18h21

Responda a las 3 preguntas siguientes en el espacio máximo permitido en cada caja. Puede usar cualquier material, pero no puede comunicarse con terceras personas.

Se ha mostrado así que no es posible obtener una definición de calculabilidad efectiva más general que la arriba propuesta por ninguna de las dos vías que se ofrecen naturalmente, a saber, (1) definir una función como efectivamente calculable si hay un algoritmo para calcular sus valores y (2) definir una función *F* (de un entero positivo) como efectivamente calculable si, para cada entero positivo m hay un entero positivo *n* tal que *F*(*m*) = *n* es un teorema demostrable.

(Church 1936, "An unsolvable problem of elementary number theory", p. 358, citado en Torretti, *El paraíso de Cantor*, p. 375)

1. Explique por qué la definición de función efectivamente calculable (2) obedecería el programa de Hilbert (3 puntos)
2. ¿Qué diferencias hay entre la noción de calculabilidad efectiva de Church y la noción de computabilidad de Turing? ¿Por qué son extensionalmente equivalentes? (3 puntos)
3. Explique por qué no se puede demostrar matemáticamente la tesis de Church (4 puntos)

Comentario profesor:

Tus dos primeras respuestas me parecen perfectas, y la tercera muy interesante, aunque no acaba donde yo esperaba. Tienes un sobresaliente

1.

La primera dificultad que se encontró Church cuando afrontó la tarea de determinar si una función es calculable fue determinar qué es la calculabilidad. Diferenció dos conceptos: (i) la calculabilidad tal como la entendemos aplicando nuestro sentido común, (ii) la calculabilidad efectiva, que definió a partir de un sistema que denominó lambda-cálculo basado en unos operadores, fórmulas y reglas (y una lista de teoremas que dedució de éstos). Una función era efectivamente calculable si se correspondía con una lambda-función generada por el sistema anterior. Ambos conceptos no se pueden demostrar como equivalentes – debido a la falta de formalización del primero -, pero Church asume que sí lo son. Así pues, el estudio de la calculabilidad efectiva es relevante por lo análogo a la idea de calculabilidad que los humanos entendemos.

Es en este contexto que enmarcamos la cita del enunciado. En ésta, el objetivo de Church al definir la función F era construir un criterio que enumerara todos los teoremas del sistema. Podemos entender la función F como el criterio con el que asignamos un gödel a cada uno de estos teoremas. El hecho de que exista tal función F significa que el conjunto de teoremas es recursivamente enumerable, ya que el valor que asigna F para un numeral puede obtenerse en relación con los gödels asignados para numerales inferiores.

Es decir, existe un criterio recursivo para construir todos los teoremas del sistema, la cual cosa hace innecesario recurrir a recursos transfinitos para definir la lógica en coherencia con los requisitos del programa de Hilbert.

Comentario profesor:

Hay varias maneras de responder a esta pregunta. Una bien sencilla es esta. La definición (2) nos dice que una función *F* (de un entero positivo) es efectivamente calculable si, para cada entero positivo m hay un entero positivo *n* tal que *F*(*m*) = *n* es un teorema demostrable. Tal como explica Torretti en la página anterior (375(, Church construye esta definición para un sistema de lógica que cumple con los desiderata del programa de Hilbert (construir un sistema formal basado en la lógica de primer orden lo bastante potente como para dar cuenta de la matemática ordinaria). Las reglas para efectuar operaciones han de ser efectivamente calculables (es decir, es decir hay un algoritmo para ejecutarlas sin preocuparse por el significado de los símbolos); el conjunto de las reglas si es infinito, ha de ser enumerable, de modo que no dé lugar a paradojas como las de los transfinitos cantorianos; y se puede establecer la correspondencia entre los teoremas demostrables en el sistema y las funciones que las representan, de modo que las verdades lógicas pueden ser demostradas con los recursos del sistema formal.

Una forma alternativa de responder, que algunos habéis usado con éxito es explicar la definición de función efectivamente calculable en el cálculo lambda y mostrar cómo este cálculo cumple con los desiderata de Hilbert sobre razonamiento formal.

2.

Posteriormente al trabajo de Church, Turing inventa el concepto que denominamos máquina de Turing: Ésta construye una mecánica para – a partir de una lista de estados y alfabeto finitos, y una cinta con infinitas casillas – evaluar unos datos de entrada mediante un procedimiento iterativo que va transformándolos en unos datos de salida. Para Turing una función es computable si hay una máquina de Turing tal que para toda palabra de entrada genera una salida correspondiente al valor de la función.

Por otro lado, en línea a lo comentado en el apartado anterior, Church denomina que una función es efectivamente calculable si es lambda-definible. Es decir, si es posible generarla a partir del lambda-cálculo anterior.

Aunque en apariencia ambas definiciones se basan en conceptos muy distintos, el hecho de que las funciones calculables de Church sean recursivamente enumerables permitiría transformar la lambda-definibilidad en un proceso iterativo, que podría asimilarse con el procedimiento de la máquina de Turing. Es por esto por lo que ambos conceptos son extensionalmente equivalentes.

La diferencia entre ambos conceptos es que la máquina de Turing aporta un mecanismo para construir esta función, mientras que las lambda-funciones no pretenden solucionar cómo generar esta sucesión de acciones que la resuelvan.

Comentario profesor:

La diferencia entre el concepto de calculabilidad de Church y el de computabilidad de Turing radica en que el concepto de calculabilidad pretende captar las operaciones de un calculista sin dar una definición completamente formal de las mismas, mientras que Turing las formaliza a partir del funcionamiento de su máquina. Esto, como vamos a ver, es importante para la respuesta siguiente. Una forma más técnica de verlo es a partir de la tesis de Church, que sostiene que una función definida sobre los números naturales es efectivamente calculable si y sólo si es recursiva. Pero es Turing quien muestra que toda función computable debe de ser recursiva. De ahí que se pueda probar la equivalencia extensional: toda función efectivamente calculable es computable etc, pero los matices de la definición de calculabilidad y computabilidad no son los mismos y por esos os pregunto.

3.

El problema de detención consiste en determinar si es posible construir un algoritmo que verifique si una máquina de Turing con una palabra de entrada determinada finaliza o en un tiempo finito. Turing analizó este problema y demostró que no tiene solución. Es decir: es indecidible si dados una MT y unos datos de entrada arbitrarios, estos obtienen una solución finita.

En línea a la equivalencia que menciono en el apartado anterior entre los dos conceptos de calculabilidad, Turing demostró que la demostración de la tesis de Church es análoga al problema de detención. Es decir: en el caso de que fuera posible decidir si un teorema del sistema de Church es verdadero o no para unos datos determinados, podríamos crear una nueva función que relacionara la función f y su asignación a verdadero/falso, y esta función resolvería el problema de detención.

Así pues, no es posible tener un criterio universal en este sistema que determine si un determinado teorema es cierto para un determinado dato.

Comentario profesor:

Torretti da indicaciones bastante claras (pp.375-76), a las que aludía en mis comentarios de las PEC. ¿Se pueden reducir los conceptos del lenguaje natural (calculabilidad etc) de los que parte Church a una definición formal susceptible de análisis matemático? Quizá se vea más claro si pensamos en la definición de computabilidad de Turing: una función computable era aquella que pudiese procesar una Máquina de Turing. ¿Pero cómo sabemos si eso agota el concepto de

computable? ¿Hacemos una encuesta para saber si a la gente le parecen "computables" todas las funciones que pueda computar una máquina de Turing? (Es decir, una prueba de la equivalencia entre el concepto informal de computabilidad y el concepto formal de Turing)