# Práctica 7

## 1. Construya una MTN que genere de manera no determinística todos los números de 8 bits. Es decir que, dado cualquier número, alguna computación de la máquina lo generará. ¿Cuántos movimientos hace la máquina?

Se quieren generar todos los números de 8 bits, es decir, desde $00000000$ hasta $11111111$.

Se define $M = \langle Q, \Sigma, \Gamma, \Delta, q_0, q_A, q_R \rangle$ con la función de transición $\Delta$ definida como:

- $\Delta(q_0, B) = \{(q_1, 0, D), (q_1, 1, D)\}$
- $\Delta(q_1, B) = \{(q_2, 0, D), (q_2, 1, D)\}$
- $\Delta(q_2, B) = \{(q_3, 0, D), (q_3, 1, D)\}$
- $\Delta(q_3, B) = \{(q_4, 0, D), (q_4, 1, D)\}$
- $\Delta(q_4, B) = \{(q_5, 0, D), (q_5, 1, D)\}$
- $\Delta(q_5, B) = \{(q_6, 0, D), (q_6, 1, D)\}$
- $\Delta(q_6, B) = \{(q_7, 0, D), (q_7, 1, D)\}$
- $\Delta(q_7, B) = \{(q_8, 0, D), (q_8, 1, D)\}$
- $\Delta(q_8, B) = \{(q_A, B, S), (q_A, B, S)\}$

Es decir, en cada estado, la máquina puede elegir entre escribir un 0 o un 1 en la cinta y moverse a la derecha, hasta completar 8 bits. Por lo tanto puede generar cualquier cadena posible de 8 bits.

La máquina realiza 8 movimientos para escribir los 8 bits, y luego un movimiento final para detenerse en el estado de aceptación. En total, la máquina hace 9 movimientos.

## 2. Sean $L_1$ y $L_2$ dos lenguajes definidos sobre $\{0,1\}^*$

$$L_1 = \{0^n 1 \mid n \geq 0\}$$

$$L_2 = \{1^n 0 \mid n \geq 0\}$$

### a. Construya una MTN $M$ tal que $L(M) = L_1 \cup L_2$

$L_1 \cup L_2 = \{0^n 1 \mid n \geq 0\} \cup \{1^n 0 \mid n \geq 0\}$

Es decir, el lenguaje contiene cadenas que consisten en una secuencia de ceros seguida de un único uno, o una secuencia de unos seguida de un único cero. La cadena vacía no pertenece a este lenguaje, dado que incluso si $n = 0$, siempre hay un símbolo adicional ($1$ o $0$).

- Las primeras cadenas de $L_1$ serían: $1, 01, 001, 0001, 00001, \ldots$
- Las primeras cadenas de $L_2$ serían: $0, 10, 110, 1110, 11110, \ldots$

Se define $M = \langle Q, \Sigma, \Gamma, \Delta, q_0, q_A, q_R \rangle$ con la función de transición $\Delta$ definida como:

- $\Delta(q_0, B) = \{(q_R, B, S)\}$
- $\Delta(q_0, 0) = \{(q_1, 0, S), (q_2, 0, S)\}$
- $\Delta(q_0, 1) = \{(q_1, 1, S), (q_2, 1, S)\}$
- $\Delta(q_1, 0) = \{(q_1, 0, D)\}$
- $\Delta(q_1, 1) = \{(q_3, 1, D)\}$
- $\Delta(q_1, B) = \{(q_R, B, S)\}$
- $\Delta(q_3, B) = \{(q_A, B, S)\}$
- $\Delta(q_3, 0) = \{(q_R, 0, S)\}$
- $\Delta(q_3, 1) = \{(q_R, 1, S)\}$
- $\Delta(q_2, 1) = \{(q_2, 1, D)\}$
- $\Delta(q_2, 0) = \{(q_4, 0, D)\}$
- $\Delta(q_2, B) = \{(q_R, B, S)\}$
- $\Delta(q_4, B) = \{(q_A, B, S)\}$
- $\Delta(q_4, 0) = \{(q_R, 0, S)\}$
- $\Delta(q_4, 1) = \{(q_R, 1, S)\}$

Lo que hace la máquina es lo siguiente: Desde el estado inicial $q_0$, la máquina puede elegir entre dos caminos no determinísticos:

1. Ir al estado $q_1$ para chequear si la cadena pertenece a $L_1$. Desde $q_1$, la máquina lee ceros hasta encontrar un uno.
   i. Cuando encuentra un uno, va al estado $q_3$ y se mueve a la derecha. En $q_3$, si encuentra un blanco, acepta. Si encuentra cualquier otro símbolo, rechaza.
   ii. Si encuentra un blanco antes de encontrar un uno, rechaza.
2. Ir al estado $q_2$ para chequear si la cadena pertenece a $L_2$. Desde $q_2$, la máquina lee unos hasta encontrar un cero.
   i. Cuando encuentra un cero, va al estado $q_4$ y se mueve a la derecha. En $q_4$, si encuentra un blanco, acepta. Si encuentra cualquier otro símbolo, rechaza.
   ii. Si encuentra un blanco antes de encontrar un cero, rechaza.

### b. Describa la traza de ejecución para las entradas $w_1 = 001$ y $w_2 = 1101$

...

# 3. ¿La reducción polinomial posee las siguientes propiedades? Justifique

## a. Reflexiva

Una relación matemática es reflexiva si para todo elemento $a$ en el conjunto, se cumple que $a$ está relacionado consigo mismo. En el contexto de la reducción polinomial, se quiere verificar si es cierto que para cualquier lenguaje $L$, se cumple que $L \ \alpha_p \ L$.

Para que se cumpla, se deben cumplir dos condiciones:

1. $L \ \alpha \ L$
   i. Esto se cumple puesto que cualquier lenguaje es reducible a sí mismo mediante la función identidad, que simplemente devuelve la misma cadena de entrada sin modificaciones.
2. La función de reducción debe ser computable en tiempo polinomial.
   i. La función identidad es computable en tiempo polinomial, ya que simplemente devuelve la misma cadena de entrada sin modificaciones.

**Por lo tanto, la reducción polinomial es reflexiva**. ✅

## b. Simétrica

Una relación matemática es simétrica si para todo par de elementos $a$ y $b$ en el conjunto, si $a$ está relacionado con $b$, entonces $b$ está relacionado con $a$. En el contexto de la reducción polinomial, se quiere verificar si es cierto que para cualquier par de lenguajes $L_1$ y $L_2$, si $L_1 \ \alpha_p \ L_2$, entonces $L_2 \ \alpha_p \ L_1$.

Esto no se cumple, porque que $L_1$ sea reducible a $L_2$ no necesariamente implica que $L_2$ sea reducible a $L_1$. Contraejemplo: El lenguaje $\text{Halting Problem}$ es reducible a cualquier lenguaje NP-completo, pero ningún lenguaje NP-completo es reducible al $\text{Halting Problem}$.

**Por lo tanto, la reducción polinomial no es simétrica**. ❌

## c. Antisimétrica

Una relación matemática es antisimétrica si para todo par de elementos $a$ y $b$ en el conjunto, si $a$ está relacionado con $b$ y $b$ está relacionado con $a$, entonces $a$ es igual a $b$. En el contexto de la reducción polinomial, se quiere verificar si es cierto que para cualquier par de lenguajes $L_1$ y $L_2$, si $L_1 \ \alpha_p \ L_2$ y $L_2 \ \alpha_p \ L_1$, entonces $L_1 = L_2$.

Esto no se cumple, porque que $L_1$ sea reducible a $L_2$ y viceversa no implica que los lenguajes sean iguales. Contraejemplo:

- $L_1 = \{0^n 1 \mid n \geq 0\}$
- $L_2 = \{1^n 0 \mid n \geq 0\}$

Se cumple que $L_1 \ \alpha_p \ L_2$ y se cumple que $L_2 \ \alpha_p \ L_1$, pero claramente $L_1 \neq L_2$.

**Por lo tanto, la reducción polinomial no es antisimétrica**. ❌

## d. Transitiva

Una relación matemática es transitiva si para toda terna de elementos $a$, $b$ y $c$ en el conjunto, si $a$ está relacionado con $b$ y $b$ está relacionado con $c$, entonces $a$ está relacionado con $c$. En el contexto de la reducción polinomial, se quiere verificar si es cierto que para cualquier terna de lenguajes $L_1$, $L_2$ y $L_3$, si $L_1 \ \alpha_p \ L_2$ y $L_2 \ \alpha_p \ L_3$, entonces $L_1 \ \alpha_p \ L_3$.

Esto se cumple, porque si existe una función de reducción polinomial de $L_1$ a $L_2$ y otra de $L_2$ a $L_3$, entonces se puede componer ambas funciones para obtener una función de reducción polinomial de $L_1$ a $L_3$. La composición de dos funciones computables en tiempo polinomial también es computable en tiempo polinomial.

**Por lo tanto, la reducción polinomial es transitiva**. ✅

## 4. ¿Es cierto que si dos lenguajes $L_1$ y $L_2$ son $NPC$ entonces $L_1 \ \alpha_p \ L_2$, y también $L_2 \ \alpha_p \ L_1$? Justifique su respuesta.

1. Si $L_1 \in NPC$, entonces $L_1 \in NP$ y $L_1 \in NPH$.
2. Si $L_2 \in NPC$, entonces $L_2 \in NP$ y $L_2 \in NPH$.
3. Como $L_1 \in NPH$, por definición, para todo lenguaje $L' \in NP$, se cumple que $L' \ \alpha_p \ L_1$.
4. Como $L_2 \in NP$, se cumple que $L_2 \ \alpha_p \ L_1$.
5. Como $L_2 \in NPH$, por definición, para todo lenguaje $L' \in NP$, se cumple que $L' \ \alpha_p \ L_2$.
6. Como $L_1 \in NP$, se cumple que $L_1 \ \alpha_p \ L_2$.
7. **Por lo tanto, se cumple que $L_1 \ \alpha_p \ L_2$ y $L_2 \ \alpha_p \ L_1$** ✅.

## 5. Sean $L_1$ y $L_2$ tales que $L_1 \ \alpha_p \ L_2$, ¿Qué se puede inferir?

### a. Si $L_1$ está en P entonces $L_2$ está en $P$

1. $L_1 \in P$
2. $L_1 \ \alpha_p \ L_2$, es decir:
   i. Existe una función de reducción $f$ tal que a cada palabra de $L_1$ se le asigna una palabra de $L_2$, y a cada palabra que no pertenece a $L_1$ se le asigna una palabra que no pertenece a $L_2$.
   ii. Esta función $f$ es computable en tiempo polinomial, es decir $f \in P$.
3. Como $L_1 \ \alpha_p \ L_2$, $L_1$ es a lo sumo tan difícil como $L_2$.
4. Contraejemplo:
   i. Sea $L_1 = \{0^n 1 \mid n \geq 0\}$
   ii. Sea $L_2 = HP$
   iii. Se puede escribir una MTD $M_f$ que compute la función de reducibilidad en tiempo polinomial $f$ de $L_1$ a $L_2$.
   iv. En este caso, $L_1 \in P$, pero $L_2 \notin P$, porque $HP \in NPH$ y además $HP \notin P$.

**Por lo tanto la afirmación es falsa**. ❌

### b. Si $L_2$ está en $P$ entonces $L_1$ está en $P$

1. $L_2 \in P$
2. $L_1 \ \alpha_p \ L_2$, es decir:
   i. Existe una función de reducción $f$ tal que a cada palabra de $L_1$ se le asigna una palabra de $L_2$, y a cada palabra que no pertenece a $L_1$ se le asigna una palabra que no pertenece a $L_2$.

ii. Esta función $f$ es computable en tiempo polinomial, es decir $f \in P$.

3. Como $L_1 \ \alpha_p \ L_2$, $L_1$ es a lo sumo tan difícil como $L_2$.
4. Si $L_1 \ \alpha_p \ L_2$ y además $L_2 \in P$, entonces $L_1 \in P$ (Por teorema 3 de la teoría)
5. Como $L_2$ es "fácil" (pertenece a P), y $L_1$ es a lo sumo tan difícil como $L_2$, entonces $L_1$ también debe ser "fácil" (pertenece a P).

**Por lo tanto la afirmación es verdadera**. ✅

# c. Si $L_2$ está en $NPC$ entonces $L_1$ está en $NPC$

1. $L_2 \in NPC$
2. $L_1 \ \alpha_p \ L_2$, es decir:
   i. Existe una función de reducción $f$ tal que a cada palabra de $L_1$ se le asigna una palabra de $L_2$, y a cada palabra que no pertenece a $L_1$ se le asigna una palabra que no pertenece a $L_2$.
   ii. Esta función $f$ es computable en tiempo polinomial, es decir $f \in P$.
3. Como $L_1 \ \alpha_p \ L_2$, $L_1$ es a lo sumo tan difícil como $L_2$.
4. Se puede reducir un lenguaje que está en $P$ a un lenguaje que está en $NPC$.
5. Contraejemplo:
   i. Sea $L_1 = \emptyset$
   ii. Sea $L_2 = SAT$
   iii. Se cumple que $L_1 \ \alpha_p \ L_2$, pero $L_1 \in P$ y $L_2 \in NPC$.

**Por lo tanto la afirmación es falsa**. ❌

# d. Si $L_2$ está en $NPC$ entonces $L_1$ está en $NP$

1. $L_2 \in NPC$
2. $L_1 \ \alpha_p \ L_2$, es decir:
   i. Existe una función de reducción $f$ tal que a cada palabra de $L_1$ se le asigna una palabra de $L_2$, y a cada palabra que no pertenece a $L_1$ se le asigna una palabra que no pertenece a $L_2$.
   ii. Esta función $f$ es computable en tiempo polinomial, es decir $f \in P$.
3. Como $L_1 \ \alpha_p \ L_2$, $L_1$ es a lo sumo tan difícil como $L_2$.
4. Como $L_2$ pertenece a $NPC$, entonces $L_2$ pertenece a $NP$.
5. Dado que $L_1$ es a lo sumo tan difícil como $L_2$, entonces $L_1$ también pertenece a $NP$, o a lo sumo a $P$, que es un subconjunto de $NP$.

**Por lo tanto la afirmación es verdadera**. ✅

## e. Si $L_1$ está en $NPC$ entonces $L_2$ está en $NPC$

1. $L_1 \in NPC$
2. $L_1 \ \alpha_p \ L_2$, es decir:
    i. Existe una función de reducción $f$ tal que a cada palabra de $L_1$ se le asigna una palabra de $L_2$, y a cada palabra que no pertenece a $L_1$ se le asigna una palabra que no pertenece a $L_2$.
    ii. Esta función $f$ es computable en tiempo polinomial, es decir $f \in P$.
3. Como $L_1 \ \alpha_p \ L_2$, $L_1$ es a lo sumo tan difícil como $L_2$.
4. Es posible reducir un lenguaje $NPC$ a uno que no sea $NPC$.
5. Contraejemplo:
    i. Sea $L_1 = SAT$
    ii. Sea $L_2 = \Sigma^*$.
    iii. Se cumple que $L_1 \ \alpha_p \ L_2$, pero $L_1 \in NPC$ y $L_2 \notin NPC$, porque $L_2 \in P$.

**Por lo tanto la afirmación es falsa**. ❌

## f. Si $L_1$ está en $NPC$ y $L_2$ está en $NP$ entonces $L_2$ está en $NPC$

1. $L_1 \in NPC$
2. $L_2 \in NP$
3. $L_1 \ \alpha_p \ L_2$, es decir:
    i. Existe una función de reducción $f$ tal que a cada palabra de $L_1$ se le asigna una palabra de $L_2$, y a cada palabra que no pertenece a $L_1$ se le asigna una palabra que no pertenece a $L_2$.
    ii. Esta función $f$ es computable en tiempo polinomial, es decir $f \in P$.
4. Como $L_1 \ \alpha_p \ L_2$, $L_1$ es a lo sumo tan difícil como $L_2$.
5. Si $L_1 \in NP$ y $L_2 \in NP$, y $L_1 \ \alpha_p \ L_2$, entonces si $L_1 \in NPC$, se cumple que $L_2 \in NPC$ (Por teorema 6 de la teoría)
6. Como se sabe que $L_1 \in NPC$, esto implica que $L_1 \in NP$ y además $L_1 \in NPH$.
7. Además se sabe que $L_2 \in NP$.
8. Por ende ambas condiciones del teorema se cumplen, y se puede concluir que $L_2 \in NPC$.

**Por lo tanto la afirmación es verdadera**. ✅

# 6. Decir si las siguientes afirmaciones son verdaderas o falsas y justificar

## a. Si $P = NP$ entonces todo lenguaje de $NPC$ pertenece a $P$

1. Se asume que $P = NP$.
2. Esto implica que todos los lenguajes que pertenecen a $NP$ también pertenecen a $P$ y viceversa.
3. Como todo lenguaje de $NPC$ pertenece, por definición, a $NP$, entonces todos los lenguajes de $NPC$ también pertenecen a $P$.

**Por lo tanto la afirmación es verdadera**. ✅

## b. Si $P = NP$ entonces todo lenguaje de $NPH$ pertenece a $P$

1. Se asume que $P = NP$.
2. Esto implica que todos los lenguajes que pertenecen a $NP$ también pertenecen a $P$ y viceversa.
3. Sin embargo, como se sabe que algunos lenguajes $NPH$ NO son decidibles, (por ejemplo $HP$), nunca puede ocurrir que todos los lenguajes de $NPH$ pertenezcan a $P$, ya que todo lenguaje en $P$ es decidible, por definición.

**Por lo tanto la afirmación es falsa**. ❌

# 7. ¿Qué se puede decir respecto del problema del viajante de comercio (TSP) si se sabe que es $NPC$, y se asume que $P \neq NP$?

## a. No existe un algoritmo que resuelva instancias de TSP

1. Se sabe que TSP es $NPC$.
2. Como TSP es $NPC$, entonces TSP pertenece a $NP$ también.
3. Todo problema que pertenece a $NP$ es decidible.
4. Como TSP es decidible, existe una máquina de Turing $M$ que lo decide y que siempre termina para todo input.
5. Sí existe un algoritmo que resuelve instancias de TSP.

**Por lo tanto la afirmación es falsa**. ❌

## b. No existe un algoritmo que eficientemente resuelva instancias de TSP

1. Se asume que $P \neq NP$.
2. Se sabe que TSP es $NPC$.
3. Como TSP es $NPC$, entonces TSP pertenece a $NP$ también.
4. Existe un algoritmo que resuelve instancias de TSP, porque TSP es decidible (al ser $NP$). Sin embargo, ese algoritmo no es eficiente.
5. Si existiese un algoritmo eficiente que resuelva TSP, entonces TSP pertenecería a $P$.
6. Pero como se sabe que TSP pertenece a $NP$, si perteneciese a $P$ también, esto implicaría que $P = NP$, lo cual contradice la suposición inicial de que $P \neq NP$.
7. Este algoritmo eficiente no puede existir.

**Por lo tanto la afirmación es verdadera**. ✅

## c. Existe un algoritmo que eficientemente resuelve instancias de TSP, pero nadie lo ha encontrado

1. Se asume que $P \neq NP$.
2. Se sabe que TSP es $NPC$.
3. Como TSP es $NPC$, entonces TSP pertenece a $NP$ también.
4. Si existe un algoritmo que resuelve instancias de TSP eficientemente, entonces TSP pertenecería a $P$.
5. Pero como se sabe que TSP pertenece a $NP$, si perteneciese a $P$ también, esto implicaría que $P = NP$, lo cual contradice la suposición inicial de que $P \neq NP$.
6. Este algoritmo no puede existir jamás bajo la suposición de que $P \neq NP$.

**Por lo tanto la afirmación es falsa**. ❌

## d. TSP no está en $P$

1. Se asume que $P \neq NP$.
2. Se sabe que TSP es $NPC$.
3. Como TSP es $NPC$, entonces TSP pertenece a $NP$ también.
4. Como se asume que $P \neq NP$, y se sabe que TSP es $NP$, entonces TSP no puede pertenecer a $P$.

**Por lo tanto la afirmación es verdadera**. ✅