Modelos de Computación (2015-2016)
Grado en Ingeniería Informática
Universidad de Granada

Cuestionario 0

José Carlos Martínez Velázquez

12 de octubre de 2015

Índice

1. El entscheidungsproblem.

Entscheidungsproblem, problema de decisión en castellano, es un problema que se remonta al siglo XVII, cuando Gottfried Leibniz después de haber conseguido crear una máquina automática de cálculo se preguntaba si sería posible crear una máquina tal que dada una frase en matemáticas pudiera determinar si ésta era un teorema o no. Concretamente la frase debía ser un conjunto de símbolos que compusieran una fórmula bien formada de la lógica de primer grado (o lógica de predicados), es decir, una hipótesis. El trabajo de la máquina consistiría en decidir si la hipótesis afirmaba una verdad no evidente por si misma (tesis). En definitiva se trataba de crear una máquina capaz de realizar demostraciones matemáticas, un asunto de vital importancia en la lógica matemática. En 1928, David Hilbert y Wilhelm Ackermann propusieron la pregunta en su formulación anteriormente mencionada, llamándola formalmente Entscheidungsproblem.

2. Church, Turing y el entscheidungsproblem.

Alonzo Church y Alan Turing intentaron resolver éste problema, pero no de la misma forma. En primer lugar, Church (1936) tuvo que definir la noción de algoritmo, con el concepto de calculabilidad efectiva, basado en su "lambda calculus". Posteriormente Turing haría lo propio basándose en la máquina de Turing.

Church decía textualmente:

> ***"Una función es efectivamente calculable si y sólo si es recursiva."***

Es fácil entender que aunque no es lo mismo ser efectivamente calculable que recursivo, una cosa implica forzosamente la otra según su tesis, más adelante comenta:

> ***"Los conjuntos de funciones calculables y funciones recursivas son necesariamente coextensivos"***

A partir de aquí, podemos deducir varias cosas. Podemos basarnos en conocimientos previos para introducir el concepto de algoritmo. Un procedimiento es algorítmico cuando se puede resolver computacionalmente de forma eficiente, es decir, cuando se puede "programar" y termina en un tiempo polinómico. Una función recursiva, si está correctamente definida, es algorítmica, por tanto podríamos ampliar las frases de church a:

> ***"Una función es efectivamente calculable si y sólo si es algorítmica."***

> ***"Los conjuntos de funciones calculables y funciones algorítmicas son necesariamente coextensivos"***

Quizás sea una afirmación más restrictiva, pero capta la esencia de lo que quería demostrar Church: si, por algún motivo, una sola función no fuese capaz de terminar en

tiempo polinómico, no sería algorítmica, entonces no sería efectivamente calculable y, por tanto, no existiría la máquina con la que soñaba Leibniz.

Entendida así, la tesis de Church adquiere un valor de verdad desconocido, puesto que no existe una prueba de verdad ni se ha encontrado ningún contraejemplo capaz de contradecirla.

Tomando como verdad la tesis de Church, sólo había que encontrar un problema cuya solución jamás pudiera ser algorítmica. Aquí es cuando entra en juego Turing.

Turing, muy poco tiempo después (1937), acercó más la tesis de Church al concepto de computabilidad. Turing mencionaba en su tesis:

"Una función es computable si y sólo si es Turing-computable"

Hay que aclarar que con Turing-computable se refería a que es posible construir una máquina de Turing tal que dado el argumento de la función como dato de entrada a la máquina, ésta produce el valor de la función como dato de salida.

Para demostrar que existen problemas no resolubles mediante computabilidad, Turing propuso el problema de la parada. Consiste en demostrar que no existe una manera automática computable de saber si todos los programas del mundo terminan. Cabe destacar que no se niega que exista la prueba para programas concretos, es decir, que hay programas para los que funcionaría esa función. Para demostrar ésto, supongamos que éste problema se puede resolver algorítmicamente, por tanto existe el programa siguiente:

```
def Termina(p,x):
        ...#Codigo del programa
```

El programa Termina recibe como entrada un programa (p) y una entrada a ese programa (x). Si el programa p termina en una serie de pasos finitos devolverá true, sin embargo, si p se queda "trabado", devolverá false.

Ahora imaginemos que tenemos una máquina de Turing que recibe como entrada un programa y usará el programa Termina() para decidir si un programa termina o no. El código sería algo así:

```
def Turing(p):
        if(Termina(p,p)):
                while True: pass #Bucle infinito
```

De manera que la si el programa p al que se le pasa como entrada él mismo, p(p), termina, Turing(p) se queda en un bucle infinito (no termina), y al contrario, si p(p) se queda en un bucle infinito, Turing(p) termina. Suponiendo que existe todo lo anterior, si ejecutáramos el siguiente código:

```
def Turing(Turing):
        if(Termina(Turing,Turing)):
                while True: pass #Bucle infinito
```

Obtendríamos lo siguiente:

$$Turing(Turing) = \begin{cases} termina \rightarrow Turing(Turing)\ no\ termina \\ \\ no\ termina \rightarrow Turing(Turing)\ termina \end{cases}$$

Lo que es una contradicción, reduciéndo el problema a lo absurdo. Con ésta demostración Turing llega a la conclusión de que existe al menos un problema no Turing-computable y por tanto no computable, por lo que el conjunto de todos los problemas del mundo no son computables.

Dado que las tesis de Church y Turing fueron prácticamente coetáneas no se habla de ellas por separado, sino que se habla de la tesis Church-Turing, pero no está mal saber que fueron estudios por separado.

3. Vida y contribución de Turing a la computabilidad.

Alan Mathison Turing (Londres, 1912- Wilmslow, Reino Unido, 1954) es considerado una de las piezas clave en el mundo de la computación, además de contribuir decisivamente en campos como la informática teórica y la criptografía. Entre sus más destacables hitos científicos encontramos: la función calculable, la máquina de Turing, el pre-desarrollo de la computadora Colossus, la desencriptadora Bombe, la prueba sobre inteligencia artificial, además de un largo etcétera de aportaciones conceptuales y técnicas para el desarrollo de la ciencia.

En 1934, Turing se graduó en la Licenciatura de Matemáticas en la Universidad de Cambridge y, en 1936 publicó el artículo "Los números computables, con una aplicación al Entscheidungsproblem" en el que ya hablaba del concepto de algoritmo y exponía las bases de su máquina de calcular: la Máquina Universal (de Turing).

Después de su estancia entre los años 1937 y 1938 en la Universidad de Princeton en Nueva Jersey, obtuvo el Doctorado y anunció el concepto de hipercomputación, que tomaba como base la Máquina Universal y preludiaba una nueva "máquina oráculo" que permitiera el estudio de problemas cuya solución algorítmica no existiera.

El ejército inglés precisó de la labor de Turing para poder combatir contra el bando alemán a partir de descifrar los códigos que su Marina emitía con la máquina Enigma y los codificadores de teletipos FISH. El resultado del trabajo capitaneado por Turing fue la máquina descifradora Bombe y varias computadoras electrónicas Colossus, consideradas, para algunos, los primeros ordenadores de la historia y, por lo tanto, el inicio de la informática tal y como la conocemos.

Desde 1952, Turing se centró en otra materia: la biología matemática. Su trabajo fue recogido en el libro "Fundamentos Químicos de la Morfogénesis" y estaba enfocado en analizar la existencia de los números de Fibonacci -sucesión de cifras que está presente en la naturaleza de forma estable- en las estructuras vegetales.

La vasta carrera de Turing se vio deteriorada por cuestiones personales. La "condición" de homosexual del matemático le llevó a ser condenado ya que en ese momento, en Inglaterra, se concebía como un delito. Ante la opción de ir a la cárcel o someterse a una castración química, Turing optó por la segunda, que le provocó trastornos físicos y en consecuencia, psicológicos. En 1954, con tan solo 42 años, Alan Turing murió envenenado con una manzana recubierta de cianuro, lo que algunos relacionan con el origen del simbolo de la empresa Apple.

(Extracto de [2])

4. Tesis de Church-Turing. Importancia.

La importancia de la tesis de Church-Turing radica en el estudio y resolución del Entscheidungsproblem planteado por Hilbert y Ackerman en 1928. La tesis ha sido tan exitosa que generalmente se da por verdadera. Los términos que se deducen de ámbas tésis son ampliamente utilizados en lo que a computabilidad y Turing-computabilidad se refiere, en el caso de la tesis de Turing. En la diferencia de ambos conceptos se encuentra la tesis de Church. Todo lo que se deduce de ésta tesis se utiliza en la teoría de computabilidad y de funciones recursivas.

5. Tesis de Church-Turing. Variantes y detractores.

Tal ha sido la importancia de la tesis de Church-Turing que, aplicada a la física, se ha llegado a afirmar que el universo es una máquina de Turing y que no es posible construir físicamente una máquina con mayor poder computacional o que sea capaz de computar funciones que no sean recursivas. Ésta quizas sea la variación más extendida de la tesis, denominada Tesis de Church-Turing fuerte, pero existen otras como:

- Tesis Física de Church-Turing (PCTT): "cualquier función que pueda ser físicamente computable, puede ser computada por una máquina de Turing." (W. Fouche, 2002)
- Tesis de Church-Turing Extendida (ECT): "si alguna función es computable por algún dispositivo hardware para una entrada de tamaño n, entonces dicha función es computable por una máquina de Turing en (T(n))k para algún k fijo (dependiente del problema)."
- La Tesis de Zuse-Fredkin: "El universo es un autómata celular"(Zuse y Fredkin, 1960)
- La Tesis M: "Todo aquello que pueda ser calculado por una máquina (en caso de que se trabaje con datos finitos de acuerdo a un número finito de instrucciones) es computable por una máquina de Turing."

Los detractores de la tesis se basan en que aunque se da por verdadera, no se sabe si es lo es ya que aunque no se ha encontrado un contraejemplo que contradiga la tesis de Church-Turing tiene un valor de verdad desconocido. Es más fácil demostrar la falsedad de la tesis que su veracidad, bastaría con encontrar un algoritmo que no sea Turing-computable. A pesar de que parece imposible, negar que sea verdadera es una posibilidad lógica.

Referencias

[1] Jaime Ramos Arenas. Sobre la naturaleza de la tesis de church. http://www.bdigital.unal.edu.co/30401/1/29213-104943-1-PB.pdf.

[2] UPF. Alan turing. http://www.upf.edu/pdi/dcom/xavierberenguer/recursos/fig_calc/_9_/estampes/2_3.htm.

[3] WikiBooks. La tesis de church-turing interpretaciones. https://es.wikibooks.org/wiki/La_tesis_de_Church-Turing/Interpretaciones.

[4] Wikipedia. Entscheidungsproblem. https://es.wikipedia.org/wiki/Entscheidungsproblem.

[5] Wikipedia. Problema de la parada. https://es.wikipedia.org/wiki/Problema_de_la_parada.

[6] Wikipedia. Tesis de church-turing. https://es.wikipedia.org/wiki/Tesis_de_Church-Turing.