AGORA —Papeles de Filosofía— (1999), 18/1: 107-132 *ISSN 0211-6642*

VIDA, OBRA Y ALGUNOS MILAGROS DE ALONZO CHURCH*

María Manzano
Universidad de Salamanca

Resumen

Este artículo está dedicado a Alonzo Church, que falleció en Agosto de 1995, tras una larga vida dedicada a la lógica. A él le debemos el cálculo lambda, la tesis que lleva su nombre y la solución al *Entscheidungsproblem*. Suyo es también *Introduction to Mathematical Logic*, vol. I, el libro que definió qué era la lógica, el enfoque y los temas básicos. Finalmente, él fue el impulsor del *Journal of Symbolic Logic*, la revista más conocida del área de la que fue editor durante décadas.

Este artículo se desarrolla en tres partes. La primera es periodística: cuento la vida de Alonzo Church, incluyendo algunas de las muchas anécdotas que me han ido llegando. La segunda está dedicada a su obra, aunque no he pretendido, en absoluto, ser exhaustiva. La tercera es algo más personal: pretendo mostrar que el gran descubrimiento de Church fue el cálculo lambda, y que a ese acierto inicial debemos muchas de sus posteriores inspiraciones, tanto como las de sus discípulos, incluyendo la Teoría de la Recursión de Kleene y la prueba de completud de Henkin. He añadido un apéndice en donde presento el cálculo lambda con tipos y una demostración de indecidibilidad de la lógica de primer orden.

Palabras clave: Church, *Entscheidungsproblem*, cálculo lambda, indecidibilidad de la lógica de primer orden.

Abstract

This paper is dedicated to Alonzo Church, who died in August 1995 after a long life devoted to logic. To Church we owe lambda calculus, the thesis bearing his name and the solution to the *Entscheidungsproblem*. His well-known book, *Introduction to Mathematical Logic*, vol. I, defined the subject matter of mathematical logic, the approach to be taken and the basic topics addressed. Church was also the creator of the *Journal of Symbolic Logic*, the best-known journal of the area, which he edited for several decades.

This paper is divided in three sections. The first is written in journalistic style: the story of the life of Alonzo Church is told, including some of the many anecdotes I have collected from different sources. The second part is devoted to his work, but is far from being exhaustive. The last part is more original; in it I attempt to show that Church's great discovery was lambda calculus and that his remaining contributions were mainly inspired afterthoughts in the sense that most of his contributions as well as some of his pupils derive from that initial achievement. Included are Kleene's Recursion Theory and the completeness proof of Henkin. I have added an appendix in which is presented the typed lambda calculus and a proof of the undecidability of first-order logic.

Keywords: Church, Entscheidungsproblem, lambda calculus, undecidability of first-order logic.

---

* La revista *Agora* y la autora agradecen el amable permiso para la presente publicación a Taylor & Francis Ltd. y a la revista *History and Philosophy of Logic*, en cuyo volumen 18 de 1997, pp. 211-232, apareció la versión orginal bajo el título «Alonzo Church: His Life, His Work and Some of His Miracles».

Aceptación: XII/1999

1. Inicio

Incluso no siendo unos fanáticos de la informática, sólo por el hecho de poder usar el correo electrónico, hemos de reconocer que el flujo de la información ha cambiado radicalmente nuestro ámbito en los últimos años. Si, como suele decirse, la información es poder, no cabe duda de que éste se ha repartido de forma impensable hace sólo unas décadas (Posiblemente no todo el poder se haya repartido, seguro que sólo lo menos importante.) Pensaba yo todo esto cuando recibía por métodos diversos, todos fulgurantes, información para realizar mi *Alonzo Church: in memoriam*. (Para hacer honor a la verdad y poner un contrapunto a este canto a las *autopistas de la información* he de decir que la fuente de información era León Henkin y John Addison en U.S.A y Wilfrid Hodges en el Reino Unido, a quienes respeto intelectualmente como se merecen y con los que tengo, además, la suerte de mantener una relación cordial pues han sido mis maestros. Quiero decir con esto, ¿habría podido de forma anónima haber abierto la espita de la información?)

¿Y Church?, ¿Cómo debió vivir estos cambios, él, que no sólo presenció casi un siglo de lógica, sino que tendría que sentirse responsable directo de su creación y de buena parte de su desarrollo?

Intentaba yo ponerme en la piel de Alonzo Church, pensando en el vértigo del salto que él había realizado, presenciado y favorecido por ser uno de los creadores más distinguidos (yo diría que el segundo en relevancia, sólo después de Gödel) de las dos disciplinas que a nivel teórico sustentan la informática: la lógica matemática y la teoría de la computación (reducibles a uno sola: la lógica matemática). Algunos dirían que su responsabilidad no sólo es a nivel teórico: Hoy se acepta[1], en general, que gracias a los trabajos de su famoso alumno Turing, pudo John von Neumann, a finales de los años cuarenta, diseñar el primer computador digital, electrónico.

Curiosamente, la respuesta me llegó casualmente unos meses después de formularla públicamente[2].

Pasé la primavera de 1996 en el CSLI de Stanford y un día, hablando con John Perry de Alonzo Church me comentó que John Etchemendy me contaría una anécdota divertida. Ocurrió en el curso 1983-84, habían invitado a Alonzo Church al CSLI para dar una conferencia y Etchemendy lo fue a recoger y le estuvo enseñando el centro. En aquel momento tenían bastantes ordenadores Xerox Dandelion, con procesadores LISP. Y puesto que dicho lenguaje está basado en el cálculo lambda de Church (extremo que parece negar su creador, John Mc Carthy, asegurando que fue un descubrimiento paralelo), John le hizo una demostración en una de aquellas Dandelion, explicándole que el lenguaje LISP está indirectamente basado en el cálculo

[1] Véase, Martin Davis [20].

[2] En Enero de 1996, en el *Congreso «Truth, Logic, Representation and World»* celebrado en Santiago de Compostela. Allí presenté «Alonzo Church: in memoriam» en una sesión dedicada a él.

lambda. Church no pareció entusiasmarse y al final, para justificar su evidente falta de interés, le comentó que él no sabía nada de ordenadores, pero que había tenido un alumno, llamado Alan Turing, que sabía bastante de eso.

2. Su vida

Fechas y datos relevantes. Alonzo Church nació en 1903 en Washinton DC; su padre, Samuel Robbins Church fue Juez de la Audiencia municipal del distrito de Columbia (Justice of the Municipal Court of the District of Columbia). Su abuelo fue bibliotecario del Senado de los Estados Unidos y su bisabuelo, que se llamaba como él, fue profesor de matemáticas y astronomía en la Universidad de Georgia.

Alonzo Church estudió en Princeton en donde fué discípulo de Oswald Veblen, un geómetra notable. En 1924 terminó su licenciatura, se casó con Mary Kuczinski en 1925, y en 1927 realizó su doctorado.

A continuciación Church pasó dos años con una *National Research Fellowship*; el primer año estuvo en Harvard, trabajando con Birkhoff (el algebrista) y con Huntington. En aquel momento en Harvard había una notable concentración de lógicos: Sheffer, Lewis y Whitehead pertenecían a dicha Universidad.

El segundo año lo pasó en Europa. Estuvo en Göttingen trabajando con los miembros de la escuela de Hilbert; el propio Hilbert y Bernays. Fue también a Amsterdam en donde estudió con el lógico intuicionista Brouwer.

A su regreso de Europa, en el otoño de 1929, comenzó a enseñar en Princeton. Durante diez años fue profesor ayudante, estuvo otros ocho como asociado y en el 1947 fue nombrado catedrático. Estuvo en Princeton hasta 1967 y entonces se fue a Los Angeles para evitar retirarse (ya que en Princeton el retiro era obligatorio a los 68). Estuvo en California como profesor en activo hasta 1990, ¡tenía 87 años!. El suyo era un contrato mixto: Flint Professor en Filosofía y profesor de Matemáticas, lo que le permitía desempeñar su labor docente en ambos departamentos; de hecho siempre se había mantenido en una posición semejante.

Veraneaba en las Bahamas, para estar tranquilo y poder pensar sin que nadie lo interrumpiera.

Fué un trabajador incansable toda su vida, y contó con un insaciable apetito intelectual hasta sus últimos años. En 1983, con motivo de su 80 cumpleaños, le hicieron una entrevista para el *UCLA Monthly* en la que declaraba[3]:

> «I will retire when one of two things happens: when I am told by the university I have to or when I'm told by my doctor I have to. I will continue as long as it can be done. It might be tomorrow, it might be a year from now, or it might be 10 years from now. But that's highly unlikely».

[3] Stuart Wolpert [1983], «A legend turns 80», *UCLA Montly*.

En esa entrevista Robert Yost, vicedecano del Departamento de Filosofía, a la pregunta de por qué Church es tan bueno contesta:

> «He is just smarter than anybody else»,

y añade que se lo lee todo y no olvida nada.

Por aquella época continuaba haciendo a pie el recorrido entre su casa y el Departamento. No quiso nunca tener coche porque no quería tenerle que dedicar el mismo tiempo que a sus clases y a sus investigaciones, dijo en la mencionada entrevista.

Se fue finalmente a Hudson en 1992 para vivir cerca de su hijo Alonzo Church Jr. Murió el 11 de Agosto de 1995. La nota necrológica del *New York Times* del 5 de Septiembre de 1995, escrita por Nicholas Wade, se titulaba: «Alonzo Church, 92, Theorist of the limits of Mathematics». La del *Independent* de Londres que escribió Wilfrid Hodges viene acompañada de una foto con el siguiente pie: «A truly rational mind».

Dice Henkin que Church mantuvo vivo su interés por la lógica hasta el final ya que en su mesilla de noche, al morir, tenía un volumen de homenaje a su persona, con varios puntos de libro situados en varias de las contribuciones, lo que muestra que lo estuvo hojeando y que se interesó por algunos de los escritos.

Dicho volumen lo compilaron Anthony Anderson y un estudiante suyo, Mikhail Zeleny. Contribuyeron, entre otros, Shepherson, Shapiro, Paulson, Meyer, Kripke, Henkin y Feferman. Pretendía ser un regalo de cumpleaños por su 90 aniversario y no se pudo publicar por falta de editor. (El volumen que Church tenía en la mesilla era una encuadernación manual.) Henkin cuenta que en vista de las dificultades para encontrar editor, él mismo sugirió a Antony Anderson que le regalaran la encuadernación manual; Church murió unas semanas después. Dos de los artículos de este volumen, el de Martin Davis [21] y el de Henkin [31], se publicaron en el *Bulletin of Symbolic Logic*.

Docente. Una carrera docente de 61 años es de por sí extraordinaria, pero sus discípulos aseguran que su estilo particular hacía su docencia inolvidable y que dejó una profunda huella en todos ellos, aunque como docente era modesto, paciente y poco expresivo. (Algunos le achacan carencia de pasión.) Esta parece ser también la opinión de Alan Turing, que pasó dos años en Princeton como alumno suyo y que hizo la tesis con él[4].

Según parece su trabajo era siempre muy minucioso, su análisis de gran calado y los detalles cuidados al extremo, pues no había detalle irrelevante para él.

David Kaplan dice que él recomendaba a sus alumnos que hicieran un curso con Church asegurándoles que los cambiaría y que, incluso si no

[4] Ver Hodges [35], p. 119.

estaban muy interesados en el tema del curso, siempre podrían contar a sus nietos que habían sido alumnos de Church.

Henkin [31] cuenta que en 1941, como parte del programa de doctorado, hizo un curso de Lógica con Church cuyo programa era el siguiente:

1. En el primer semestre se estudiaban varios sistemas de lógica proposicional y de primer orden, se introducía el concepto de forma normal, se demostraba completud y se discutían los teoremas de Löwenheim-Skolem. La prueba de completud era la de Gödel (la de Henkin, obviamente, no existía) y se remarcaba la naturaleza constructiva de la prueba.

has been used by some in connection with simple type theory.) Using my own notation I prefer to write this principle of extensionality as

$$\varphi(x) \equiv_x \psi(x) \supset \varphi = \psi$$

Now if we write Sir Walter Scott is the author of Waverley as $(\iota z)S(z) = (\iota x)A(x, (\iota y)W(y))$ the result of eliminating the descriptions is

$$(\exists x_1).\ S(z) \equiv_z z = x_1.\ (\exists y_1).$$
$$(\exists z_1)[W(y) \equiv_y y = z_1.\ A(x, z_1)] \equiv_x x = y_1.\ x_1 = y_1$$

Thus on Russell's theory of descriptions King George wished to know whether Sir Walter Scott is the author of Waverley must be rewritten as:

King George wished to know whether
$(\exists x_1).\ S(z) \equiv_z z = x_1.\ (\exists y_1).(\exists z_1)[W(y) \equiv_y y = z_1.$
$A(x, z_1)] \equiv_x x = y_1.\ x_1 = y_1$

If we let ψ be the propositional function corresponding to $(\exists z_1)[W(y) \equiv_y y = z_1.\ A(x, z_1)]$ and φ the propositional function corresponding to $S(z)$, this becomes.

King George wished to know whether
$(\exists x_1).\ \varphi(z) \equiv_z z = x_1.\ (\exists y_1).\ \psi(x) \equiv_x x = y_1.$
$x_1 = y_1$

Then because we know that as a matter of fact $\varphi(x) \equiv_x \psi(x)$, there follows by the principle of extensionality for propositional functions and by substitutivity of equality:

King George wished to know whether
$(\exists x_1).\ \varphi(z) \equiv_z z = x_1.(\exists y_1).\ \varphi(x) \equiv_x x = y_1.$
$x_1 = y_1$

2. En el segundo semestre se estudiaba un sistema de segundo orden y, en particular, se introducía la Aritmética de Peano con gran detalle y se demostraban los resultados de incompletud de la aritmética y de la lógica de segundo orden. Para demostrar incompletud se introducían las funciones

recursivas, pero sólo las primitivas. Las funciones recursivas generales no se estudiaban, pero se mencionaba el papel que juegan en los casos en los que queramos establecer que no existe un procedimiento de decisión.

Según Henkin aunque el contenido del curso no era, como habeis visto, en absoluto sorprendente, sí que lo era el estilo de su maestro, la manera que tenía de trasmitir su concepción de la lógica. Hacía, según parece, paradas frecuentes para aclarar que se seguía el *método logístico*; delimitar claramente qué era lenguaje y metalenguaje, cómo el lenguaje formal había de ser establecido de manera completamente efectiva y porqué el metalenguaje (Inglés, en su caso) había de limitarse. Uno puede hacerse una idea de cómo eran sus cursos leyendo su manual, el libro *Introduction to Mathematical Logic*, vol. I.

Tesis doctorales dirigidas. A lo largo de su dilatada vida Church dirigió 31 tesis doctorales, sus discípulos son los siguientes:

Alfred Foster, Alan Turing, Stephen Kleene y Barkley Rosser, en los años 30.

Enrique Bustamante, Leon Henkin y John Kemeny, en los años 40.

Martin Davis, Dana Scott, Hartley Rogers, Norman Shapiro, Maurice L'Abbé, Willian Boone, Thacher Robinson, Michael Rabin, Simon Kochen, Aubert Daigneault y Raymond Smullyan, en los 50.

Peter Andrews, Robert Ritchie, James Guard, James Bennet, Robert Winder, Wayne Richter, Gustav Hensel, Willian Easton, Joel Robbin y Donald Collins, en los 60.

Richard Malitz y Anthony Anderson, en los 70.

Gary Mar, en los 80.

La útima tesis la dirigió cuando sobrepasaba los 80. (Henkin recuerda a un alumno mexicano, llamado Zubieta, del que duda si hizo la tesis con Church en los 50. Esta persona aparece en el prefacio del libro de Church, *Introduction to Mathematical Logic*; se le agradece el haber confeccionado el índice y el haber detectado errores.)

Su Persona. Todos lo describen como una persona amable, de buen humor y muy, muy ponderado en sus juicios. Algunos incluso se extrañaban de que un genio como él pudiera ser un ser humano tan equilibrado.

(Yo me imagino que durante muchos años tuvo que soportar la comparación con el también profesor en Princeton, Kurt Gödel, de quien no puede decirse, ni remotamente, que fuera un ser equilibrado. Estando yo en Berkeley, durante el curso 1977-1978, murió Gödel, y circularon por allí algunas anécdotas suyas. La más simpática, otras eran algo patéticas, era de cuando iban a darle la nacionalidad americana. Tenía que entrevistarse con un funcionario del gobierno e hizo a Einstein que lo acompañara porque, según parece, estaba asustado pues pensaba que posiblemente le preguntarían su opinión acerca de la Constitución americana, que él se había leído concienzudamente, y, si tenía que ser sincero y leal, habría de declarar que estaba plagada de contradicciones).

De Church, no circulan anécdotas tan divertidas, pues él era poco estridente; no sólo racional, sino también razonable. Era una persona de gran corpulencia y enorme tamaño (incluso su letra era grande), que se cuenta usó en alguna ocasión para poner un muro entre un orador implacable y su exhausta audiencia, cumpliendo así su labor de moderador de una conferencia que se alargaba en exceso.

Una foto suya aparece en el artículo que a él dedica la *Encyclopedia of Computer Science and Engineering*. Otra foto aparece en el libro de Barendregt sobre el cálculo lambda.

Una de las anécdotas que también se cuenta en la nota necrológica que escribió Hodges es que en una reunión de evaluación con otros profesores aceptó aprobar a un alumno con un trabajo deficiente a condición de que se comprometiera a no escribir ningún artículo sobre la relevancia filosófica del Teorema de Gödel, yo subscribiría una propuesta semejante.

Church estuvo casado durante 50 años, su esposa murió en 1976, y tuvo tres hijos: Alonzo, Mary Ann y Mildred. La segunda de ellas está casada con John Addison, el reconocido lógico de Berkeley, y viven cerca del Cerrito.

Profesor Visitante. Además de desempeñar la función de catedrático de Filosofía y Matemáticas en Princeton y Los Angeles, como profesor visitante estuvo en:

1. Universidad de Illinois (1956-57)
2. Universidad de Berkeley (primavera de 1960)
3. Universidad de California en Los Angeles.(1960-61)

Nombramientos honoríficos. En 1967 fue elegido miembro de la Academia americana de Artes y Ciencias y de la Academia Internacional de Filosofía de la Ciencia.

En 1978 fue nombrado miembro de la Academia Nacional de Ciencias y de la Academia Británica.

En 1969 fue investido *Doctor Honoris causa* por la Case Western Reserve University, en 1985 en la de Princeton y en 1990 lo hizo la State University of New York en Buffalo. (El día antes de recibir este grado honorífico, se celebró en dicha universidad un simposio en su honor, que organizó John Corcoran.)

Journal of Symbolic Logic. Cuando en 1935 se fundó la *Association for Symbolic Logic*, Church se convirtió en editor de su revista, el *Journal of Symbolic Logic* y estuvo a cargo de la sección de crítica hasta 1979. Dicha revista alcanzó unas cotas elevadas de objetividad y durante muchos años fue *la* revista de Lógica por antonomasia.

Muchos afirman que como editor de su sección de crítica era muy estricto; jamás permitió que nadie se excediera en sus comentarios. Cuentan que él mantenía que una crítica podía ser muy negativa sin ensañamiento y, sobre todo, no permitía la mordacidad ni la ironía, pues estaba convencido de que

en esto no se puede dar marcha atrás, ni deja espacio para una posible defensa.

Según David Kaplan, la opinión de Church al respecto era:

> «Evaluations can be rebutted, and errors can be corrected, but sarcasm can neither be rebutted nor corrected».

También dicen que los que se retrasaban en mandar sus comentarios recibían *the Bomb*, una carta demoledora.

Church se mantuvo al corriente de las publicaciones en lógica durante muchos años y en el Journal aparecía una magnífica sección bibliográfica puesta al día periódicamente.

3. Su Obra

3.1. Primeros escritos

El primer artículo de Church [4] se publicó en 1924 y el último [4] en 1995.

En su tesis doctoral Church se propone hacer en la Teoría de conjuntos lo que otros hicieron al crear la geometría no euclídea. De la misma manera que Lobachevsky consideró el axioma de las paralelas como un postulado cuya aceptación podía ser cuestionada, y, por lo tanto, digna de consideración la teoría axiomática resultante al añadir su negación al resto de los axiomas de Euclides, Church hizo lo propio con la teoría de conjuntos de Zermelo. Church propuso diferentes axiomas, que contradicen el axioma de elección, como alternativas a la axiomática de Zermelo.

En su A. Church [1926], «Alternatives to Zermelo's assumption», *Transaction of the American Mathematical Society*, vol. 29, pp. 178-208, se planteaba como cierta la posibilidad de que dicha axiomática no constituyera una teoría completa, incluso añadiendo el axioma de elección, pues pensaba que caso de serlo debería ser posible construir una función F del tipo requerido por el axioma de elección. Sin embargo, la naturaleza no constructiva del axioma de elección impedía, claramente, dicha posibilidad. Algunos años después, al crear su cálculo lambda, intentaría derivarlo sin éxito: tenía que ser postulado como axioma. Esto contrastaba con lo que sucedía con otros axiomas de la teoría de Zermelo que postulaban la existencia de conjuntos para los que el cálculo lambda no necesitaba más que sus propias reglas de formación de expresiones para asegurar su existencia.

En un artículo de 1928, claramente bajo la influencia de su maestro Brouwer, A. Church [1928], «On the law of the excluded middle», *Bulletin of the American Mathematical Society*, Vol. 34, pp. 75-78, Church se plantea la posibilidad de eliminar la ley del tercio excluso. En general, su actitud hacia los principios de la lógica no es dogmática, pues su postura está abierta a considerar variantes de los sistemas lógicos obtenidos al modificar dichos

principios. Frente a esto reaccionaron fuertemente algunos lógicos, Langford entre otros, considerando que la lógica es el sistema de las proposiciones que expresan hechos analíticos de naturaleza formal y que por lo tanto no puede haber más que un sistema, *el sistema*.

Ese mismo año se plantea con precisión el problema de la decidibilidad de la lógica de primer orden, considerándose crucial, y unos años más tarde tanto Church como Turing (según parece, independientemente) lo resuelven negativamente. Como veremos en la sección 4, es justamente este problema, el décimo de Hilbert, lo que hace que ambos se vean forzados a precisar y definir la noción de computabilidad efectiva.

3.2. Introduction to Mathematical Logic, vol. I

El libro de Church, *Introduction to Mathematical Logic*, vol. I, es el libro de lógica por antonomasia, el que definió qué era la lógica, explicitó el enfoque que la caracterizaría y los temas básicos durante varias generaciones. El segundo volumen fue escrito sólo en parte pero circuló como manuscrito mecanografiado, pues estaba depositado en la biblioteca de la Universidad de Princeton. Una versión previa del manual apareció en 1944 como volumen de *Annals of Mathematical Studies* y en 1946 ya estaba el manuscrito depositado en la biblioteca.

Este libro, cuyo índice se corresponde con el curso que Henkin dice haber recibido[5], tiene un capítulo introductorio, de 70 páginas, que es un auténtico ensayo. En una carta personal, León Henkin me comenta al respecto:

> «I have several times in recent decades re-read the long Introductory chapter of the book, and each time was excited by the richness of its ideas. It contains the basis for much future work by logicians, I am sure».

Se hace allí un análisis muy agudo e inteligente de los supuestos ontológicos y semánticos que asumimos al elegir la lógica clásica. Y también se suministran las razones que justifican la aceptación de dichos principios. Una parte importante de dicho capítulo de introducción se dedica a la presentación y esclarecimiento de los conceptos matemáticos y lingüísticos que constituirán los pilares del análisis lógico: nombres, funciones, constantes, variables y proposiciones. Se estudian también otros propiamente lógicos; a saber, los conectores y cuantificadores. La obra de Frege es usada y comentada en esta parte, haciéndose hincapié en la distinción entre sentido y referencia. De especial interés es el apartado 07 de dicha introducción, que está precisamente dedicado a la exposición del método logístico; es una joya. En realidad allí lo que se hace es explicitar los principios que necesariamente aceptamos cuando hacemos lógica formal; principios que ahora no nos sor-

[5] Esto no es completamente cierto, pues el capítulo V, dedicado a la lógica de segundo orden, contiene la prueba de completud que su alumno hizo a finales de los años cuarenta.

prenden, como el de la distinción estricta entre lenguaje y metalenguaje, pero que en aquel momento tenían gran importancia pues se estaba aún saliendo de una fase de confusión. También establece con claridad cuáles son los requisitos de *efectividad* del lenguaje y del sistema lógico. Dichos requisitos se exigen:

1. Al especificar los signos primitivos (dado un signo hay que poder determinar si es o nó un signo primitivo).
2. Al definir fórmulas (dada una sucesión de signos hay que poder determinar si es o nó una fórmula).
3. Al especificar los axiomas (dada una fórmula hay que poder determinar si es o nó un axioma).
4. Al definir las reglas de inferencia (cuando se propone una fórmula como conclusión de una inferencia inmediata se debe poder determinar si es correcta, conforme a las reglas).

Como el propio Church señala, el que se cumplan estas condiciones nos permite afirmar que el concepto de prueba es efectivo en el sentido de que haya un método mediante el cual, dada una secuencia finita de fórmulas, podamos determinar si es o nó una prueba. Insiste en la necesidad de contar con procedimientos de prueba efectivos, pues lo contrario significaría que siempre se puede solicitar una prueba de que la prueba dada lo es de hecho, y ello nos llevaría a una infinita recurrencia. Añade que la prueba de un teorema del lenguaje objeto debe hacerse sin recurrir a la interpretación, sin usar la semántica, y que hay tres razones para que así sea: (1) la distinción entre forma y contenido; un razonamiento lógico es correcto por su forma, (2) la demostración ha de usar exclusivamente los axiomas y las reglas de inferencia y (3) que los sistemas lógicos, por usar recursos matemáticos y lingüísticos limitados son mucho más seguros que las interpretaciones dadas en la semántica, en donde se involucran más recursos. Hoy diríamos: se requiere más potencial de teoría de conjuntos para la semántica que para el cálculo (aceptando que la teoría de conjuntos es nuestro substrato matemático).

Sin embargo, la noción de ser un teorema no es necesariamente efectiva en el sentido de que tenga que existir un método mediante el cual, dada una fórmula, nos permita determinar si es o deja de ser un teorema.

En el libro se insiste en que el concepto de efectividad utilizado es el intuitivo; es la noción de algoritmo usada comúnmente en matemáticas (por ejemplo, el algoritmo de Euclides para hallar el máximo común divisor, o la criba de Eratóstenes para determinar los números primos menores que uno dado). Cuando tenemos un algoritmo que determina si se cumple o nó una cierta propiedad, el conjunto o relación definido por ella se denomina decidible. Si el algoritmo determina los valores de una función, a ésta se la llama efectivamente computable (o simplemente, computable). Church resalta el hecho de que para distinguir métodos efectivos de los que no lo son es

suficiente con el concepto intuitivo, y en una nota a pie de página[6] se indica que la definición precisa está en un artículo del autor y se dan las referencias de los artículos de Turing, Post, Kleene, Church y Hilbert & Bernays, que introducen otras definiciones y en donde se prueban las equivalencias pertinentes.

La descripción intuitiva del concepto de procedimiento efectivo (o, equivalentemente, de función efectivamente computable) es suficiente cuando lo que se quiere es establecer que una función es, de hecho, computable. En el libro se muestra que la lógica proposicional es decidible (apartado 15: «Tautologies, the decision problem») simplemente aclarando que las tablas de verdad constituyen un procedimiento de decisión[7]. Para la lógica de primer orden no se demuestra indecidibilidad; se prueba, eso sí, que hay ciertos conjuntos decidibles de fórmulas de primer orden[8]. De hecho, en el libro no se puede demostrar indecidibilidad porque no se ha definido matemáticamente el concepto de procedimiento efectivo o algoritmo que permita demostrar que un conjunto sea indecidible o una función no computable.

Es curioso que en el libro no se desarrolle ninguna de las tres grandes contribuciones de Church a la lógica y a la informática: el cálculo lambda, la tesis que lleva su nombre y la demostración de la indecidibilidad de la lógica de primer orden. Yo creo que esta decisión fue tomada sobre la base de acertadas consideraciones pedagógicas.

No se puede acabar hablando de este libro sin comentar los suculentos apartados históricos y las notas a pie de página. Dichas notas son una muestra de lo extraordinariamente detallista que era Church, de la finura de su análisis, de su erudición, y también de cómo y hasta qué punto su libro era producto de una superposición de reescrituras. Hoy en día, con los tratamientos de texto desaparecen las trazas de las distintas versiones precedentes, y con la aversión de los editores a las notas a pie de página, solicitándonos que en todo caso pasen al final, se están prácticamente extinguiendo. El libro de Church, cargado de notas, me encanta, especialmente el larguísimo capítulo de Introducción, y estoy de acuerdo con Henkin en que hay todavía en él material para trabajos futuros, especialmente para

[6] En la página 52 del libro, la nota 119 que reproduzco a continuación: «For a disscussion of the question and proposal of a rigorous definition see a paper by the present writer in the *American Journal of Mathematics*, vol. 58 (1936), pp. 345-363, especially §7 thereof. The notion of efectiveness may also be described by saying that an effective method of computation, or algorithm, is one for which it would be possible to build a computing machine. This idea is developed into a rigorous definition by A. M. Turing in the *Proceedings of the London Mathematical Society*, vol. 42 (1936-1937), pp. 230-265 (and vol. 43 (1937), pp. 544-546). See further: S. C. Kleene in the *Mathematische Annalen*, vol. 112 (1936), pp. 727-742; E. L. Post in the *Journal of Symbolic Logic*, vol. 1 (1936), 103-105; A. M. Turing in the *Journal of Symbolic Logic*, vol. 2 (1937), pp. 153-163; Hilbert and Bernays, *Grundlagen der Mathematik*, vol. 2 (1939), Supplement II».

[7] Esta es la prueba de Post [48].

[8] La decidibilidad de la lógica de predicados monarios, tanto la de primer orden como la de segundo orden, es un resultado de Löwenheim [45].

los interesados en una filosofía de la lógica rigurosa, que no dé la espalda a la tradición fregeana.

3.3. La Bella y la Bestia (Lógica e Informática)

Cuando digo que Church debía sentirse responsable directo de la base teórica sobre la que se sustenta la Informática estoy pensando en las tres contribuciones de Church arriba mencionadas, fruto de su trabajo en los años treinta, y que han tenido un impacto decisivo en esta disciplina. Todas ellas directamente relacionadas con el concepto de computabilidad (o recursión), que es, obviamente, el concepto central de la informática.

Informática. La informática trata de los computadores y de la computación. A la informática le concierne la invención y análisis de los algoritmos, el diseño de programas y de lenguajes de programación para expresar algoritmos y la construcción de sistemas de computación, máquinas incluídas, para ejecutar o implementar los lenguajes de programación.

Lógica. A la lógica se le atribuye como principal objetivo el análisis y codificación del razonamiento; le interesa el significado y la transformación de los enunciados que empleamos en nuestro discurso sobre el mundo.

En lógica se define con precisión un lenguaje artificial, se atribuye significado a los enunciados del lenguaje mediante modelos matemáticos y se define un cálculo; esto es, un sistema de axiomas y reglas que rigen la transformación de los enunciados del lenguaje formal. Dependiendo del nivel de abstracción que vayamos a necesitar y de la realidad a tratar hay varios tipos de lógica.

Por otra parte, ha sido en la lógica matemática en donde se ha desarrollado la teoría de la computabilidad, definido el concepto matemático de recursividad y caracterizado matemáticamente el poder de los algoritmos. Se hizo en los años 30, antes de la aparición de los ordenadores, pero si no se hubiera hecho entonces se habría hecho más tarde, en informática.

La Lógica en la Informática. La lógica aparece en cualquiera de los niveles en los que nos situemos para describir tanto el funcionamiento de un computador como la ejecución de un programa en una máquina. Aparece al nivel inferior, implicada en el hardware; aparece en la cascada de lenguajes que desde el usuario bajan al lenguaje máquina, el que el ordenador entiende; aparece también cuando, situados en un nivel metateórico, razonamos acerca de los programas; es decir, cuando hacemos lógica de programas.

Pues bien, las contribuciones de Church inciden directamente en todos estos niveles. No sólo define el concepto matemático de recursividad, sino que nos proporciona herramientas para crear un lenguaje de programación y establece también, claramente, la frontera de la computabilidad. Yo creo

que una parte importante de estos resultados hay que atribuírselos a la enorme fortuna del lenguaje y cálculo λ que brevemente presentaré al final de este artículo.

4. Algunos Milagros de Church

En lo que sigue intentaré hacer plausible mi tesis de que el gran descubrimiento de Church fue, sin lugar a dudas, el del cálculo lambda, y que el resto de sus contribuciones, así como las de algunos de sus discípulos se deben a este acierto inicial. El hilo argumental será éste:

4.1. El cálculo lambda. En las primeras décadas de este siglo seguramente se consideraba imprescindible para un lógico que se preciara el presentar un sistema lógico propio. Church creó el cálculo lambda tomando como base el concepto de función y distinguiendo claramente entre el valor de una función para un argumento, $F(x)$, y la propia función, $\lambda x F(x)$. Para ilustrarlo pongamos uno de los ejemplos que el propio Church usa[9]:

> «If we say, (x^2+x) *is greater than 1,000* we make a statement which depends on x and actually has no meaning unless x is determined as some particular natural number. On the other hand, if we say (x^2+x) *is a primitive recursion function*, we make a definite statement whose meaning in no way depends on a determination of the variable x (so that in this case x plays the rôle of an apparent, or bound variable). The difference between the two cases is that in the first case the expression (x^2+x) serves as an ambiguous, or variable, denotation of a natural number, while in the second case it serves as the denotation of a particular function. We shall hereafter distinguish by using (x^2+x) when we intend an ambiguous denotation of a natural number, but $(\lambda x(x^2+x))$ as the denotation of the corresponding function —and likewise in other cases».

El elegir el concepto de función como base fue ya un gran acierto, pero el distinguir en el lenguaje los valores de la función de la función misma, lo fue aún mayor. Si a esto añadimos la característica de que la formalización permite que una función anide a otras, el lenguaje está preparado para expresar la recursión. De hecho, a diferencia de los sistemas axiomáticos fundacionales, la existencia de los objetos matemáticos (en nuestro caso, funciones) no se postula, sino que son generados por las propias reglas de formación de expresiones.

Este lenguaje es tan compacto y tan adecuado para expresar funcionalidad que se convirtió en lenguaje de programación. Como ya comenté anteriormente, se atribuye la inspiración para crear el lenguaje LISP, desarrollado por John Mc Carthy, al cálculo lambda.

El cálculo lambda, aunque llevaba circulando varios años, no aparece en prensa hasta principios de los cuarenta[10]. A comienzos de los años treinta

[9] Véase [10], p. 6 de la edición de 1951.

[10] El pequeño volumen [10] es interesante. Una presentación moderna se encuentra en el libro de Hindley [33] y en el de Barendregt [2].

Church ya sabía que tanto la función del siguiente como la suma eran λ-definibles, pero fue su alumno Kleene quien fue gradualmente descubriendo lo amplia que era la clase de las funciones definibles mediante lambda y estos resultados constituyeron parte de su tesis. Este es el origen de la teoría de la recursión. Sin embargo, los resultados de la investigación les llevaron más lejos de lo esperado. Veamos lo que dice Martin Davis [19] al respecto:

> «Church published a pair of substantial papers on the system he developed and set his students Stephen C. Kleene and J. Barkley Rosser to work on it. Their work was extremely effective, if not exactly what one dreams of having one's graduate students accomplis for one: Kleene and Rosser proved that Church's system was inconsistent!».

Pese a ese tropiezo inicial, pronto se vió que la definibilidad mediante lambda equivalía a la recursividad y a la computabilidad mediante máquinas de Turing, proporcionando así un argumento de gran fuerza a favor de la denominada tesis de Church; es decir, la de tomarlas como una definición posible de computabilidad efectiva o algoritmo. Además, usando la mencionada tesis, Church demostró la indecibilidad de la lógica de primer orden.

Veremos más adelante cómo el cálculo lambda sirvió también de base a la Teoría de tipos, produciendo una presentación muy elegante de la misma. Y no sólo eso, dió origen a la prueba de completud de Henkin, la que seguimos actualmente en cualquier lógica, no sólo en la teoría de tipos con lambda.

4.2. Entscheidungsproblem. Uno de los temas que más interesaba en aquel momento (estamos en los años treinta) era el de la decidibilidad. Hilbert y Ackerman [34] lo expresan así:

> «El *Entscheidungsproblem* se resuelve cuando uno conoce un procedimiento que le permite decidir, mediante un número finito de operaciones, acerca de la validez, respectivamente la satisfacibilidad, de una expresión lógica».

Este problema estaba en el centro de los intereses de la escuela de Hilbert, quienes por aquel entonces esperaban una solución positiva del famoso problema décimo. Pronto, sin embargo, se pasó a esperar una respuesta negativa pues ya se veía que en caso contrario traería consecuencias poco plausibles. En particular, tras la demostración de Gödel de la incompletud el panorama y las expectativas eran completamente diferentes[11].

Como comenté anteriormente, mostrar que un conjunto es decidible no requiere más que señalar el algoritmo que lo genera. Y puesto que la noción intuitiva basta para identificar a un procedimiento como efectivo, puede hacerse sin una definición matemática de computabilidad. Sin embargo, para demostrar que no es decidible no basta con tener una noción intuitiva, se necesita una definición en regla.

[11] Gandy [24] expone maravillosamente cuál era la situación.

O sea, que del deseo de resolver problemas de indecidibilidad se derivaba la necesidad imperiosa de contar con una definición matemática de computabilidad. Y en particular, el resolver el famoso problema décimo de Hilbert pudo muy bien servir de acicate en Church para ponerse a la tarea de proporcionar tal definición[12]. Puesto que las investigaciones de Kleene y las suyas propias habían mostrado el enorme poder expresivo del lenguaje introducido por Church, parecía razonable pensar que una tal definición podía ser precisamente la λ-definibilidad.

Por lo que al *Entscheidungsproblem* se refiere, Church demostró que para la lógica de primer orden no es posible encontrar un algoritmo que determine si una fórmula es o nó válida. El artículo en el que Church prueba este teorema [8] apareció en 1936 y está basado en un artículo suyo del mismo año [7] y en la gödelización[13]. (Al final de este artículo he añadido una prueba de indecidibilidad de la lógica de primer orden; el argumento es el de Church, la presentación es la común actualmente).

4.3. Computabilidad: recursividad. La Teoría de la recursión clásica abarca el estudio de las funciones definidas sobre los naturales, pero la teoría de la recursión actual, al ir desarrollando su potencial propio y sus métodos específicos, ha alcanzado un notable desarrollo abstracto y aplicaciones insospechadas. Los orígenes de la teoría clásica pueden hallarse en Dedekind, cuando en 1888 introduce el estudio de las funciones definibles sobre el conjunto de los números naturales usando ecuaciones y, recurrentemente, la inducción sobre los naturales que él había formulado y precisado. De ahí viene, justamente, el que se adoptara el nombre de Teoría de la recursión.

Por lo que respecta a su estadio presente , cuyo radio de acción cubre la totalidad de las funciones efectivamente computables, los orígenes hay que buscarlos en el grupo de Princeton; empezó con Church, pero si hay que atribuirle un padre, éste es Kleene. El fue quien la impulsó, definió y acotó. Suyos son los teoremas de la forma normal y el de la recursión[14].

En cuanto a la definición misma, comentamos con anterioridad que circulaban varias versiones de este concepto, aunque, como veremos en el siguiente apartado, había cierta resistencia a aceptarlas como definición. Varios de estos conceptos aparecieron en los años 30 para caracterizar nociones que en un principio parecían diferentes: la primera era la caracteri-

[12] Sabemos que fue tambien ésta la razón que movió a Turing a definir el concepto de computable mediante una máquina abstracta; sus famosas *máquinas*.

[13] También se dice que el teorema de incompletud, caso de no haber sido probado por Gödel, podría haberse derivado de los resultados de Church; veamos lo que dice Shoenfield [51] en una nota a pie de página al respecto: «Kleene has pointed out that if Gödel had not already proved his Incompleteness Theorem, Church might well have derived it from his results. The reader interested in such a derivation of the Incompleteness Theorem can consult my book *Mathematical Logic*. Of course, this takes nothing away from Gödel's supreme achievement in discovering the Incompleteness Theorem».

[14] Davis [18] es una antología de los artículos básicos de este campo. En Kleene [43] se hace un estudio histórico de los inicios.

zación de Gödel de las funciones definidas mediante recursión, la segunda era la noción de función definible con el operador λ, que Church y Kleene introdujeron, y la tercera, era la de función computable mediante una máquina abstracta, las máquinas de Turing. Pronto se demostró que los tres nociones definían las mismas funciones. Si bien estaba claro que todas las funciones definidas mediante cualquiera de los procesos anteriores era efectivamente computable, este resultado seguía sin servir para demostrar que un conjunto es indecidible; necesitamos la conversa. Por su propia naturaleza esta afirmación no puede ser demostrada, y si no se considera adecuado que sea una definición, hay que proporcionarle otro *status*.

4.4. Tesis de Church. Otro de los grandes méritos suyos es el haber formulado la tesis de que el concepto matemático preciso de recursión, tal y como lo había definido Turing, se correspondía exactamente con el concepto intuitivo de computabilidad efectiva.

Shoenfield [51] dice que la tesis de Church surgió de manera casual, yo diría algo muy parecido, que surgió de forma natural. Habiendo introducido su sistema lambda y estando interesado en el alcance de las funciones λ-definibles, se fue dando cuenta gradualmente (como dije, en parte gracias a su alumno Kleene) de que dicha clase era muy, pero que muy extensa. Estaba claro que las funciones λ-definibles eran computables y parecía razonable pensar que todas las computables eran λ-definibles. Es decir, que la noción intuitiva de computabilidad (o efectividad) se correspondía exactamente con la de λ-definibilidad. En un principio la formuló como definición y luego Kleene la cambió al *status* de tesis, con el que la conocemos actualmente. En su primera formulación computabilidad efectiva fue identificada con λ-definibilidad, públicamente fue anunciada como recursividad y más tarde fue generalmente aceptada como computabilidad mediante máquinas de Turing. Las vicisitudes de la tesis pueden verse bien en Davis [19].

Aunque por tratarse de una tesis no se puede demostrar matemáticamente, hay razones poderosas para aceptarla:

1. La primera es una razón basada en la experiencia acumulada: todos los algoritmos conocidos son recursivos. Gandy [24] añade al respecto:

> «Kleene's dissertation shows, as we might now put it, the power of the λ-calculus as a high-level programming language. A λ-term can take other λ-terms (subroutines), instead of numbers, as inputs; and, if taken in the right order, successive conversions do correspond to a natural way of implementing the program wich the λ-term represents. Of course Kleene did not think in these terms, but this way of putting it helps one to appreciate the intuitions which led Church and Kleene to believe in the truth of the thesis».

2. La segunda está basada en la equivalencia entre los tres conceptos introducidos para establecer la noción de algoritmo: resulta tranquilizador que desde planteamientos tan distintos se lleguen a definir exactamente las mismas funciones.

3. La tercera, que fue la que sin duda convenció a Gödel, está relacionada con el análisis que Turing hizo del concepto de computabilidad; no se interesa tanto por la naturaleza de las funciones computables como por el propio proceso de computación. Al comienzo de su trabajo Turing afirma:

> «The real question at issue is *What are the possible processes which can be carried out in computing a number?*».

Las funciones se construyen a partir de funciones elementalísimas de las que no cabe duda de su carácter algorítmico mediante procesos que tampoco plantean dudas al respecto.

4. De hecho, el argumento anterior es muy similar al dado por Church, que expone Gandy [24] así:

> «*The Step-by step Argument*. This is Church's chief argument. He considers the evaluation of a value *fm* of a function either by the application of an algoritm, or by the derivation of $fm = n$ from axioms about f in some formal system. In each case the evaluation proceeds in a series of steps. If each step is recursive then f will be recursive. As evidence for the premise he points out that Gödel had shown that the steps in a proof in Gödel system P are primitive recursive».

5. Finalmente, la razón que convenció a Kleene: la del fracaso del argumento diagonal. Llamamos diagonalización al proceso que mimetiza el que Cantor introdujo para demostrar que $\wp\omega$ no es biyectable con ω y que permite señalar conjuntos (en el caso mencionado, $Y=\{x \mid x \notin f(x)\}$) que no han sido incluídos en una enumeración, pretendidamente completa de un determinado dominio.

Ahora se podía demostrar que un conjunto no era decidible pues bastaba con mostrar que no era recursiva la función que cifraba la pertenencia a él. Esto lo hizo Church en 1936 para un problema del cálculo lambda. En el mismo año, apoyándose en los resultados de Gödel, Church demostró que si fuera decidible la lógica de primer orden, también lo sería el sistema del artículo anterior.

4.5. El cálculo lambda y la teoría de tipos. En primer lugar, el cálculo λ de Church permite indagar la naturaleza de las operaciones matemáticas y hemos visto que se consiguen resultados de gran importancia asociados a la definibilidad mediante λ. En segundo lugar, y de eso nos ocupamos ahora, el cálculo lambda sirve también para hacer una magnífica presentación de la teoría de tipos.

Posiblemente para solucionar el problema de inconsistencia del cálculo lambda sin tipos, Church formuló la teoría de tipos que Russell y Withehead habían introducido en los Principia en su potente y peculiar lenguaje lambda y para él definió un cálculo con algunas de las reglas de la conversión lambda. El resultado fue una maravilla, pues a la capacidad formalizadora de las expresiones con lambda se añadía la naturalidad de la representación con tipos. (En mi opinión la solución propuesta en la teoría de tipos para solu-

cionar el problema de las paradojas, la de distinguir niveles o tipos de variables, es muy razonable ya que para representar un paisaje montañoso, lleno de niveles, tal y como nos agrada imaginarnos a la jerarquía de conjuntos, o de tipos, no nos sirve un lenguaje de representación plano, que sea incapaz de dar cuenta de ello; necesitamos un lenguaje que incluya las curvas de nivel.)

Church [9] propone un lenguaje λ con tipos simples en el que las denominadas paradojas lógicas o matemáticas, como la de Russell, o la de Cantor, desaparecen porque la propia sintaxis del lenguaje no permite que se formen las construcciones autorreflexivas causantes de las contradicciones. Las llamadas paradojas semánticas, tales como la del mentiroso, se evitan al adoptar la semántica de Tarski, como resultado de distinguir claramente lenguaje objeto y metalenguaje. Haciendo esto no precisa la complicación adicional de establecer órdenes, tal y como hace Russell en su teoría ramificada. (En el apéndice de esta artículo presento el lenguaje de Church para la teoría de tipos simple[15]).

Pasados muchos años Church publica un artículo [13] en el que se compara la solución de Russell con la de Tarski.

Otra versión del cálculo lambda es la que publica Church [11] en los cincuenta. En ella se toma la tesis de Frege, conforme a la cual para entender un lenguaje no nos basta con conocer la estructura del lenguaje simbólico y el universo de objetos a los que el primero se refiere; hay que interponer una tercera instancia de entes abstractos; a saber, los sentidos o conceptos. Church en el artículo citado desarrolla una teoría matemática del sentido, en la acepción arriba mencionada, y de su relación con los objetos del universo. Para ello utiliza el lenguaje de tipos del artículo de 1940 y le añade una nueva jerarquía de tipos para los sentidos.

4.6. Completud de Henkin. Cuando se introduce un lenguaje formal, con un cálculo deductivo asociado, y una semántica, es una cuestión obligada la de indagar acerca de la adecuación de estos dos procedimientos de selección de fórmulas. Gödel lo había resuelto positivamente para la lógica de primer orden y negativamente para cualquier sistema lógico capaz de contener la aritmética.

El cálculo lambda para la teoría de tipos, con la semántica habitual sobre una jerarquía estándar de tipos era capaz de expresar la aritmética y por consiguiente no podía ser otra cosa que incompleto.

No obstante, Henkin demostró que si se interpretan las fórmulas de una manera menos rígida, aceptando otras jerarquías de tipos que no tengan necesariamente que contener a todas las funciones, sino sólo a las definibles, se prueba fácilmente que toda consecuencia de un conjunto de hipótesis es demostrable en el cálculo. Las fórmulas válidas en esta nueva semántica,

[15] En Manzano [46] se presenta también este lenguaje de Teoría de tipos con λ, así como uno de segundo orden que cuenta también con el abstractor lambda. Parte del libro de Andrews [1] trata de teoría de tipos.

llamada general, se reducen hasta coincidir con las generadas por las reglas del cálculo.

Como todo el mundo sabe, la prueba de completud de Henkin consiste en demostrar que todo conjunto consistente tiene un modelo. Para hacerlo se extiende el conjunto consistente a uno máximamente consistente y ejemplificado y se construye el modelo que las fórmulas de este conjunto máximamente consistente están describiendo; pues estos conjuntos precisamente se caracterizan por ser una descripción pormenorizada de un modelo.

Por supuesto, no entraré aquí en el detalle de este teorema de Henkin, lo único que me interesa señalar es en qué modo su descubrimiento está relacionado con el cálculo lambda.

Según cuenta su creador, estaba tratando de representarse mentalmente las funciones que pueden ser nombradas mediante expresiones con λ. Claramente, si tenemos un universo de individuos numerable, el de conjuntos de individuos será supernumerable, pero puesto que el lenguaje es sólo numerable, los conjuntos que pueden ser nombrados es sólo numerable. El clik definitivo se produjo, según parece, cuando se dió cuenta de que al intentar representarse a los objetos de esta jerarquía de tipos, para eliminar repeticiones, estaba usando las reglas de conversión lambda y que por lo tanto implicaba a la sintaxis y al cálculo. Para identificar a los objetos nombrados mediante M^{α} y N^{α} usaba al cálculo como criterio; en especial que, $\vdash M^{\alpha} = N^{\alpha}$. Finalmente, la prueba se cerró al darse cuenta de que para que el universo de los objetos nombrados mediante proposiciones (el de los valores de verdad) se redujera a sólo dos habría de ampliar los axiomas hasta que constituyeran un conjunto máximamente consistente, de suerte que las clases de equivalencia inducidas por la relación $\vdash M^{\alpha} = N^{\alpha}$ se redujeran a dos.

Extraeré una serie de frases claves del bonito e ilustrativo relato que hace Henkin [31] de su propio descubrimiento.

> «I decided to try to see just which objects of the hierarchy of types did have names in ℑ».
>
> ...
>
> «As I struggled to see the action of functions more clearly in this way, I was struck by the realization that I have used λ-conversion, one of the formal rules of inference in Church's deductive system for the language of the theory ℑ . All my efforts had been directed toward *interpretations* of the formal language, and now my attention was suddenly drawn to the fact that these were related to the *formal deductive system* for that language».
>
> ...
>
> «As soon as I observed this, it occurred to me that if we were to add further cwffs of type 0 to the list of formal axioms, this would have the effect of reducing the number of elementes in $\mathcal{D}_{0}$ and that ultimately, by taking a maximal consistent set of axioms, the number of elements in $\mathcal{D}_{0}$ would be two».

5. Apéndice

5.1. Teoría de Tipos

En lo que sigue presentaré brevemente la teoría simple de tipos del artículo de Church de 1940, complementada en su parte semántica con los de Henkin [27] y [30].

Jerarquía de tipos. La teoría de tipos allí expuesta, que llamaré $\mathfrak{I}$ podría describirse así: Los tipos están estructurados en una jerarquía que cuenta con:

$\mathcal{D}_1$ es un conjunto no vacío; el de individuos de la jerarquía.

$\mathcal{D}_0$ es el dominio de valores de verdad (como estamos en lógica binaria, dichos valores se reducen a *V* y *F*)

El resto de los dominios se construyen a partir de los tipos básicos: si $\mathcal{D}_\alpha$ y $\mathcal{D}_\beta$ han sido construídos ya, definimos $\mathcal{D}_{(\alpha\beta)}$ como el dominio formado por todas las funciones de $\mathcal{D}_\beta$ en $\mathcal{D}_\alpha$. Los conjuntos se representan mediante funciones características y así $\mathcal{D}_{(0\beta)}$ es el conjunto potencia de $\mathcal{D}_\beta$, formado por todos los subconjuntos de $\mathcal{D}_\beta$.

Para hablar acerca de esta jerarquía se introduce un lenguaje formal. El del artículo de Church es, como dijimos, un lenguaje de tipos, que razonablemente posee distintas clases de variables, que corresponden a los diferentes tipos y se cuenta además con el abstractor.

Lenguaje

Alfabeto. Contamos con variables de diferentes tipos, con un superíndice que lo indica.

En este artículo de Church se introducen también las constantes $N_{(00)}$ y $A_{((00)0)}$ (para la negación y la conjunción) y $\Pi_{(0(0\alpha))}$ (nos permitirá expresar cuantificación). Finalmente $\iota_{(\alpha(0\alpha))}$ es un selector.

Expresiones. Hay dos formas fundamentales de formar expresiones:

Las simples: Fórmulas o constantes solas son fórmulas cuyo tipo es el del índice.

Las compuestas son de dos clases: (1) Con λ. Si X^β es una variable y M^α es una expresión, $(\lambda X^\beta M^\alpha)$ es una expresión de tipo $(\alpha\beta)$ (2) También, si $F^{(\alpha\beta)}$ y B^β son expresiones, $F^{(\alpha\beta)}B^\beta$ es una expresión de tipo α.

Interpretación. La interpretación de estas expresiones es como sigue:

Las variables toman valores en los universos o dominios correspondientes. Las constantes tendrán valores decididos de antemano; por ejemplo, $N_{(00)}$ será negación y $A_{((00)0)}$ sera disyunción.

La expresión $(\lambda X^\beta M^\alpha)$ se interpretará como una función de $\mathcal{D}_\beta$ en $\mathcal{D}_\alpha$. (Por ejemplo, $(\lambda X^\beta X^\beta)$ será la función identidad. Sin embargo, $(\lambda X^\beta \phi)$ se interpretará como la clase de los elementos de $\mathcal{D}_\beta$ que verifican la condición expresada por la fórmula ϕ.)

Finalmente, la expresión $F^{(\alpha\beta)}B^{\beta}$ se interpreta como el valor de la función que interprete a $F^{(\alpha\beta)}$ para el valor que interprete a B^{β}.

Por su parte, la función $\Pi_{(0(0\alpha))}$ asigna a cada subconjunto de $\mathcal{D}_{\alpha}$ (que será un elemento de $\mathcal{D}_{(0\alpha)}$) el valor F excepto cuando dicho subconjunto sea el propio $\mathcal{D}_{\alpha}$, la totalidad. Así, podemos pensar que este predicado de predicados afirma de ellos su universalidad. Se entenderá, pues, que $\Pi_{(0(0\alpha))}$ $(\lambda X^{\alpha}\phi)$ exprese cuantificación, $\forall X^{\alpha}\phi$. Literalmente afirma: la clase de los individuos que verifican ϕ es universal, es todo el universo $\mathcal{D}_{\alpha}$ (Hay un lenguaje de la teoría de tipos que me gusta aún más, es el que tiene λ e igualdad como únicos signos primitivos. En él los conectores y los cuantificadores se definen usando sólo estos signos. Por ejemplo, $\forall X^{\alpha}\phi$ se define mediante esta fórmula: $(\lambda X^{\alpha}\phi)=(\lambda X^{\alpha}X^{\alpha}=X^{\alpha})$[16]).

La interpretación de $\iota_{(\alpha(0\alpha))}$ es la de un selector que a cada elemento de $\mathcal{D}_{(0\alpha)}$ (es decir, a cada subconjunto de $\mathcal{D}_{\alpha}$) le asigna un elemento de $\mathcal{D}_{\alpha}$. Cuando la interpretación de la fórmula $(\lambda X^{\beta}\phi)$ es una clase unitaria el selector asigna a dicha clase ese único elemento y así $(\iota_{(\alpha(0\alpha))}(\lambda X^{\alpha}\phi))$ se abrevia como $(\iota X^{\alpha}\phi)$ considerándose como un descriptor.

La diferencia de interpretación entre las constantes $N_{(00)}$, $A_{((00)0)}$ y $\Pi_{(0(0\alpha))}$ por un lado, y $\iota_{(\alpha(0\alpha))}$) por otro, es que en el primer caso ésta está fijada de antemano y de forma unívoca. En el segundo se dice que ha de ser cualquier función de elección del tipo adecuado, sin determinarse cual. Parece ser que esta asimetría, esta indeterminación, está en la base de la investigación que llevó a Henkin a descubrir sus pruebas de completud. De manera que a ella debemos una pieza clave de toda lógica, la reina de las pruebas de los sistemas formales, pues su procedimiento, a diferencia del usado por Gödel para demostrar completud para la lógica de primer orden, es tan versátil que se adapta fácilmente a otras lógicas.

Cálculo. El cálculo deductivo de dicho artículo de Church incluye seis reglas de inferencia y once axiomas. Las tres primeras reglas son reglas de sustitución, incluyendo las reglas de λ-conversión; entre otras, la siguiente

$$\frac{\left(\lambda X^{\beta}M^{\alpha}\right)N^{\beta}}{S^{N\beta}_{X\beta}\left(M^{\alpha}\right)}$$

(una aplicación de esta regla nos permitiria pasar de $(\lambda X^{\alpha}RX^{\alpha})A$ a *RA.)* y su conversa.

Hay también una regla de instanciación, *modus ponens* y generalización.

Además de tener los axiomas proposicionales y los de cuantificadores, se incluye el siguiente axioma para las descripciones

$(f^{(0\alpha)}X^{\alpha}) \rightarrow (\forall Y^{\alpha}(f^{(0\alpha)}Y^{\alpha} \rightarrow X^{\alpha} = Y^{\alpha}) \rightarrow f^{(0\alpha)}(\iota_{(\alpha(0\alpha))}f^{(0\alpha)}))$

Se añade también extensionalidad, elección e infinitud, pues Church quería que sirviera como alternativa a la teoría de conjuntos, como sistema

[16] Ver Henkin [29] y [30], también está en Andrews [1].

fundacional. El axioma de definición de clases y relaciones no es preciso añadirlo ya que se sigue de las reglas de conversión lambda.

$(f^{(0\alpha)}X^{\alpha}) \rightarrow (f^{(0\alpha)}(\iota_{(\alpha(0\alpha))}f^{(0\alpha)}))$ es la elegante formulación del axioma de elección que en este lenguaje propone Church.

$\forall X^{\beta}(f^{(\alpha\beta)}X^{\beta} = g^{(\alpha\beta)}X^{\beta}) \rightarrow f^{(\alpha\beta)} = g^{(\alpha\beta)}$ es extensionalidad.

Henkin [31] sostiene lo siguiente:

> «Several features of the theory $\Im$ sketched above were interesting to me, but I was especially attracted by the neatness and shortness of the formula expressing the axiom of choice. It seemed to me that the symbol $\iota_{\alpha(0\alpha)}$ was put into the formal languaje of $\Im$ originally to serve the function of the definite article *the*, as expressed in Axiom 9^{α} (el de las descripciones), and that its availability to provide such a succint formulation of the axiom of choice was a fortuitous circumstance that must have come to Church as an inspired afterthought».

En este artículo se introducen como abreviaturas las constantes para cada número natural y se define a éstos, siguiendo a Russell y a Frege. Se prueba para ellos, como teoremas, los postulados de Peano.

En particular, el cero, el uno y el dos se definen así:

$0^{\alpha'} = \lambda f^{(\alpha\alpha)}\lambda X^{\alpha}X^{\alpha}$

$1^{\alpha'} = \lambda f^{(\alpha\alpha)}\lambda X^{\alpha}(f^{(\alpha\alpha)}X^{\alpha})$

$2^{\alpha'} = \lambda f^{(\alpha\alpha)}\lambda X^{\alpha}(f^{(\alpha\alpha)}(f^{(\alpha\alpha)}X^{\alpha}))$

La función del siguiente y el concepto de número natural también son definidos mediante expresiones con lambda. Incluye Church en su artículo la definición de la función del predecesor de un número natural, uno de los logros de Kleene [37], que sirvió a ambos para calibrar la enorme potencia de la λ-definibilidad.

5.2. Indecidibilidad de la Lógica de Primer Orden

La indecidibilidad de la lógica de primer orden no se demuestra directamente, se utiliza algún sistema finitamente axiomatizable del que ya sabemos que es indecidible. Church [7] utiliza el de su artículo precedente [8], normalmente ahora solemos usar el sistema **Q** (la aritmética de Robinson), una subteoría de la aritmética que tiene sólo siete axiomas; a saber:

$\mathbf{Q}_1 \equiv \forall xy(\sigma x = \sigma y \rightarrow x = y)$

$\mathbf{Q}_2 \equiv \forall x(c \neq \sigma x)$

$\mathbf{Q}_3 \equiv \forall x(x \neq c \rightarrow \exists y(x = \sigma y)$

$\mathbf{Q}_4 \equiv \forall x(x + c = x)$

$\mathbf{Q}_5 \equiv \forall xy(x + \sigma y = \sigma(x + y))$

$\mathbf{Q}_6 \equiv \forall x(x \times c = c)$

$\mathbf{Q}_7 \equiv \forall xy(x \times \sigma y) = (x \times y) + x)$

La maquinaria introducida por Gödel nos permite decir que un conjunto de fórmulas es decidible, si el conjunto de los números de Gödel de sus elementos es recursivo. Así, una teoría es decidible si y sólo si el conjunto

de los números de Gödel de sus teoremas es recursivo o, de forma equivalente, es recursiva su función característica. Dada una teoría decidible hay un método efectivo para decidir si una sentencia es un teorema: se halla su número de Gödel y se usa ese número como argumento en la función característica. La sentencia será un teorema si y sólo si el valor es 1. Por el contrario, si una teoría no es decidible, y si aceptamos la tesis de Church, no habrá un método efectivo para decidir si una sentencia es un teorema. Si lo hubiera, la función característica de los números de Gödel de sus teoremas sería efectivamente computable y, por lo tanto, recursiva, usando la tesis de Church.

La prueba de que **Q** no es decidible[17] usa los recursos de Gödel (es decir, gödelización y diagonalización) y ciertas propiedades de **Q** (en particular, que todas las funciones recursivas son representables en **Q**).

El argumento de Church se puede presentar como sigue:

(1) Sea $\theta \equiv \wedge_{1\,i\,7} \mathbf{Q}_i$ (La conjunción de los axiomas de **Q**).

(2) Por el teorema de la dedución tenemos que: $\mathbf{Q} \vdash \varphi \Leftrightarrow \vdash \theta \rightarrow \varphi$ (Haciendo uso de los teoremas ya disponibles de completud y corrección, podemos parafrasear este resultado diciendo que un test de validez equivale a un test de pertenecia a la teoría **Q**: para decidir si φ es un teorema, hagamos el test a $\theta \rightarrow \varphi$.)

(3) Designemos mediante $\#\alpha$ al número de Gödel de la fórmula α. Sea f una función que a cada número natural n le asigna $\#(\theta \rightarrow \delta_n)$, donde, $\#\delta_n = n$. Esta función es claramente recursiva. Para cada fórmula φ, de número de Gödel n, $f(n) = \#(\theta \rightarrow \varphi)$.

(4) Sea $\mathcal{P}_0 = \{\#\gamma \mid \vdash \gamma\}$; es decir, el conjunto de los números de Gödel de los teoremas de los teoremas lógicos de la lógica de primer orden.

(5) Sea $\mathcal{P} = \{\#\varphi \mid \mathbf{Q} \vdash \varphi\}$. Por lo tanto, $\mathcal{P} = \{\#\varphi \mid \vdash \theta \rightarrow \varphi\} = \{n \mid f(n) \in \mathcal{P}_0\}$. Pero $\mathcal{P}$ no es recursivo, puesto que es el conjunto de los números de Gödel de los teoremas de **Q**, y sabemos que **Q** no es decidible.

(6) Por consiguiente, $\mathcal{P}_0$ tampoco es recursivo.

(7) La conclusión es que la lógica de primer orden no es decidible; no hay test de validez.

[17] Una demostración de este teorema puede leerse en Boolos [3], en la página 175, basada en resultados demostrados en el libro. Desgraciadamente, este gran lógico también murió recientemente, este mismo año de 1996.

Remark 1. En 1985, en un Congreso del ASL, Robin Gandy[18], quien fuera alumno de Turing y, según se dice, el último que estuvo con él antes de su muerte, dió una conferencia en la que se preguntaba, «*¿por qué todo sucedió en 1936?*». Desde entonces he ido entendiendo mejor sus razones para hacerse semejante pregunta.

Referencias

[1] Andrews, P. B. [1986]. *An Introduction to Mathematical Logic and Type Theory: To Truth through Proof.* Orlando, Fla: Academic Press.
[2] Barendregt, H. P. [1981]. *The Lambda Calculus, its Syntax and Semantics.* Amsterdam: North Holland Publishing Company.
[3] Boolos, G. S. y Jeffrey, R. C. [1992]. *Computability and Logic.* Cambridge University Press. (edición revisada y ampliada, la primera es de 1974).
[4] Church, A. [1924]. «Uniqueness of the Lorentz transformation». *American mathematical monthly.*
[5] — [1926]. «Alternatives to Zermelo's assumption». *Transaction of the American Mathematical Society*, vol. 29, pp. 178-208.
[6] — [1928]. «On the law of excluded middle». *Bulletin of the American Mathematical Society.* vol. 34, pp. 75-78.
[7] — [1936]. «An unsolvable problem of elementary number theory». *American journal of mathematics*, vol. 58
[8] — [1936]. «A note on the Entscheidungsproblem». *The Journal of Symbolic Logic*, vol. 1, Number 1. Marzo de 1936, pp 40-41
[9] — [1940]. «A formulation of the simple theory of types». *The Journal of Symbolic Logic*, vol. 5, pp. 56-68.
[10] — [1941]. «The calculi of lambda conversion». *Annals of Mathematical Studies.* number 6, 1941. Princeton University Press- Oxford University Press.
[11] — [1951]. «A formulation of the logic of sense and denotation». *Structure, Methods and Meaning, Essays in Honor of Henry M. Sheffer*, New York. pp. 3-24. (Revisado en NOUS: vol. 7 (1973), pp. 24-33; vol. 8 (1974), pp. 135-156; and vol. 27 (1993), pp. 141-157.)
[12] — [1956]. *Introduction to Mathematical Logic.* vol I. Princeton: Princeton University Press.
[13] — [1976] «Comparison of Russell's resolution of the semantical antinomies with that of Tarski». *The Journal of Symbolic Logic*, vol. 41, núm. 4.
[14] — [1995] «A theory of the meaning of names». *The heritage of Kazimierz Ajdukiewicz.* Rodopi
[15] Church, A. y Kleene, S. C. [1936]. «Formal definitions in the theory of ordinal numbers». *Fundamenta Mathematica*, vol. 28, pp. 11-21.

[18] Tristemente también desaparecido en 1995. Aquella conferencia se convirtió en un texto magnífico [24].

[16] Church, A. y Quine, W. V. [1952]. «Some theorems on definability and decidability». *The Journal of Symbolic Logic*, vol. 17. núm. 3.
[17] van Dalen, D. [1983]. «Algoritms and Decision Problems: A Crash Course in Recursion Theory». (en Gabbay y Guenthner 1983)
[18] Davis, M. [1965]. *The Undecidable*. Rave Press. Nueva York.
[19] —[1982]. «Why Gödel Didn't Have Church's Thesis». *Information and Control*, 54, pp. 3-24.
[20] — [1988]. «Mathematical Logic and the origins of modern computers». (en Herken [32]).
[21] — [1995]. «American Logic in the 1920s». *Bulletin of Symbolic Logic*, vol. 1, Number 3, Sept. 1995.
[22] Feferman, S. [1988]. «Turing in the Land of O(z)». (en Herken [32]).
[23] Gabbay, D. y Guenthner, F. [1983]. *Handbook of Philosophical Logic*, vol. I. Dordrecht: Reidel Publishing Company.
[24] Gandy, R. [1988] «The Confluence of Ideas in 1936». (en Herken [32]).
[25] Heijenoort, J. ed. [1967]. *From Frege to Gödel*. Harvard: Harvard University Press.
[26] Henkin, L. [1949]. «The completeness of the first order functional calculus». *The Journal of Symbolic Logic*, vol. 14, pp. 159-166.
[27] — [1950]. «Completeness in the theory of types». *The Journal of Symbolic Logic*, vol. 15, pp. 81-91.
[28] — [1953]. «Banishing the rule of functional variables». *The American Mathematical Monthly*, vol. 67, núm. 4.
[29] — [1963]. «A theory of propositional types». *Fundamenta Mathematicae*, vol. 52, pp. 323-344.
[30] — [1975]. «Identity as a logical primitive». *Philosophia*, vol. 5, pp. 31-45.
[31] — [1996]. «The discovery of my completeness proofs, Dedicated to my teacher, Alonzo Church, in his 91st year», *Bulletin of Symbolic Logic*, vol. 2, Number 2, June 1996. (presentado el 24 de Agosto de 1993 en el *XIX International Congress of History of Science*, Zaragoza, Spain).
[32] Herken, R. ed. [1988]. *The Universal Turing Machine: A Half-Century Survey*. Oxford: Oxford University Press.
[33] Hindley, R. y Seldin, J. [1986]. *Introduction to combinators and λ-calculus*. Cambridge: Cambridge University Press.
[34] Hilbert, D. y Ackermann, W. [1928]. *Grundzüge der theoretischen Logik*. Springer-Verlag. Berlín.
[35] Hodges, A. [1983]. *Alan Turing: The Enigma*. Publicado por Burnett Books Ltd (la edición que yo uso es de 1992, de Vintage. Londres).
[36] Horn, B. K P. y Winston, P. H. [1981]. *LISP*. Reading: Addison-Wesley Publishing Company.
[37] Kleene, S. C. [1935]. «A theory of positive integers in formal logic». *American journal of mathematics*, vol. 57, pp. 153-173, 210-244.
[38] — [1936a]. «General recursive functions on natural numbers». *Mathematische Annalen*, vol. 112 (1936), pp. 727-742, (en [18]).

[39] — [1936b]. «λ-definability and recursiveness». *Duke Mathematical Journal*, vol. 2, pp. 340-353.
[40] — [1936c]. «A note on recursive functions». *Bulletin of the American Mathematical Society*, vol. 42, pp. 544-546.
[41] — [1952]. *Introduction to metamathematics*. Amsterdam: North Holland Publishing Company.
[42] — [1967]. *Mathematical Logic*. Wiley.
[43] — [1981]. «Origins of recursive function theory». *Annals of History of Computing*, vol. 3, pp. 52-65.
[44] Langford, C. H. [1928]. «Concerning logical principles». *Bulletin of the American Mathematical Society*, vol. 28, pp. 16-40.
[45] Löwenheim, L. [1915]. «On the possibilities in the calculus of relatives», (en [25]).
[46] Manzano, M. [1996]. *Extensions of First Order Logic*. Cambridge: Cambridge University Press.
[47] Odifreddi, P. [1989]. *Classical Recursion Theory*. Amsterdam: North Holland Publishing Company.
[48] Post, E. L. [1921]. «Introduction to a general theory of elementary propositions», (en [25]).
[49] — [1927]. «Finite combinatory processes». *The Journal of Symbolic Logic*, vol. 2, pp. 103-105.
[50] Shoenfield, J. R. [1967]. *Mathematical Logic*. Reading: Addison-Wesley Publishing Company.
[51] — [1995]. «The mathematical work of S. C. Kleene». *Bulletin of Symbolic Logic*. vol, 1. Núm 1, pp 9-43.
[52] Smullyan, R. [1993]. *Recursion Theory for Metamathematics*. Oxford Logic Guides 22. Oxford: Oxford University Press.
[53] Russell, B. [1908]. «Mathematical logic as based on the theory of types», (en [25]).
[54] Russell, B. y Whitehead, A. [1910-13]. *Principia Mathematica*, vol. 1-3. Cambridge: Cambridge University Press.
[55] Tarski, A., Mostowski, A. y Robinson, R. M. [1953]. *Undecidable Theories*. Amsterdam: North Holland Publishing Company.
[56] Turing, A. [1936]. «On computable numbers with an application to the Entscheidungsproblem». *Proceedings of the London Mathematical Society*, vol. 42, 1936-37, pp. 230-265; también vol. 43, 1937, pp. 544-546.
[57] — [1937]. «Computability and λ-definability». *The Journal of Symbolic Logic*, vol. 2, pp. 153-163.