

# INF155: Informática Teórica

## Ayudantía 6: Computabilidad

Aldo Berrios Valenzuela

Miércoles 25 de Mayo de 2016

### 1. Ayudantía

---

**Definición 1.1** (Decibilidad). Hay una “máquina” que siempre responde “sí” o “no”.

*/\* Las demostraciones son bastante informales \*/*

**Ejemplo 1.1.** ¿Imprime un programa  $P$  “Hola mundo”.

**Demostración.** Supongamos que existe un programa  $H$  que detecta cuando un programa  $P$  (valga la redundancia) con entrada de datos  $D$  imprime “Hola mundo”, arrojando un “sí” cuando  $P$  imprime “Hola mundo” y “no” en caso contrario.

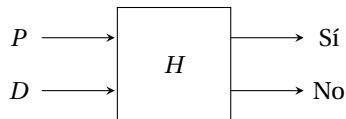


Figura 1: Si  $P$  con entrada de datos  $D$  en algún momento imprime “Hola mundo”, entonces  $H$  imprime “Sí”, y “No” en caso contrario.

Modificamos  $H$  (llamado  $H'$ ), de tal forma que ahora sólo necesite un programa  $P$  de entrada (los datos  $D$  se guardan en memoria).

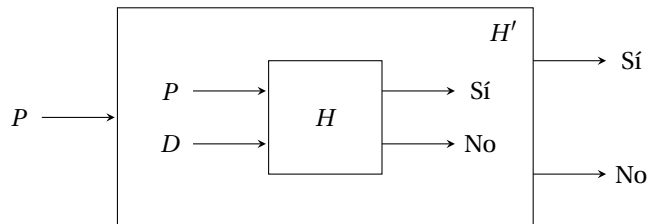


Figura 2: Los datos de  $P$  ahora están almacenados en memoria de  $H'$ .

Modificamos  $H'$  tal que si  $P$  imprime “Hola mundo”, entonces imprime “Sí”, y “Hola mundo en caso contrario”. El resultado es  $H''$  (Figura 3).

Finalmente, si simulamos  $H''$  sobre si mismo, vemos que si imprime “Hola mundo”, entonces  $H''$  muestra “sí”. Pero  $H''$  imprimió “Hola mundo”, lo que es una clara contradicción. En consecuencia  $H$  no existe.  $\square$

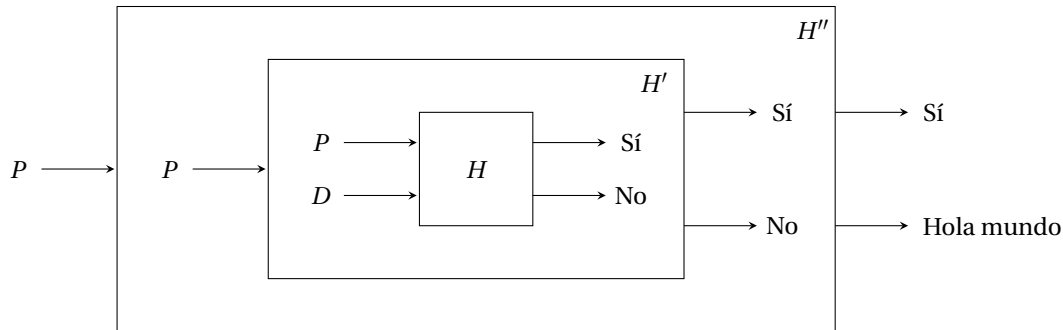


Figura 3: Cuando  $H''$  detecta que “Programa” escribe “Hola, mundo”, en lugar de arrojar como salida un “No”, saldrá un “Hola, mundo”.

**Definición 1.2** (Reducción de Problemas). Un problema  $P$  se reduce al problema  $Q$  si hay un algoritmo (procedimiento finito) que dado  $\sigma \in \Sigma$  entrega  $\sigma''$  tal que:

$$\sigma \in P \Leftrightarrow \sigma' \in Q$$

Se anota  $P \leq Q$  ( $P$  es más fácil que  $Q$ ).

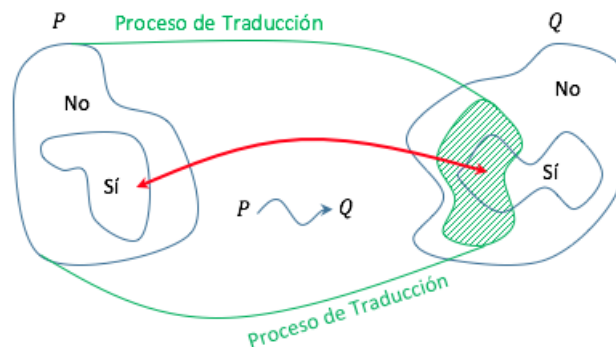


Figura 4: El proceso de transformación traduce el problema  $P$  a un problema  $Q$  que es más difícil.

Las soluciones en  $P_2 \Rightarrow P_1$  (si tenemos solución en  $P_2$ , entonces tenemos solución para  $P_1$ ). Por otro lado, el contrapositivo de esto es  $\overline{P_1} \Rightarrow \overline{P_2}$  (si no tenemos solución para el problema  $P_1$ , entonces tampoco hay para  $P_2$ ). Por lo general, nosotros usaremos el segundo caso ( $\overline{P_1} \Rightarrow \overline{P_2}$ ) para nuestras demostraciones.

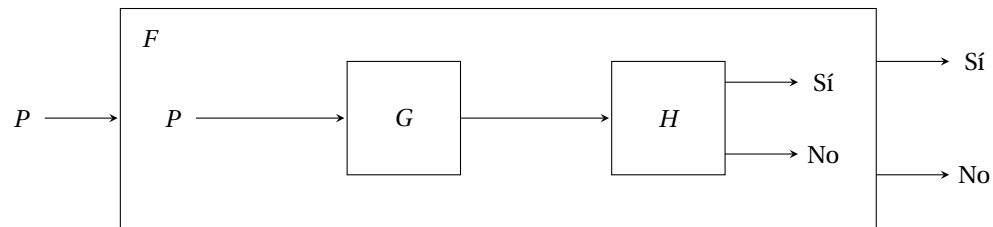
**Ejemplo 1.2.** Demuestre que  $Q$ , que detecta cuando se llama a `foo()` es indecible.

**Demostración.** Supongamos que existe un programa decidable  $H$  que detecta cuando se llama a `foo()`. Sea  $P$  un programa que en algún momento imprime “Hola mundo”,  $G$  un programa que transforma cada `printf(“Hola mundo”)` en `foo()`.

Hacemos lo siguiente:

- Creamos `foo()`
- Reemplazamos `printf` por una versión a memoria (en lugar de imprimir, guardamos el resultado en el buffer)
- Si en el buffer está “Hola mundo”, llamar a `foo()`.

En el fondo lo que hacemos es:



Lo que hace  $F$  es detectar si se imprime "Hola mundo". Se supone que  $H$  es un detector de `foo()`, pero la máquina  $F$  no existe, por ende,  $H$  no puede existir.  $\square$

**Ejemplo 1.3.** /\* Este corresponde al ejercicio final que tengo escrito en papel \*/