

# Practica 3 FTC Indecibilidad

## Ejercicio 1. ¿Qué es una MT universal?

- **Definición:** Una MTU es una máquina de Turing capaz de simular cualquier otra máquina de Turing ( $M$ ) dada su descripción codificada y una entrada  $w$ .
- **Funcionamiento:**
  - Recibe como entrada ( $\langle M \rangle, w$ ), donde  $\langle M \rangle$  es la codificación de  $M$  y  $w$  es la cadena de entrada.
  - Simula paso a paso el comportamiento de  $M$  sobre  $w$  usando cintas de trabajo.
  - Acepta o rechaza según lo haría  $M$  con  $w$ .
- **Codificación:**
  - Tanto las MT como las cadenas se representan en un formato estandarizado (ej., binario).
  - La codificación incluye estados, transiciones, símbolos y movimientos (D, L, E).
- **Enumeración de MT:**
  - Las MT pueden ordenarse canónicamente (por longitud de código y orden alfanumérico).
  - Existe una MT que, dado un índice  $i$ , genera la codificación de la  $i$ -ésima MT ( $\langle M_i \rangle$ ).
- **Importancia:**
  - Base para demostrar la **indecidibilidad** de problemas (ej., el problema de la parada).
  - Precursora del concepto de **programa almacenado** en computación.

## Ejercicio 2. Explicar cómo enumeraría los números naturales pares, los números enteros, los números racionales (o fraccionarios), y las cadenas de $\Sigma^*$ siendo $\Sigma = \{0, 1\}$ .

Para enumerar cada uno de estos (que es posible) lo haremos de la siguiente forma:

- *Naturales pares:* Arrancamos con  $i = 0$ , y en cada paso para mostrar el par sumamos 2.
- *Números enteros:* Arrancamos con  $i = 0$ , cada paso aumenta en 1 el  $i$ , y a partir de 1 mostramos el número con el símbolo negativo y luego el valor absoluto de  $i$ , por ejemplo 0, -1, 1, -2, 2, ...
- *Números racionales:* En este caso deberemos llevar 2 números contadores (cadenas),  $i$  y  $j$  por ejemplo:
  - Puede ser sumando denominador y numerador. Ejemplo, arranco en 1 y genero todas las fracciones donde sumando numerador y denominador me de 1 (0/1), luego las que den 2 (1/1), luego 3 (1/2, 2/1..), etc. Así garantizo una generación de racionales posibles por cada natural que uso como índice, e incluso puedo evitar las fracciones simplificables.
  - Evidentemente hay que skippear las imposibles como 0/0, y 1/0, etc.
- $\Sigma^*$  siendo  $\Sigma = \{0, 1\}^*$ :

1.  $i = 0$
2. Escribo en una cinta una cantidad de ceros igual a  $i$  (enunciamos el 0 con  $i$  dígitos)
3.  $j = 0$
4. Sumo 1 a la cadena binaria: arrancando del lado derecho, si encuentro un 1, lo cambio por un 0 y voy hacia la izquierda cambiando todos los 1 por 0 hasta llegar un 0 y lo cambio por un 1 (Suma binaria).
5.  $j++$ , si  $j < i^2$  entonces repito el paso 4, sino  $i++$  y vuelvo al paso 2.
  - En este caso no hay repetidos porque hay que escribir todo el lenguaje y aunque  $0 = 00$  en binario no calienta. No hay que manejar overflow porque cuando llega todos 1 corta dado que sería la  $i^2$  combinación.

**Ejercicio 3. Dar la idea general de cómo sería una MT que, teniendo como cadena de entrada un número natural  $i$ , genera la  $i$ -ésima fórmula booleana satisfactible según el orden canónico. Comentario: asumir que existen una MT  $M1$  que determina si una cadena es una fórmula booleana, y una MT  $M2$  que determina si una fórmula booleana es satisfactible.**

Este problema requiere primero que la formulas booleanas sean enumerables, ordenables de mayor a menor y según orden alfanumérico para desempatar. Suponiendo que logramos codificar las formulas booleanas en binario, podemos hacer lo siguiente:

- La siguiente MT  $M$  obtiene la combinación  $i$ -ésima (Formula  $i$ ) según el orden canónico.

Dado  $i$ , la MT  $M$  hace:

1. Hace  $n := 0$ .
2. Genera la siguiente cadena  $v$  según el orden canónico.
3. Aplica  $M1$ , y luego  $M2$ . Si alguna rechaza vuelve al paso 2.
4. Si  $n = i$ , devuelve  $v$  y para. Si  $n \neq i$ , hace  $n := n + 1$  y vuelve al paso 2.

Es lo mismo que crear una maquina de turing que enumere maquinas de turing.

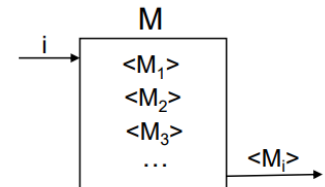
## Enumeración de las máquinas de Turing

- Codificar las máquinas de Turing permite enumerarlas, por ejemplo en el **orden canónico**:
  - Los códigos se ordenan de menor a mayor longitud.
  - Los códigos con longitudes iguales se ordenan según el orden alfanumérico.
- Por ejemplo, las primeras cadenas según el orden canónico, con ceros, unos, paréntesis y comas, son:

$\lambda$	0	1	(	)	,	00	01	0(	0)	0,	10	11	1(	1)	1,	(0	(1	((	...
	000	001	00(	00)	00,	010	011	01(	01)	01,	0(0	0(1	0((	...	etc.				

- La siguiente MT  $M$  obtiene la MT  $i$ -ésima ( $MT M_i$ ) según el orden canónico. Dado  $i$ , la MT  $M$  hace:

- Hace  $n := 0$ .
- Genera la siguiente cadena  $v$  según el orden canónico.
- Valida que  $v$  sea el código de una MT. Si no lo es, vuelve al paso 2.
- Si  $n = i$ , devuelve  $v$  ( **$v$  es el código de la MT  $M_i$** ) y para.  
Si  $n \neq i$ , hace  $n := n + 1$  y vuelve al paso 2.



07

**Ejercicio 4. Sea  $M_1$  una MT que genera en su cinta de salida todas las cadenas de un lenguaje  $L$ . Dar la idea general de cómo sería una MT  $M_2$  que, usando  $M_1$ , acepte una cadena  $w$  si  $w \in L$ .**

Cada vez que  $M_1$  genere una cadena en su cinta de salida lo comparamos con el  $w$ , si  $w$  y la cadena generada son iguales entonces acepta, sino repite hasta que haga todas las cadenas del lenguaje  $L$ , si este es finito o quedara loopeando. (Lo corregí porque antes asumía que  $M_1$  tenía un orden de generación).  
Corrección: *Quizas la forma de resolver sin asumir nada extra es, como  $M_2$  utiliza o simula  $M_1$ , lo que puede hacer  $M_2$  es cada vez que  $M_1$  genera un string en su cinta de salida, comprarlo con  $w$ . Si es igual acepta, sino repite lo mismo hasta que eventualmente (o no) encuentre que  $M_2$  genere una cadena igual a  $w$ . Con esto omitimos asumir un orden de generación en  $M_1$  que no sabemos si existe.*

**Ejercicio 5. El lenguaje  $LU = \{(M, w) \mid M \text{ acepta } w\}$  se conoce como lenguaje universal, y representa el problema general de aceptación. Probar que  $LU \in RE$ . Ayuda: construir una MT que acepte  $LU$ .**

Esencialmente me esta pidiendo que haga la maquina universal, obviamente vamos saltarnos como simulas cada maquina específicamente y asumimos que se puede hacer:

- Separamos el la maquina en su cinta y luego la entrada en otra cinta.

2. Simulamos el siguiente paso de  $M$  sobre  $w$ , si acepta aceptamos, si rechaza rechazamos y si no para seguimos ejecutando pasos.

MTU esta en RE entonces:

- **Acepta correctamente:**  $U$  acepta  $(M, w)$  si y solo si  $M$  acepta  $w$ , por lo que  $U$  reconoce  $LU$ .
- **No decide  $LU$ :** Si  $M$  no se detiene con  $w$ ,  $U$  tampoco se detiene, lo que significa que  $U$  no es un decidor para  $LU$  (no resuelve el problema de la parada). Por lo tanto,  $LU$  es RE pero no recursivo (no está en  $R$ ).

En este caso MTU semidecide  $LU$ , ya que todo depende de la maquina de entrada, si la maquina de entrada es  $R$  entonces seria  $R$ , pero como puede entrar cualquier  $M$  y  $w$  esta en RE.

**Ejercicio 6. Una función  $f : A \rightarrow B$  es total computable si existe una MT  $M_f$  que la computa para todo elemento  $a \in A$ . Sea la función  $f_{01} : \Sigma^* \rightarrow \{0, 1\}$  tal que:  $f_{01}(v) = 1$ , si  $v = \langle M, w \rangle$  y  $M$  para a partir de  $w$ .  $f_{01}(v) = 0$ , si  $v = \langle M, w \rangle$  y  $M$  no para a partir de  $w$  o bien  $v \neq \langle M, w \rangle$ . Probar que  $f_{01}$  no es total computable. Ayuda: ¿con qué problema se relaciona dicha función?**

En este caso se relaciona con el HP porque:

- $f : A \rightarrow B$  se aplica a una función  $f_{01}$  que recibe cualquier cadena  $v$  (que representa un par  $\langle M, w \rangle$ ) y retorna 0 o 1, donde:
  - 1 significa que la máquina  $M$  se detiene con entrada  $w$ ,
  - 0 significa que  $M$  no se detiene con  $w$ .
- Sabemos que  $f_{01}$  no puede existir porque el Problema de la Parada (HP) es indecidible. Pero, para llegar a una contradicción, supongamos que existe una máquina  $P$  que computa  $f_{01}$ .
- Ahora construimos una máquina  $D$  que toma  $\langle M \rangle$  como entrada y hace lo siguiente:
  - Ejecuta  $P(\langle M \rangle, \langle M \rangle)$  (es decir, consulta si  $M$  se detiene con su propio código).
  - Si  $P$  retorna 1 (o sea,  $M$  para con  $\langle M \rangle$ ), entonces  $D$  entra en un loop infinito.
  - Si  $P$  retorna 0 ( $M$  no para con  $\langle M \rangle$ ), entonces  $D$  se detiene.
- Esto lleva a una paradoja cuando  $D$  recibe su propio código  $\langle D \rangle$ :
  - Si  $P(\langle D \rangle, \langle D \rangle)$  retorna 1 (o sea,  $D$  para con  $\langle D \rangle$ ), entonces  $D$  no para (por su construcción).
  - Si  $P(\langle D \rangle, \langle D \rangle)$  retorna 0 ( $D$  no para con  $\langle D \rangle$ ), entonces  $D$  sí para.
- Como ambas posibilidades son contradictorias, la suposición inicial (que  $P$  existe) es falsa.

**Ejercicio 7. Responder breve y claramente cada uno de los siguientes incisos (en todos los casos, las MT mencionadas tienen una sola cinta):**

**a. Probar que se puede decidir si una MT  $M$ , a partir de la cadena vacía  $\lambda$ , escribe alguna vez un símbolo no blanco. Ayuda: ¿Cuántos pasos puede hacer  $M$  antes de entrar en un loop?**

Depende de la cantidad de estados, técnicamente sería  $n \cdot 2$  donde  $n$  es la cantidad de estados es  $n$  y 2 porque es la cantidad de posibles "símbolos" (blanco y no blanco). En este caso NO nos importa la longitud de la cinta puesto que estaríamos trabajando implícitamente con un solo lugar.

**b. Probar que se puede decidir si una MT  $M$  que sólo se mueve a la derecha, a partir de una cadena  $w$ , para, Ayuda: ¿Cuántos pasos puede hacer  $M$  antes de entrar en un loop?**

Es decidible, el hecho que no pueda volver a lugares anteriores evita loops, hace máximo  $p$  pasos donde  $p$  es la longitud de la cadena  $w$  y al llegar a  $p+1$  loopea.

**c. Probar que se puede decidir si dada una MT  $M$ , existe una cadena  $w$  a partir de la cual  $M$  para en a lo sumo 10 pasos. Ayuda: ¿Hasta qué tamaño de cadenas hay que chequear?**

Si hay una cadena  $w$  a partir de la cual  $M$  para en a lo sumo 10 pasos esto representa que hay un máximo de pasos, 10.

**d. ¿Se puede decidir si dada una MT  $M$ , existe una cadena  $w$  de a lo sumo 10 símbolos a partir de la cual  $M$  para? Justificar la respuesta.**

Es decidible, si aplicamos formulas sabremos que para cada  $w$  de tamaño  $K$  tendremos máximo de  $K \cdot N_1 \cdot N_2^K$  pasos para completarla antes de que se detecte un loop, en este caso basta con generar uno por una cada cadena y ejecutarla durante la cantidad de pasos máxima, si se pasa vamos a la siguiente, si acepta aceptamos, y si genera todas rechazamos.

Es como la versión acotada del HP.