Teorema de Gödel-Turing

(Adaptado de la presentación de Roger Penrose en *Shadows of the Mind*)

En todo computador digital, los programas se llevan a una representación binaria que pueda ser ejecutada por el procesador. Esta es la versión en lenguaje de máquina del programa. Esta representación se puede ver como un largo número binario q. Obviamente existen muchos programas y todos tienen un número binario distinto, llamemos estos programas[1]:

$$C_0, C_1, ..., C_q, ...$$

Los programas aceptan argumentos como entradas. Los argumentos proporcionan la información a ser procesada por el programa. Sin importar el número y tipo de argumentos, todos los argumentos se llevan finalmente a la máquina en una representación binaria y el conjunto de todos los argumentos de entrada corresponde con un número binario n, posiblemente muy grande. En general, utilizamos la notación

$$C_q(n)$$

para referirnos a la ejecución del programa q con entradas n.

Una propiedad de interes es saber si el programa termina o no con una determinada entrada. Un programa puede no terminar en casos triviales porque posea un loop sin fin, o en situaciones más generales porque no se conozca una demostración de que el algoritmo termine[2]. Indiquemos por T el predicado "el programa termina", de modo que la proposición "el programa $C_q(n)$ termina" se escribe:

$$T(\ C_q(n)\ )$$

Supongamos que existe un programa A(q,n) que acepta 2 entradas y que termina si el programa $C_q(n)$ no termina, es decir, asumimos para efecto de contradicción la premisa:

1. $\forall q\ (\ T(A(q,n)) \Rightarrow \neg T(C_q(n))\ )$ Premisa

Al aceptar como válida esta premisa para todo posible programa q, entonces debe cumplirse para el programa n:

2. $T(A(n,n)) \Rightarrow \neg T(C_n(n))$ 1, Particularización universal

1 Es claro también que no todos los números binarios corresponden a un programa funcional, pero esto no es relevante para la presentación que sigue. En particular, se podría diseñar un procesador en el que todo op-code no estándar fuese interpretado como una instrucción *no-operation* (NOP).
2 Un ejemplo: Encontrar un número par mayor que 2 que no sea la suma de 2 primos. Hasta el momento no se sabe si ese número existe. Para más detalles ver La Conjetura de Goldbach.

Siendo A(n,n) un programa, entonces debe corresponderse con uno de los programas de la colección de todos los posibles programas $\{C_0, C_1, ..., C_q, ...\}$. En concreto, sea k el programa que corresponde:

3. $A(n,n) = C_k(n)$ A es un miembro del conjunto de los programas.

Un caso particular de (2), es cuando la entrada del programa es k, esto da:

4. $T(A(k,k)) \Rightarrow \neg T(C_k(k))$ 2, $S^n{}_k$

y de la equivalencia en (3), entonces (4) puede reescribirse como:

5. $T(C_k(k)) \Rightarrow \neg T(C_k(k))$

Al estar utilizando un sistema de derivaciones lógicas consistente[3], la implicación en (5) es verdadera. Observando que se trata de una proposición de la forma $P \Rightarrow \neg P$, debe ocurrir que $T(C_k(k))$ sea falsa:

6. $T(C_k(k)) = F$ 5, Tabla de verdad de $\Rightarrow$

7. $T(A(k,k)) = F$ Sustitución de 3 en 6

En otras palabras se ha encontrado un programa concreto $C_k(k)$ que no termina, para el cual el programa A(k,k) no termina, contradiciendo la suposición inicial: que A termina en casos en los $C_q(n)$ no termina. Esto lleva a concluir que no existe un programa general A que demuestre que este programa no termina.

Este resultado se puede generalizar aún más (Gödel) para concluir que el sistema de derivaciones de la lógica de primer orden es **incompleto**. No existe un programa (procedimiento mecánico, algoritmo) que demuestre la validez de toda expresión lógica.

3 No es posible derivar una falsedad si se aplican correctamente las reglas de inferencia del sistema.