966 **Younger** (103) dieron de manera independiente sendos algoritmos para reconocer si un lenguaje es CF en tiempo acotado por n^3. En 1979, **Cook** demostró que todos los lenguajes incontextuales deterministas se pueden reconocer utilizando a la vez tiempo n^3 y espacio $(\log n)^2$ (152)

Los conceptos de autómata a pila y lenguaje incontextual fueron extendidos de varias maneras. Una primera extensión es la constituida por el **"autómata stack"**, cuya pila es accesible a la mirada del observador y que fue introducido por **Grinsburg, Greibach** y **Harrison** en 1965 (101). Como una variación se definió el **"autómata stack no borrador"** (108). Existe toda una jerarquía de lenguajes entre los incontextuales y los contextuales, definidos por modificadores del autómata stack (108) y (110). Las familias de los lenguajes incontextuales y las de los definidos por autómatas stack no borradores son incomparables (108). Además, la familia de lenguajes aceptados por autómatas stack no borradores es la única que tiene propiedades de cierre diferente, ya que por ejemplo, no es cerrada para substitución (108). En el mencionado artículo de Ginsburg, Greibach y Harrison también se demuestra que todo conjunto

enumerable recursivamente se puede obtener como intersección de dos DCFL.

El estudio de las propiedades de cierre de diferentes funciones de lenguajes, puso de manifiesto la similitud de las demostraciones de dichas propiedades. Esta constatación sugirió la idea de la existencia de una teoría unificadora para una amplia clase de familias de lenguajes. En 1967 surgieron dos teorías: los autómatas globo de Hopcroft-Ullman (107) y las familias abstractas de lenguajes (AFL) de Ginsburg y Greibach (105). De las dos teorías la que más influencia posterior ha tenido han sido las AFL.

Una AFL es una familia de lenguajes cerrada para las operaciones de unión, producto, Kleene - +, morfismo inverso, intersección con los lenguajes regulares y morfismo λ-exempto. Una AFL-plena es una AFL que además es cerrada por la estrella de Kleene y para todo tipo de homomorfismo. Ginsburg y Greibach también definieron las familias abstractas de autómatas (AFA) y relacionaron AFAs y AFLs. Utilizando estos conceptos, Greibach dió en 1967 una clasificación más fina de los lenguajes incontextuales (106). Por otra parte, Nivat, en 1967, utilizando AFL y el teorema de representación de Schützenberger anteriormente mencionado, definió el concepto de transductor racional y dió sus correspondientes caracterizaciones (10). Esta técnica fue utilizada en 1971 por Boasson para estudiar los lenguajes reconocidos por un autómata a contador (118) y por Goldstine en 1972 para estudiar la familia de lenguajes acotados (125). Más adelante veremos ejemplos de aplicaciones de los AFL para resolver problemas en otras áreas de la teoría de lenguajes y de la complejidad.

En 1968 el biólogo A. Lindenmeyer introdujo los sistemas-L, una clase de lenguajes que simulan el desarrollo de los sistemas biológicos (111). Las ideas de los sistemas-L ya estaban de alguna manera implícitas en un artículo previo de Chomsky (156). En los sistemas L las reglas de producción se aplican en paralelo y no existen diferencias entre termina-

les y no terminales cuando las reglas de producción de las gramáticas que generan estos lenguajes son del tipo 2, tenemos los sistemas OL. Las reglas de producción de algunos sistemas-L pueden ser descritas como conjuntos llamados tablas y cada símbolo del sistema debe ser escrito utilizando únicamente reglas de la misma tabla; a estos sistemas-L se les denomina sistemas-L por tablas y son los más utilizados. El lenguaje de este tipo más interesante es el ETOL. El libro de Rozemberg y Salomaa (134) recoge la mayoría de resultados sobre los sistemas-L.

La teoría de los AFL permite relacionar los sistemas L con la jerarquía de Chomsky. En 1970 Ginsburg y Greibach (115) generalizan el teorema de representación de Chomsky-Schützenberger, definiendo los conceptos de AFL-principal, super-AFL e hyper-AFL. Después caracterizaron los lenguajes regulares como la familia mínima de AFL-plenas; los CFL coinciden con el mínimo Super-AFL, y los ETOL coinciden con el mínimo hiper-AFL (134) pp.250-301.

TEORIA DE LA COMPLEJIDAD.

En el apartado precedente, hemos descrito el desarrollo de clasificaciones para los lenguajes que se encuentran por debajo de los del tipo 0; en el presente consideramos cuestiones de complejidad de las máquinas que reconocen lenguajes tipo 1 o por encima.

En 1965 Hartmanis y Stearns, conjuntamente con Lewis, continuaron su línea de trabajo sobre la complejidad de las M.T. En (97) y (98) estudiaron a fondo la complejidad espacial de las máquinas de Turing y crearon la jerarquía de complejidad espacial: entre la clase de lenguajes que utilizan espacio (número de celdas) constante y la clase de lenguajes que utilizan espacio log log n, no existe ninguna clase de complejidad espacial intermedia. Hartmanis, Lewis y Stearns no llegaron a entender este hecho en toda su generalidad, puesto que desconocían que en 1964 Trankhteubrot había demostrado que este tipo de "saltos" aparecen en

todas las medidas de complejidad (89). El artículo de Trankhteubrot estaba escrito en ruso y pasó desapercibido para la comunidad científica internacional, hasta el extremo de que Borodin redescubrió en 1972 el resultado, que hoy se conoce con el nombre de Teorema de Borodin (Borodins gap Theorem) (122).

En (97) Hartmanis, Lewis y Stearns demostraron, entre otros resultados, que los lenguajes CF se pueden reconocer en espacio $(\log n)^2$, siendo n la longitud de la palabra de entrada, e introdujeron varios tipos de técnicas nuevas para demostrar cotas inferiores de la complejidad espacial de lenguajes.

Rozenberg en 1964 (88) consideró modelos de autómatas con múltiples cintas de entrada y con múltiples cabezales de lectura sobre una misma cinta de entrada. Un problema que dejó sin resolver en su artículo es el de determinar si, para un autómata que sólo puede leer la cinta de entrada en una dirección, K cabezas de lectura son mejores que K-1, esto es, aceptan un conjunto más grande de lenguajes. El problema fue solucionado positivamente en 1976 por Yao y Rivest (143). Por otra parte en 1974, Aanderaa (132) generalizó el resultado de Rabin mencionado anteriormente (87), demostrando que en tiempo real para máquinas de Turing deterministas K + 1 cintas reconocen más lenguajes que K cintas.

En 1966, Hennie y Stearns demostraron que el costo de pasar de una MT con K cintas a otra con 2 cintas, es n log n (102). Anteriormente Hartmanis y Stearns habían demostrado que el costo de pasar de una MT de K cintas a una MT con 1 cinta era cuadrático (42).

Blum, en 1967, propuso una axiomatización de la teoría de la complejidad (104). Su artículo también incluía el teorema de la aceleración, conocido con el nombre de "Blums Speed-up Theorem". Este teorema demuestra la existencia de lenguajes para los que no existe ninguna MT que los reconozca en tiempo óptimo. Hartmanis menciona

que una copia manuscrita de este artículo ya circulaba por las universidades en 1963 y que él se sintió impresionado cuando lo leyó por vez primera (/HA-83/ pg.48).

En 1969, W. Savitch (113) investigando el problema del LBA que ya hemos mencionado, demostró que para una MT acotada por espacio polinómico, el determinismo y no determinismo tienen potencia equivalente, es decir, que DSPACE (S(n)) = NSPACE (S^2(n)) para cualquier S(n) > log n. Es éste un resultado sorprendente, cuyo equivalente para el caso de tiempo polinómico no ha podido ser demostrado.

Retrocediendo unos años, en 1964 y 1965 dos artículos de Cobham (82) y Edmonds (90), respectivamente, ponían énfasis en la necesidad de clasificar los problemas decidibles en tratables (aquéllos para los que existen algoritmos deterministas que resuelven el problema en tiempo polinómico respecto a la longitud del problema y que constituyen la actualmente denominada clase P) y aquellos problemas decidibles que no son tratables, pero tampoco son "excesivamente malos". Cook, inspirándose en la teroría de la función recursiva, formalizó estos conceptos definiendo las clases P y NP (la clase de problemas resolubles en tiempo polinómico mediante algoritmos no deterministas) y demostró que el problema SAT, consistente en determinar si una fórmula booleana en forma normal conjuntiva es o no satisfactible, pertenece a la clase NP-Completa (120). Al año siguiente, Karp (126), rehizo y extendió los resultados de Cook utilizando la m-reducción en lugar de la reducción de Turing que utilizaba Cook. Esto clarificó conceptos, sobre todo de cara a su aplicación práctica en el análisis de la complejidad de problemas. En el artículo mostró que más de cincuenta problemas de combinatoria e investigación operativa pertenecían a la clase de NP-Completa. Desde entonces hasta el momento, se ha demostrado la pertenencia de más de 600 problemas a dicha clase. La no pertenencia de un problema a la clase NP-Completa significa que no existen algoritmos deterministas que lo resuelvan en tiempo polinómico, para cualquier entrada del problema. Claramente se tiene que $P \subseteq NP$; Karp conjeturó que $P \neq NP$. Este se ha

convertido en el problema más importante (no resuelto) de la Informática Teórica y gran parte de los resultados obtenidos en complejidad abstracta han tenido como motivación el solucionar dicho problema.

Paralelamente, Book y Greibach demostraban en 1969 que todo lenguaje en NTIME(n) es la imagen de un homomorfismo no borrador sobre una intersección de CFLs, lo que permite demostrar que la clase NTIME(n) es un AFL principal (112). Posteriormente, conjuntamente con Wegbseit (114), demostraron que NSPACE(n) también es un AFL principal. Como consecuencia de lo anterior, Book en 1972 demostró que P = NP si y sólo si P es cerrado bajo un homomorfismo no borrador y ello a su vez si y sólo si NTIME(n) $\subset$ P (123). Esta forma de estudiar las clases P y NP es diferente de la anteriormente expuesta.

Mientras tanto en Rusia L. A. Levin, de manera independiente, publica en 1973 sus resultados en donde establecía una teoría de P y NP muy similar a la de Karp (131). Hasta 1976 el mundo occidental desconoció los trabajos en paralelo de sus colegas soviéticos sobre las clases de complejidad (GL-80).

En 1972 Meyer y Stockmeyer demostraron que el problema de ver si dos expresiones regulares, con la operación cuadrado, son o no equivalentes, necesita como mínimo espacio, y por tanto tiempo, exponencial. Este fue el primer problema decidible para el que se demostró la imposibilidad de ser resuelto en tiempo o espacio polinómico (128). Posteriormente se han demostrado resultados equivalentes para otros problemas, así por ejemplo Fisher y Rabin demostraron la exponencialidad de la complejidad de la aritmética de Presburger (133). En el artículo mencionado, Meyer y Stockmeyer crearon una jerarquía infinita similar a la aritmética, pero imponiendo la condición de polinomialidad, que comienza en la clase P y abarca hasta la clase NSPACE. Esta jerarquía se conoce con el nombre de jerarquía polinómica y fue estudiada extensamente en 1977 por Stockmeyer (149). A diferencia de la jerarquía aritmética, no se sabe si las inclusiones (todas o algunas) de los miembros

de la jerarquía son o no son estrictas. A lo máximo que se ha llegado es a un reciente resultado de **Karp** y **Lipton**, que asegura que si SAT es Turing-reducible en tiempo polinómico a un conjunto esparso (un conjunto en donde el número de cadenas de longitud n está acotado por un polinomio en n) entonces la jerarquía polinómica colapsa en $\sum_2^P$ (152).

Numerosos artículos se han escrito sobre la estructura de la clase NP bajo la hipótesis de $P \neq NP$, con la finalidad última de encontrar razones en la estructura interna de los problemas NP-Completos que demuestren $P \neq NP$. **Ladner** demostró en 1975 que, bajo la hipótesis $P \neq NP$, existen problemas NP que no son P ni NP-Completos (136). En 1981 este resultado se cumplió, demostrándose que bajo la misma hipótesis existen infinitas clases entre P y NP-Completa (155). Se estudian las clases DSPACE (logn) y NSPACE (logn) (113) y (123). Sintetizando la cuestión, hoy en día se conoce:

$$DSPACE(\log n) \subsetneq NSPACE(\log n) \subsetneq P \subsetneq NP \subsetneq PSPACE = NSPACE$$

sabiéndose que NSPACE (log n) $\neq$ PSPACE, y desconociéndose si el resto de las inclusiones son estrictas o no.

Berman y **Hartmanis** estudiaron aspectos de densidad en problemas en NP. En 1977, demostraron que todos los conjuntos NP-completos que se conocían eran **p-isomorfos** (existía una reducción polinómica que era un isomorfismo) y conjeturaron que ésta era una propiedad de los conjuntos NP-completos, conjetura que aún no ha sido demostrada (145). Al año siguiente, **P. Berman** demostró que si $P \neq NP$, entonces no existen **conjuntos unilaterales** (definidos sobre un alfabeto unario) en NP-completa (150). En 1980 **Mahaney** demostraba que si $P \neq NP$, entonces no existen conjuntos esparsos en NP-completa (153). Las preguntas correspondientes sobre la existencia de lenguajes uniliterales, o esparsos, en la clase intermedia entre P y NP-Completa, es un problema aún sin solucionar. Por otra parte, **Adelman** y **Manders** demostraron en 1977 que ciertos problemas tenían la propiedad de pertenecer a P si y sólo si NP = C0 -

NP (144). Todos estos trabajos, aunque no consiguieron establecer que $P \neq NP$, reforzaron dicha conjetura.

Un sorprendente artículo de **Baker, Gill** y **Solovay** en 1975 demostraba que, cuando se relativiza la teoría de la complejidad existen oráculos A y B tales que $P^A = NP^A$ y $P^B \neq NP^b$, siendo P^X y NP^X las clases de lenguajes reconocidos por MT con oráculo X deterministas y no deterministas, respectivamente (135). Este resultado parece indicar que los métodos clásicos de diagonalización y simulación no funcionarán en el problema de demostrar que $P \neq NP$, puesto que dichos métodos relativizan. Esto ha llevado a algunos informáticos teóricos, a conjeturar la independencia del problema $P = NP$ respecto de cualquier axiomática (ver el cap.7 /HA.76/). Por otra parte, **Kozen** demostró que si existe una demostración de que $P \neq NP$, tiene que existir una demostración por diagonalización de ese mismo resultado (151). En los últimos años, se han construido todo tipo de oráculos, para separar clases ralativizadas de complejidad. Una versión detallada de dichos resultados se puede encontrar en la tesis de **J. Balcázar** (157).

Otro aspecto de la complejidad ha sido la creación de diferentes modelos de computación estudiando su potencia respecto a la máquina de Turing. De esta manera se incorporan en el modelo de computación diferentes características que la máquina de Turing no posee. Un primer modelo es la **Máquina de Acceso Aleatorio** (RAM). Este modelo de máquina fue utilizado por vez primera en 1963 (78). En 1973 **Cook** y **Reckhow** demostraron que toda MT básica puede ser simulada por una RAM en $O(T(n) \log T(n))$ y que a su vez una RAM puede ser simulada por una MT básica en $O(T^3(n))$, en donde $T(n)$ es la complejidad temporal genérica de un algoritmo sobre la máquina a simular (130). La particularidad de la RAM es el ser una aproximación más realista al ordenador convencional que la MT, y muchos libros de texto básicos de Informática Teórica utilizan este modelo en lugar de la MT, pero en nuestra opinión la simplicidad de la MT, la hace difícil de desbancar como modelo teórico.

Hemos visto que cuando considerábamos la complejidad temporal de las computaciones, mediamos el número máximo de pasos (el tiempo máximo) sobre todas las posibles entradas de una cierta longitud. Pero puede ocurrir que un problema o lenguaje sea reconocido en un tiempo rápido para casi **todas** sus entradas, excepto para un conjunto muy pequeño de éstas. Por ejemplo, en 1972, **Klee** y **Minty** demostraron que el SIMPLEX, el método clásico de investigación operativa, tenía algunas entradas para las que el costo era exponencial, aunque en la práctica es un método muy eficiente (127). Un modelo de máquina que toma en cuenta estas consideraciones es la **máquina de Turing probabilista** (PTM), que fue estudiada en 1972 por **J. Gill** (124). Previamente y de forma independiente **de Leeuw** (45) y **Rabin** (79), habían estudiado el modelo básico de autómata probabilista. La PTM de Gill es una MT que en un cierto momento y con una cierta probabilidad toma decisiones sobre el camino a seguir. Un lenguaje es aceptado por una PTM, con una cierta probabilidad. Gill da ejemplos de lenguajes reconocidos más rápidamente por PTM que por DTM. Si R denota la clase de lenguajes reconocidos por PTM se sabe que $P \subseteq R \subseteq NP$, pero no se conoce si las inclusiones son estrictas o no, aunque una vez más se conjetura que $P \neq R \neq NP$.

Adleman y **Manders** en un artículo posterior propusieron que en una nueva consideración los problemas tratables fuesen aquellos computados en tiempos polinómicos y con probabilidad de error muy pequeña, por una PTM (144).

El modelo de algoritmo probabilista ha sido aplicado con éxito al problema de la **primalidad:** el decidir, si un número es primo. Se conocía la pertenencia de este problema a la clase NP desde 1975 (138), pero todavía no se ha podido demostrar si es P o es NP-completo, aunque **Miller** ha dado una fuerte evidencia a favor de la hipótesis de la pertenencia a P (141). En 1977 de manera independiente, **Rabin** (146) y **Solovag-Strassen** (148), propusieron dos algoritmos probabilistas que con error muy pequeño decidían en tiempo lineal si un número es o no es primo.

Recientemente, se ha demostrado toda una serie de resultados sobre relativizaciones de la clases probabilistas. Para un resumen de los principales resultados consúltese la anteriormente mencionada tesis (157).

Otro modelo de máquina utilizado ha sido la **máquina de Turing alternante** (ATM) diseñada en 1977 de manera independiente por **Chandra** y **Stockmeyer** (139) y **Kozen** (140). Este es un modelo de máquina que combina el no determinismo y el paralelismo. La ATM es más potente que la NTM, lo que la convierte en un modelo óptimo para estudiar la jerarquía logarítmica, el equivalente a la jerarquía polinómica, pero entre las clases LOGSPACE y P.

Las medidas de complejidad que utilizan modelos de máquinas, son **medidas uniformes,** en el sentido de que son independientes de la longitud de la cadena de entrada, y consideran el tiempo o espacio necesarios para reconocer un lenguaje o computar una función. Existen otras medidas, las **no uniformes,** la principal de las cuales es el **costo Bodeano.**

Tomando como base los trabajos ya mencionados de **Shannon** (23) (36), **Lupanov** y sus escuelas utilizaron como medidas de complejidad el mínimo número de funciones bodeanas Λ, V ó necesarias para computar un problema. A esta medida se la denomina el costo combinacional. En 1965 Lupanov (55) utilizó medidas de complejidad de este tipo para calcular la complejidad estructural de ciertas funciones (99) (117). Este tipo de estudios abrió un nuevo campo, el de la **complejidad algebraica** que estudia el mínimo número de operaciones necesarias para realizar funciones algebraicas (desde la suma hasta evaluación de polinomios o evaluación de transformadas) (154), /KN.81/. En 1958, **Lupanov** (55) demostró que el costo combinacional de "casi todas" las funciones tenía una cota superior de tipo exponencial. **Nechiporuk** (116) y **Krapchenko** (121) demostraron que existían cotas inferiores del orden n^2 para el costo combinacional de unas clases bastante amplias de funciones. En los años 70, hubo un cierto interés por esta medida, motivada sobre todo por la esperanza de poder conseguir costos inferiores, exponenciales para algún problema en NP-completa y demostrar la desigualdad $P \neq NP$. Los

trabajos de **Savage, Happer** y otros son un exponente de los intentos realizados en esa dirección (129) y (137). Un buen resumen de esos trabajos es el libro de **Savage** (147). En 1976, **Schnorr** demostró para funciones finitas la equivalencia entre MT y circuitos combinacionales (142). Esto no significa que la clase P sea equivalente a la clase de funciones con costo combinacional polinómico, de hecho recientemente **Meyer** ha demostrado la equivalencia entre las funciones con costo combinacional polinómico y las computaciones de DTM con oráculo esparso (este resultado se puede encontrar en (152)). En la actualidad los circuitos combinacionales se utilizan como una herramienta muy valiosa para simplificar demostraciones que vía MT son difíciles de realizar.

Otra medida de complejidad, bastante antigua y que hoy se vuelve a reconsiderar, es la llamada **complejidad de Kolmogrov** introducida de manera independiente en 1965 por **Kolgmorov** (96) y en 1966 por **Chaitin** (100), con la intención de dar una significación precisa al concepto de **contenido aleatorio de una cadena finita.** La complejidad **Kolmogrov** de una cadena finita es la longitud del programa más corto que se puede computar a partir de esa cadena. Esta medida evalúa la cantidad de aleatoriedad o información que contiene dicha cadena. Recientemente se han definido y estudiado versiones en donde se acota el tiempo. Con este nuevo parámetro se consigue no sólo el clasificar las cadenas como aleatorias o no aleatorias, sino también el medir la cantidad de aleatoriedad detectable en cualquier paso de la computación. Esta observación permitió a **Hartmanis** el demostrar que toda computación con un alto grado de no determinismo es equivalente a una computación determinista seguida de otra computación polinómica no determinista, con una complejidad de Kolmogrov pequeña (158). De este resultado se desprenden una serie de consecuencias sobre la estructura de las computaciones así como la construcción de oráculos que separan clases de lenguajes. Otras aplicaciones de la medida Kolmogrov se pueden encontrar en el reciente artículo de **Sipser** (156).

.6. Conclusiones.

Las páginas anteriores han pretendido ofrecer un breve esbozo histórico de las áreas de investigación que se agrupan bajo el término **Informática Teórica.** El objetivo inicial fue tratar de encontrar procedimientos que sistematizaran el cálculo de ciertos problemas algorítmicos. Después, una mezcla de mentalidad escolástica y científica llevó a plantear los grandes problemas: a construcción de un lenguaje universal que permitiese expresar todos los problemas y la construcción de un procedimiento universal que permitiese computar todos los problemas expresados en dicho lenguaje. A principios del siglo XX se construyó el primer modelo de lenguaje universal y poco después se contestó negativamente a la posibilidad de encontrar un algoritmo universal. En la década de los treinta se estudiaron diferentes modelos de computación, así como la clasificación de problemas en **"computables"** y **"no computables"** y la subdivisión de éstos en diferentes grados de no computabilidad. A partir de los años sesenta el énfasis fue puesto en clasificar los problemas computables de acuerdo a su tratabilidad, es decir, en función de la existencia o no de algoritmos eficientes para computarlos. Esto llevó a estudiar propiedades estructurales de los problemas, tratando de discernir cuál es el motivo de que algunos problemas, los NP-completos, parezcan tener complejidad equivalente: exponencial para el determinismo y polinómica para el no determinismo. Paralelamente, en los años cincuenta también se desarrolló la teoría de los lenguajes formales, motivada sobretodo por la introducción de los lenguajes de programación y por los intentos de traducción automática de los lenguajes naturales. Mientras las teorías de la complejidad y de la recursividad consideran sobretodo la clasificación de los problemas o lenguajes a partir de la clase P, la teoría de lenguajes clasifica los lenguajes en clases por debajo de P, utilizando esquemas gramaticales o modelos de computación, más restringidos que la máquina de Turing.

A lo largo de este desarrollo histórico nos hemos ceñido a lo que consideramos **Informática Teórica,** una disciplina que crea y explora fundamentos teóricos en busca de las ideas que permitan un posterior desarrollo de los sistemas informáticos. Como ya hemos mencionado, la relación con otras áreas de la informática se estrechan, siendo, por ejemplo, difícil disociar los lenguajes formales de los lenguajes de programación, o el diseño de algoritmos de la complejidad. Disciplinas como la

semántica algebraica y la lógica de programas están a caballo entre la Informática Teórica y la Programación. Tradicionalmente, también en la distribución de la FIB, estas áreas caen dentro del campo de la Programación. Por otra parte, también hemos apuntado el hecho de que el desarrollo de la Informática Teórica coincide con el desarrollo histórico de algunas partes de la **Inteligencia Artificial.** Existe, no obstante, una diferencia metodológica y de objetivos entre ambas disciplinas, que ya explicitaremos más adelante.

También existen otros campos que no hemos mencionado y que tienen mucha relación con la complejidad: La **Criptografía** y el **diseño de algoritmos mixtos "hard-soft".** La criptografía ha sido muy influenciada por la Informática Teórica, pero se puede considerar más como una muestra de influencia pionera de la Informática Teórica, que como parte integrante de ésta. Respecto a los algoritmos paralelos híbridos (sistólicos, etc.), su estudio y la creación de modelos teóricos (ver por ejemplo el reciente artículo de Cook (152)), han comenzado muy recientemente.

La influencia de las Matemáticas no ha sido despreciable, y mirando a la historia expuesta se podría considerar la Informática Teórica como una rama de la Matemática. No hay que olvidar, sin embargo, un aspecto característico de la Informática Teórica: su fin es conseguir mejores sistemas de computación y clasificar los problemas de acuerdo a su dificultad en ser computados por diferentes modelos. La investigación en Infomática Teórica requiere crear y explorar conceptos teóricos hasta encontrar una conceptualización adecuada de la realidad que se pretende comprender, retrazando los restantes conceptos creados. Desgraciadamente, los "hábitos" de la Matemática pura han prevalecido demasiadas veces en Informática Teórica, y el campo está plagado de resultados y generalizaciones insignificantes desde el punto de vista informático. La Informática Teórica es ya un área llena de problemas importantes por resolver, algunos de los cuales han sido descritos en las páginas anteriores, y a medida que vayan produciéndose nuevos avances en informática, irán incrementándose. Es, por consiguiente, un área en su plenitud ytodavía muy lejos de la decadencia. Quisiéramos terminar este resumen con la opinión de J.Hartmanis sobre la historia reciente de la Informática Teórica:

"Cuando rememoro el desarrollo de la Informática Teórica, me vienen a la mente dos ideas casi contradictorias. Por una parte, estoy profundamente impresionado de lo bien que lo hemos hecho. El progreso en algunas áreas de la Informática Teórica ha sido mucho más rápido de lo que cabía esperar y muchos de los conceptos y resultados desarrollados han tenido un gran impacto en el posterior desarrollo de la Informática. Por otra parte, estoy sorprendido por la cantidad de cosas sin ningún valor que también se han desarrollado. La Informática Teórica, está plagada de innumerables artículos de dudosa calidad, artículos que ocultan su falta de profundidad detrás de oscuras formalizaciones matemáticas y que tratan problemas sin ninguna relevancia teórica ni práctica" (/HA.80/ pg. 50).

7. Bibliografía.

Las referencias utilizadas y mencionadas a lo largo del desarrollo anteriormente expuesto, serán divididas en dos partes. Las de uso general, que se refieren a la historia y que vendrán dadas por orden lexicográfico de los autores, y las referencias que son historia y como tales son mencionadas, que vendrán ordenadas cronológicamente, formando dicha lista un índice de los hechos mencionados anteriormente.

REFERENCIAS GENERALES

/BA.53/ BABINI, J.
"Historia sucinta de la matemática".
Espasa-Calpe (1.953)

/CO.83/ COLOMER, E.
"Ramón Llull ¿Precursor de la informática?"
L'Avenç 64.
Octubre 1983.

/DS.76/ DATTA, B.; SINGH, A.N.
"History of Hindu mathematics".
Dover 1976

/DA.70/ DAVIS, M.
"The Undecidable: Basic papers on undecidable propositions, unsolvable problems and computable functions".
Raven Press 1970

/FA.79/ FARRINGTON, B.
"Ciencia griega".
Icaria 1979.

/GL.80/ GENS, G.; LEVNER, E.
"Complexity of approximation algorithms for combinatorial problems: A survey"
SIGACT News, Fall 1980.

/GR.81/ GREIBACH, S.
"Formal Languages: Origens and directions".
Annals of the History of Computing, 3,1, 1981 14-41.

/HA.81/ HARTMANIS, J.
"Observations about the development of Theoretical Computer Science".
Annals of the History of Computing 3,1, 1981, 42-51.

/HA.78/ HATMANIS, J.
"Feasible Computatious".
CBMS-NSF Monograph, SIAM 1978

/HE.77/ HEIJENOORT, J.Van
"From Frege to Gödel"
Harvard Univ. Press 1977.

/HE.69/ HERMES, H.
"Enumerability, decidibility, computability"
2 nd. ed. Springer-Verlag 1969.

/HT.67/ HOLLINGDALE, S.H.; TOOTILL, G. C.
"Computadores electrónicos".
Alianza Editorial 43 1967.