INF155 - Informática Teórica

Ayudantía #7

NP-Complete Warriors

Semestre 2022-2

UNIVERSIDAD TECNICA FEDERICO SANTA MARIA
DEPARTAMENTO DE INFORMÁTICA

Computabilidad

Una de las primeras cosas que nos dicen en nuestros primeros años de U es: *si lo puedes imaginar, lo puedes programar*.

Computabilidad

Una de las primeras cosas que nos dicen en nuestros primeros años de U es: *si lo puedes imaginar, lo puedes programar*.

¿Pos qué creen? era mentira. Hay problemas que un computador NO puede resolver. Esto es el concepto de *indecibilidad* (o no decidibilidad). Ocurren cuando nos enfrentamos a problemas que producen un tiempo infinito en producir una respuesta.

A continuación veremos el ejemplo clásico y la base que nos permitirá entender todo lo que se viene: **el detector de *hola mundo***

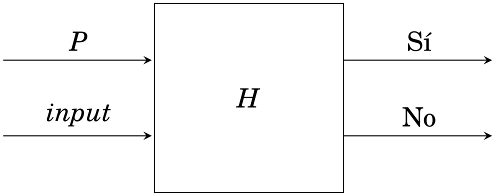
Detector de *Hola Mundo*

Problema: dado un programa, ¿es posible determinar si en algún momento imprime *Hola Mundo*?

Consideremos programas escritos en Python (da lo mismo el lenguaje de programación). Lo que se busca construir es un programa H, el cual es capaz de detectar si otro programa imprime "Hola Mundo". Si el programa de entrada P escribe en algún momento "Hola Mundo", la salida de H será "Sí"; en caso contrario, será "No".

Detector de *Hola Mundo*

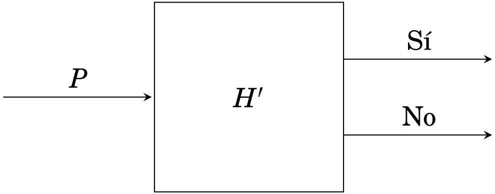
El programa H puede representarse con el *modelo de las cajitas*:



Ahora, comenzaremos a realizar modificaciones sobre este programita...

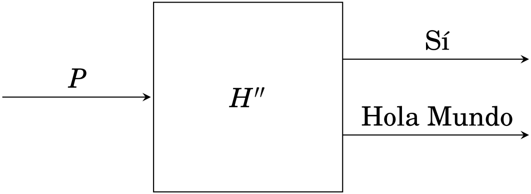
Detector de *Hola Mundo*

Primer cambio: modificaremos H de tal forma que solo lea el programa P y use el texto de dicho programa como *input*. A este programa lo llamaremos H' y lo representamos de la siguiente forma:



Detector de *Hola Mundo*

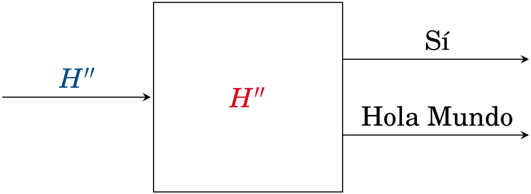
Segundo cambio: modificamos el programa H' de tal forma que en vez de decir "No", dice "Hola Mundo". A este programa lo llamaremos H'' y lo representamos de la siguiente forma:



Detector de *Hola Mundo*

¿Qué pasa si entregamos H″ como input de H″?

Detector de *Hola Mundo*



- Si H'' imprime "Hola Mundo", H'' imprime "Sí". $\Rightarrow\Leftarrow$
- Si H'' imprime "Sí", H'' imprime "Hola Mundo". $\Rightarrow\Leftarrow$

Detector de *Hola Mundo*

Dado que obtenemos una contradicción, no se puede construir H''. Como H'' resulta de modificar un programa hipotético H, entonces H tampoco puede existir. Por lo tanto, **es imposible construir un programa que determine si otro programa, en algún momento, imprime *Hola Mundo***.

Reducción de problemas

Tenemos un problema que sabemos que ningún programa informático puede resolver. ¿Cómo demostramos si algún otro problema puede ser solucionado o no por una computadora?

1. Desarrollar un programa paradójico que hace dos cosas contradictorias (como con "Hola Mundo")
2. Demostrar que si podemos resolver el nuevo problema, entonces usamos aquella solución para un problema que sabemos que es indecidible.

Ejemplo

Demostraremos que es imposible determinar que un programa, al leer datos, ejecuta la función `f`.

Demostración: reduciremos el problema "¡Hola, mundo!" a nuestro problema. Dado un programa P que puede o no escribir "¡Hola, mundo!" al leer los datos D, hacemos las siguientes modificaciones:

- Para evitar accidentes, renombramos todos los identificadores propios del programa P anteponiéndoles el prefijo `hola_`. Por ejemplo, si hay alguna función `f`, pasa a llamarse `hola_f`.
- Reemplazamos todos los *prints* por nuestra propia versión, la cual acumula lo escrito en memoria y que al ser llamada verifica si lo ya escrito es "¡Hola, mundo!". De ser así, invocan a `f`.

Estas modificaciones pueden realizarse mediante un *script*.

Ejemplo

Llamemos P' al programa resultante. Vemos que P' llama a `f` exactamente si P escribe "¡Hola, mundo!". Si pudiéramos determinar si P' llama a `f`, tendríamos un detector de "¡Hola, mundo!", cosa que sabemos que es imposible de construir. Luego, el problema es indecidible.

$\square$

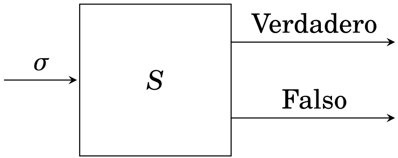

PREPÁRENSE PARA LOS PROBLEMAS.
Y MÁS VALE QUE TEMAN.

Problemas

- Un **problema** P es un lenguaje sobre Σ.
- Una **instancia** del problema P es una palabra $\sigma \in \Sigma^*$.
- **Resolver** el problema P es construir un artefacto S que sea capaz de reconocer dicho lenguaje, es decir:

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

Lo que en el modelo de *cajitas* es equivalente a:



Problemas

Formalmente, hay 2 tipos de problemas:

- **Problema de búsqueda:** se desea hallar la solución.
- **Problema de decisión:** se desea determinar si una palabra pertenece a un lenguaje.

En este curso nos enfocaremos en los segundos, ya que tenemos una base teórica sólida acerca de los lenguajes.

Definiciones

- Un problema $P \subseteq \Sigma^*$ es ***decidible*** si existe un algoritmo capaz de responder en un tiempo finito a la pregunta: ¿$\sigma \in P$?
- **Tesis de Church-Turing:**

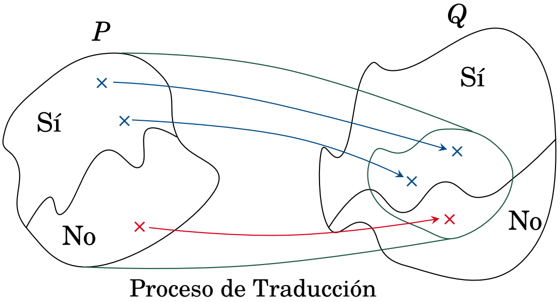
  "Todo lo que puede hacerse mediante computación puede ser realizado por una máquina de Turing"

  "Todo algoritmo es equivalente a una máquina de Turing"
- Un problema P se **reduce** al problema Q si existe un algoritmo que, dado un $\sigma \in \Sigma$, entrega un σ' tal que:

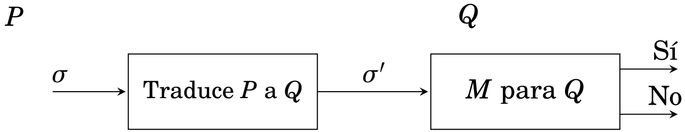
$$\sigma \in P \Leftrightarrow \sigma' \in Q$$

  La reducción se anota como $P \leq Q$, y se lee: Q *es, a lo menos, tan difícil como* P.

Reducción de problemas



Reducción de problemas



Observación: siempre transformamos una instancia del problema *conocido* en una instancia del problema *desconocido*

Máquinas de Turing (TM)

Una *máquina de Turing* (TM) consta de:

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

Donde:

Q : Conjunto finito de estados.

Σ : Alfabeto de entrada.

Γ : Alfabeto de la cinta. $\Sigma \subseteq \Gamma$.

δ : Función de transición. $\delta : Q \times \Gamma \to 2^{Q \times \Gamma \times \{I, D\}}$, donde I es el movimiento del cabezal una posición hacia la izquierda, y D hacia la derecha.

$q_0 \in Q$: Estado inicial.

$B \in \Gamma$: Símbolo blanco, $B \notin \Sigma$. Considere símbolos en $\Gamma \smallsetminus \Sigma$ como "auxiliares".

$F \subseteq Q$: Estados finales. Para $q \in F$ se cumple $\delta(q, a) = \varnothing$.

TM - Ejemplo

Supongamos que necesitamos construir una TM que determine si un *string* binario contiene una cantidad impar de $0's$. La idea es tener dos estados, uno para cuando se haya leído una cantidad par de 0's y otro para cuando se haya leído una cantidad impar.

TM - Ejemplo

Supongamos que necesitamos construir una TM que determine si un *string* binario contiene una cantidad impar de $0's$. La idea es tener dos estados, uno para cuando se haya leído una cantidad par de 0's y otro para cuando se haya leído una cantidad impar.
Con eso en mente, definimos nuestra TM M:

$$Q = \{q_{\text{Par}}, q_{\text{Impar}}, q_Y, q_N\}$$

$$\Sigma = \{0, 1\}$$

$$\Gamma = \{0, 1, B\}$$

$$q_0 = q_{\text{Par}}$$

$$B = B$$

$$F = \{q_Y, q_N\}$$

TM - Ejemplo

La función de transición δ se muestra en la siguiente tabla:

Estado	**0**	**1**	**B**
q_{Par}	$(q_{\text{Impar}}, 0, D)$	$(q_{\text{Par}}, 1, D)$	$(q_N, B, -)$
q_{Impar}	$(q_{\text{Par}}, 0, D)$	$(q_{\text{Impar}}, 1, D)$	$(q_Y, B, -)$

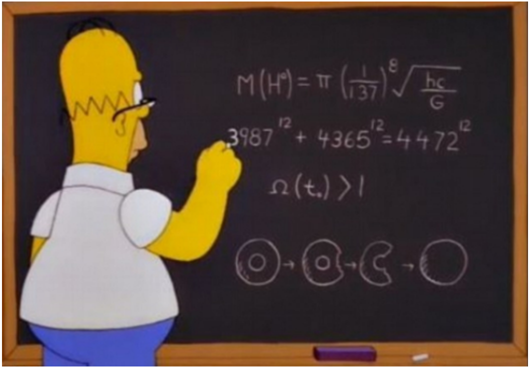
Note que nuestra TM tiene 2 estados finales: uno de aceptación (q_Y) y uno de rechazo (q_N). Además, **siempre se detiene**.

TM - Ejemplo

Implementamos nuestra TM en el simulador:

```
name: Cantidad impar de ceros
init: qPar
accept: qImpar

// Se lee: si estoy en qPar y leo un 0, entonces me cambio a qImpar,
     escribo un 0 en la cinta y muevo el cabezal hacia la derecha.
qPar,0
qImpar,0,>

qImpar,0
qPar,0,>

qPar,1
qPar,1,>

qImpar,1
qImpar,1,>
```

Ejercicios


$M(H^0) = \pi \left(\frac{1}{137}\right)^8 \sqrt{\frac{hc}{G}}$
$3987^{12} + 4365^{12} = 4472^{12}$
$\Omega(t_0) > 1$

Ejercicio 1

Esboce cómo puede usarse la no decibilidad del problema de detectar “Hola, mundo” para demostrar que el determinar si alguna vez se asigna un valor a la variable x no es decidible.

Solución Ejercicio 1

Reduciremos el problema de detectar *Hola Mundo* al problema de determinar si se le asigna un valor a la variable x; si pudiéramos resolver este último, podríamos resolver el primero. Tomemos un programa cualquiera. Lo modificaremos de tal forma que sólo cuando escribe *Hola Mundo* le asigna un valor a la variable x.

Solución Ejercicio 1

Para evitar accidentes, cambiamos el nombre a todas las variables x para no alterar el significado del programa. Por ejemplo, podríamos anteponer una w a cada identificador del programa. Esto es simple de hacer. Luego identificamos todos los puntos en que escribe, y registramos lo escrito en una nueva variable auxiliar.

Solución Ejercicio 1

Si lo último que ésta contiene es *Hola mundo*, ejecutamos la sentencia $x = 1$. Si pudiéramos determinar si se asigna un valor a x, podríamos determinar si un programa escribe *Hola Mundo*, lo que sabemos imposible. Por lo tanto, este problema es indecidible.

□

Ejercicio 2

Comente las siguientes afirmaciones:

- Es buena idea demostrar que un problema P2 es indecidible reduciéndolo a otro indecidible conocido P1, es decir, demostrar la proposición “si P1 es decidible, entonces P2 es decidible”.
- Es buena idea demostrar que un problema P2 es indecidible reduciendo un problema P1 conocido e indecidible a P2, es decir, demostrar “si P2 es decidible, entonces P1 es decidible”.

Solución Ejercicio 2

Reducir P_2 desconocido a P_1 conocido ($Q \rightarrow P$): realizar esto sería incorrecto, ya que no podemos asegurar con certeza de que se mapearán todas las instancias de P_2 en P_1. Además, con este sentido de reducción afirmamos que $P_2 \leq P_1$, es decir, que P_1 es **al menos** tan difícil de resolver como P_2, pero no sabemos nada acerca del *comportamiento* de P_2. Debido a esto, hacer la reducción en este sentido no nos permitiría hacer conclusiones válidas respecto al *comportamiento* de P_2.

Solución Ejercicio 2

Reducir P_1 conocido a P_2 desconocido ($P \rightarrow Q$): realizar esto sí es buena idea, ya que podemos asegurar que todas las instancias de P_1 serán mapeadas en P_2, por lo que si P_2 tiene solución entonces P_1, que está "totalmente contenido en P_2", tiene solución. Esto se puede afirmar con la relación que se desprende de la reducción: $P_1 \leq P_2$, es decir, que P_2 es **al menos** tan difícil de resolver como P_1. Como conocemos el *comportamiento* de P_1, podemos deducir el comportamiento de P_2. Es más, para demostrar que un problema es indecidible usamos el contra-positivo de esta proposición:

$$P_1 \text{ es indecidible } \Rightarrow P_2 \text{ es indecidible}$$

Ejercicio 3

Construya una TM que, dado dos números enteros positivos **x** e **y**, compute $x+y$. Para esto suponga lo siguiente:

- $\Sigma = \{0,1\}$
- $\Gamma = \{0,1,B\}$
- **x** e **y** están separados por un 0 en la cinta y serán representados mediante 1's tal que hay tantos 1's como la magnitud de x, luego un 0, y finalmente tantos 1's como la magnitud de y. Por ejemplo: para **(x,y) =** (3,2), en la cinta encontraremos $B\underbrace{111}_{x}0\underbrace{11}_{y}B$.

Solución Ejercicio 3

Tenemos muchas formas para resolver este ejercicio, sin embargo nos iremos por la más fácil de implementar. La idea es la siguiente:

- Si tenemos **x** veces 1's, un 0 y luego **y** veces 1's, entonces si pudiésemos eliminar el 0 existente entre x e y, ya habríamos computado x + y.
- Por ejemplo: para $(x,y) = (3,2)$ queremos hacer lo siguiente:

$$111011 \rightarrow 11111$$

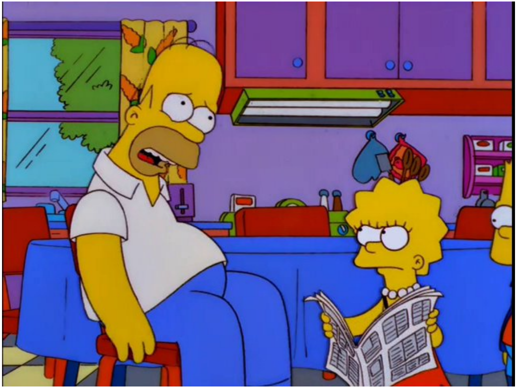
- Para lograr este resultado debemos reemplazar el 0 entre medio por un 1, y luego eliminar el último 1 de la cinta para que nuestro resultado sea consistente con el cómputo.

Solución Ejercicio 3

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ nuestra TM buscada, donde $Q = \{q_0, q_1, q_2\}$ y $F = \{q_2\}$. Entonces, δ queda definida en la siguiente tabla de transiciones:

Estado	**0**	**1**	**B**
q_0	$(q_0, 1, R)$	$(q_0, 1, R)$	(q_1, B, L)
q_1	-	(q_2, B, L)	-

¿Dudas?



INF155 - Informática Teórica

Ayudantía #7

NP-Complete Warriors

Semestre 2022-2

UNIVERSIDAD TECNICA FEDERICO SANTA MARIA
DEPARTAMENTO DE INFORMÁTICA