

INF155: Informática Teórica

Clase 18: Computabilidad

Aldo Berrios Valenzuela

Martes 10 de mayo de 2016

1. Computabilidad

¿Qué es computación? ¿Que puede (o no) hacer?. Un clásico:

```
#include <stdio.h>
main()
{
    printf("Hola, mundo!\n");
}
```

Pregunta: ¿Podemos escribir un programa que detecte “Hola, mundo!”? (dado un programa, determina si lo primero que escribe es “Hola, mundo”)

Demostración. Consideremos los programas en “C”, con verdaderos enteros como `int`; sin reales. Buscamos construir un programa H (programa que detecta si **otro** programa escribe “Hola, mundo”) (Figura 1). Los programas

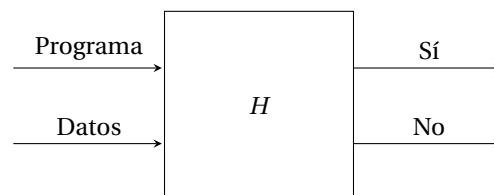


Figura 1: Si el programa de entrada “Programa” escribe en algún momento “Hola, mundo”, la salida de H será “Sí”. En caso contrario será “No”. Nótese que estamos suponiendo que H está escrito en C y usa su biblioteca estándar.

leen de entrada estándar únicamente y escriben a salida estándar exclusivamente.

Primer cambio: Modificamos H de manera que lea sólo el programa, y use el texto del programa como datos. Lo llamaremos H' (Figura 2).

Segunda Modificación: Modificamos H' de manera que en vez de escribir “No”, escribe “Hola mundo”. El resultado es H'' (Figura 3).

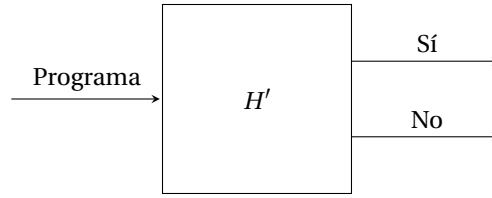


Figura 2: El texto que “Programa” utiliza como datos se guarda previamente en memoria antes de su ejecución.

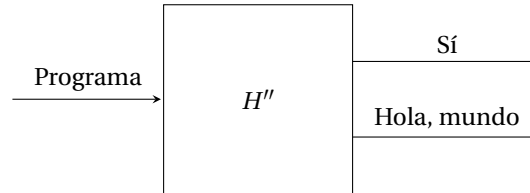


Figura 3: Cuando H'' detecta que “Programa” escribe “Hola, mundo”, en lugar de arrojar como salida un “No”, saldrá un “Hola, mundo”.

¿Qué pasa si alimentamos H'' con H'' ? **Hint:** Suponga que H'' es un archivo .c gigantesco (no modularizado)

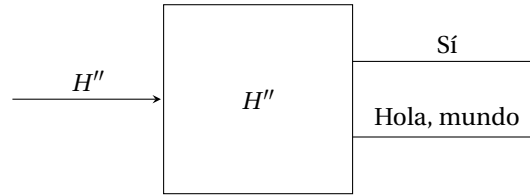


Figura 4: Independientemente de lo que lea H'' siempre habrá una contradicción.

Ocurre lo siguiente:

- Si H'' leyendo H'' responde “Sí”, debiera escribir “Hola, mundo”. Es una contradicción.
- Si H'' leyendo H'' responde “Hola mundo”, debiera escribir “Sí”. Es una contradicción.

Por lo tanto, H no existe. □

Observación: Nótese que cada vez que construimos una modificación del programa H o H' , el siguiente programa sólo es subconjunto de lo anterior (Figura 5)

Definición 1.1. Un problema P es un lenguaje infinito sobre Σ . Resolver el problema es construir un artilugio S que reconosca P . O sea, dado $\sigma \in \Sigma^*$,

$$S(\sigma) = \begin{cases} \text{Verdadero,} & \sigma \in P \\ \text{Falso,} & \sigma \notin P \end{cases}$$

Observación: Si limitamos σ finito, entonces sólo basta con hacer una tabla que abarque todos los casos posibles de σ cuando el problema es Verdadero, o es Falso (Tabla de resultados pre-calculados).

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

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

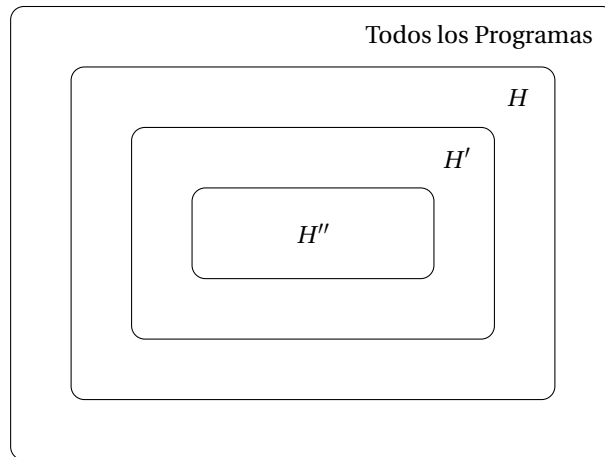


Figura 5: Si H' no funciona, entonces H' y H tampoco. Esto funciona de la misma manera que los lenguajes regulares y las gramáticas de contexto libre (Si no es regular, entonces no es de contexto libre)

Se anota $P \leq Q$ (P es más fácil que Q). Dicho de otra forma, lo que hacemos es transformar el problema a un ámbito distinto que resuelve el mismo problema. Por ejemplo, recuerde en MAT023 cuando resolvíamos EDO's con transformada de Laplace.

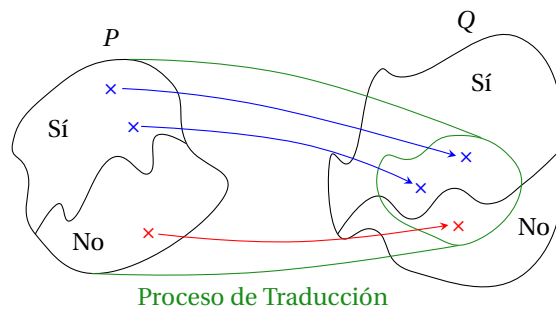


Figura 6: El proceso de transformación traduce el problema P a un problema Q que es más difícil.