Práctica 4

1. Construir una máquina de Turing que escriba en la primera cinta las palabras de $\{0, 1\}^*$ en orden canónico separadas por un símbolo ";". Obviamente esta máquina nunca se detiene.

Las primeras palabras de $\{0, 1\}^*$ en orden canónico son:
$\lambda; 0; 1; 00; 01; 10; 11; 000; 001; 010; 011; 100; 101; 110; 111; \ldots$

Es decir, se debe construir una MT que vaya sumando 1 en binario hasta el infinito, empezando desde la cadena vacía λ.

Para esto se construirá una MT de dos cintas.

```
// Inicialmente la cinta está vacía, por lo que se escribe la cadena vacía seguida de ";"
q0, B, B
q1, ;, D, 0, S

// Se copia la segunda cinta a la primera y se pone el símbolo ";"
q1, B, 0
q1, 0, D, 0, D

q1, B, 1
q1, 1, D, 1, D

q1, B, B
q2, ;, S, B, I

// Se invierte el bit del extremo derecho de la segunda cinta
q2, ;, 0
q3, ;, S, 1, I

q2, ;, 1
q4, ;, S, 0, I

// Se mueve al extremo izquierdo de la segunda cinta
q3, ;, 0
q3, ;, S, 0, I

q3, ;, 1
q3, ;, S, 1, I

q3, ;, B
q1, ;, D, B, D

// Suma binaria con overflow
q4, ;, 0
q3, ;, S, 1, I

q4, ;, 1
q4, ;, S, 0, I

q4, ;, B
q1, ;, D, 0, S
```

2. Sean $\Sigma = \{a, b\}$ y $\mathscr{L}$ el conjunto de todos los lenguajes definidos sobre Σ. Diga si las siguientes afirmaciones son verdaderas o falsas:

a. $\mathscr{L} - R = \emptyset$

1. $\mathscr{L}$ incluye a todos los lenguajes posibles sobre el alfabeto Σ.
2. R son los lenguajes decidibles.
3. $\mathscr{L} - R$ son todos los lenguajes que no son decidibles.
4. Este conjunto no es vacío porque existen lenguajes recursivamente enumerables que no son decidibles.
5. Contraejemplo: $\overline{L_D} \in \mathscr{L}$ pero $\overline{L_D} \notin R$

Por lo tanto la afirmación es falsa. ❌

b. $RE - R \neq \emptyset$

1. RE son todos los lenguajes recursivamente enumerables.
2. R son todos los lenguajes decidibles.
3. $RE - R$ son todos los lenguajes que son recursivamente enumerables pero no decidibles.
4. Este conjunto no es vacío porque existen lenguajes que son recursivamente enumerables pero no decidibles.
5. Ejemplo: $L_u \in RE$ pero $L_u \notin R$

Por lo tanto la afirmación es verdadera. ✅

c. $\{\lambda\} \in (\mathscr{L} - \text{CO-RE})$

1. $\{\lambda\}$ es el lenguaje que contiene solo la cadena vacía. Este lenguaje es decidible porque todo lenguaje finito es decidible. $\{\lambda\} \in R$.
2. $\mathscr{L}$ incluye a todos los lenguajes posibles sobre el alfabeto Σ.
3. CO-RE son todos los lenguajes cuyo complemento es recursivamente enumerable.
4. Como $R \subset \text{CO-RE}$, se tiene que $\{\lambda\} \in \text{CO-RE}$.
5. Por lo tanto, $\{\lambda\} \in \text{CO-RE}$ implica que $\{\lambda\} \notin (\mathscr{L} - \text{CO-RE})$.

Por lo tanto la afirmación es falsa. ❌

d. $RE \cup R = \mathscr{L}$

1. RE son todos los lenguajes recursivamente enumerables.
2. R son todos los lenguajes decidibles.
3. $\mathscr{L}$ incluye a todos los lenguajes posibles sobre el alfabeto Σ.
4. Se sabe que $R \subseteq RE$.
5. Por lo tanto $RE \cup R = RE$.
6. Se reformula la afirmación como $RE = \mathscr{L}$.
7. Esto no se cumple porque existen lenguajes que no son recursivamente enumerables.
8. Contraejemplo: $\overline{L_u} \in \mathscr{L}$ pero $\overline{L_u} \notin RE$

Por lo tanto la afirmación es falsa. ❌

e. $\Sigma^* \in R$

1. Σ^* es el conjunto de todas las cadenas posibles formadas por símbolos del alfabeto Σ.
2. R son todos los lenguajes decidibles.
3. $\Sigma^* \in R$ se cumple porque se puede construir una MT que acepte todas las cadenas posibles del lenguaje y siempre se detenga:
    i. $\delta(q_0, B) = (q_A, B, S)$
    ii. $\delta(q_0, x) = (q_A, x, S) \quad \forall x \in \Sigma.$

Por lo tanto la afirmación es verdadera. ✅

f. $\emptyset \in RE$

1. $\emptyset$ es el lenguaje vacío, es decir, el lenguaje que no contiene ninguna cadena.
2. RE son todos los lenguajes recursivamente enumerables.
3. $\emptyset \in RE$ se cumple porque se puede construir una MT que rechaza todas las cadenas posibles del lenguaje:
    i. $\delta(q_0, B) = (q_R, B, S)$
    ii. $\delta(q_0, x) = (q_R, x, S) \quad \forall x \in \Sigma.$

Por lo tanto la afirmación es verdadera. ✅

g. $\text{CO-RE} = RE$

1. RE son todos los lenguajes recursivamente enumerables.
2. CO-RE son todos los lenguajes cuyo complemento es recursivamente enumerable.

3. $RE \neq \text{CO-RE}$ porque existen lenguajes que son recursivamente enumerables pero cuyo complemento no lo es.
4. Si la afirmación fuera cierta, se cumpliría que $\text{CO-RE} - RE = \emptyset$.
5. Contraejemplo: $L_D \in (\text{CO-RE} - RE)$, por lo que lo anterior no se cumple.

Por lo tanto la afirmación es falsa. ❌

h. $(\mathscr{L} - RE) = \text{CO-RE}$

1. $\mathscr{L}$ incluye a todos los lenguajes posibles sobre el alfabeto Σ.
2. RE son todos los lenguajes recursivamente enumerables.
3. CO-RE son todos los lenguajes cuyo complemento es recursivamente enumerable.
4. $(\mathscr{L} - RE) = \text{CO-RE}$ no se cumple porque existen lenguajes que no son recursivamente enumerables y cuyo complemento tampoco lo es.
5. Contraejemplo: $L \in (\mathscr{L} - RE)$ pero $L \notin \text{CO-RE}$, con L el lenguaje visto en la teoría.

Por lo tanto la afirmación es falsa. ❌

i. $ab \in \Sigma^*$

1. Σ^* es el conjunto de todas las cadenas posibles formadas por símbolos del alfabeto Σ.
2. ab es una cadena válida formada por símbolos del alfabeto $\Sigma = \{a, b, \dots\}$.
3. $ab \in \Sigma^*$ se cumple porque es una cadena posible formada por símbolos del alfabeto.

Por lo tanto la afirmación es verdadera. ✅

j. $\text{CO-R} \subset \text{CO-RE}$

1. CO-R son todos los lenguajes cuyo complemento es decidible.
2. CO-RE son todos los lenguajes cuyo complemento es recursivamente enumerable.
3. Se sabe que $R = \text{CO-R}$.
4. Reemplazando, se tiene: $R \subset \text{CO-RE}$.
5. Lo anterior se cumple por teorema.

Por lo tanto la afirmación es verdadera. ✅

k. $a \in R$

1. R son todos los lenguajes decidibles.
2. a es un símbolo individual del alfabeto Σ.

3. $a \notin R$ porque R está formado por lenguajes (que son conjuntos), no por símbolos individuales. Por lo tanto, a no puede pertenecer a R.

Por lo tanto la afirmación es falsa. ❌

I. $\{a\} \in RE$

1. RE son todos los lenguajes recursivamente enumerables.
2. $\{a\}$ es el lenguaje que contiene únicamente la cadena "a".
3. Este lenguaje es recursivamente enumerable porque se puede construir una MT que acepte la cadena "a" y rechace todas las demás cadenas:
   i. $\delta(q_0, a) = (q_A, a, S)$
   ii. $\delta(q_0, B) = (q_R, B, S)$
   iii. $\delta(q_0, x) = (q_R, x, S) \quad \forall x \in (\Sigma - a)$.

Por lo tanto la afirmación es verdadera. ✅

3. Si $L \in (RE - R)$

a. ¿Existirá alguna máquina de Turing que rechace parando en q_R si su entrada está en L y rechace loopeando si su entrada no está en L?

Sí, existe una MT que cumple con las condiciones:

1. Para cada cadena válida $w \in L$, la máquina va al estado de rechazo q_R y se detiene.
2. Para cada cadena inválida $z \notin L$, la máquina va a un estado de loopeo q_R y nunca se detiene.

Se puede construir con las siguientes deltas:

- $\delta(q_0, w) = (q_R, w, S) \quad \forall w \in L$
- $\delta(q_0, z) = (q_L, z, S) \quad \forall z \notin L$
- $\delta(q_L, z) = (q_L, z, S)$

b. ¿Existirá alguna máquina de Turing que rechace loopeando si su entrada está en L y rechace parando en q_R si su entrada no está en L?

No, porque como se sabe que L no es decidible, no existe una MT que siempre rechace todas las cadenas que no pertenecen a L deteniéndose en el estado de rechazo q_R.

Si la máquina se detuviera para todas las cadenas que no están en L, entonces estaría decidiendo el lenguaje, lo cual contradice la premisa de que $L \in (RE - R)$.

c. De existir, que lenguaje reconocería esta máquina de Turing?

Para el inciso a), la máquina reconoce el lenguaje vacío, $L = \emptyset$.

4. Sea $L = \{w \mid$ Existe alguna Máquina de Turing M que acepta w$\}$ ¿$L \in R$? Justifique.

L es otra forma de escribir al lenguaje Σ^*, porque todas las cadenas posibles w pueden ser aceptadas por alguna máquina de Turing M. Por lo tanto, $L = \Sigma^*$.

$\Sigma^* \in R$ se cumple porque existe una máquina de Turing M que acepta todas las cadenas posibles del lenguaje y siempre se detiene. Por ejemplo, la máquina con la siguiente delta:

- $\delta(q_0, B) = (q_A, B, S)$
- $\delta(q_0, x) = (q_A, x, S) \quad \forall x \in \Sigma$

5. Conteste y justifique:

a. ¿$\mathscr{L}$ es un conjunto infinito contable?

1. $\mathscr{L}$ es el conjunto de todos los lenguajes posibles sobre un alfabeto Σ.
2. Cada lenguaje $L \in \mathscr{L}$ es un subconjunto de Σ^*.
3. Por lo tanto, el conjunto de todos los lenguajes $\mathscr{L}$ es el conjunto de partes del conjunto Σ^*: $\mathscr{L} = \mathcal{P}(\Sigma^*)$.
4. Se sabe que todo conjunto de partes de un conjunto infinito contable es un conjunto infinito incontable (por teorema de Cantor).
5. Por lo tanto, $\mathscr{L}$ no es un conjunto infinito contable, si no **incontable**.

b. ¿RE es un conjunto infinito contable?

1. RE es el conjunto de todos los lenguajes recursivamente enumerables.

2. Para que RE sea infinito contable, debe existir una correspondencia uno a uno con los números naturales, es decir, para cada lenguaje $L \in RE$ debe existir un número natural $n \in \mathbb{N}$.
3. Se sabe que el conjunto de todas las máquinas de Turing es infinito contable.
4. Además se sabe que cada lenguaje RE, por definición, posee al menos una máquina de Turing que lo reconoce.
5. Por lo tanto, RE es un conjunto infinito contable, ya que para todo lenguaje recursivamente enumerable L, existe al menos una máquina de Turing M que lo reconoce. Y como el conjunto de todas las máquinas de Turing es infinito contable, entonces el conjunto de lenguajes recursivamente enumerables RE también es infinito contable.

c. ¿$\mathscr{L} - RE$ es un conjunto infinito contable?

1. $\mathscr{L}$ es el conjunto de todos los lenguajes posibles sobre un alfabeto Σ.
2. RE es el conjunto de todos los lenguajes recursivamente enumerables.
3. $\mathscr{L} - RE$ es el conjunto de todos los lenguajes que no son recursivamente enumerables.
4. Se sabe que $\mathscr{L}$ es un conjunto infinito **incontable** y que RE es un conjunto infinito **contable**.
5. Como la diferencia entre un conjunto infinito incontable y un conjunto infinito contable es siempre un conjunto infinito incontable, entonces $\mathscr{L} - RE$ es un conjunto infinito **incontable**.

d. Existe algún lenguaje $L \in \mathscr{L}$ tal que L sea infinito no contable?

1. $\mathscr{L}$ es el conjunto de todos los lenguajes posibles sobre un alfabeto Σ.
2. Todo lenguaje de $\mathscr{L}$ es un subconjunto de Σ^*.
3. Σ^* es el conjunto de todas las cadenas posibles formadas por símbolos del alfabeto Σ, y se sabe que es infinito contable.
4. Todo subconjunto de un conjunto infinito contable es también un conjunto infinito contable (o finito).
5. Por lo tanto, no existe ningún lenguaje $L \in \mathscr{L}$ que sea infinito no contable.

6. Sea L un lenguaje definido sobre Σ. Demostrar que:

a. $\overline{L} \notin R \implies L \notin R$

1. Se reescribe la implicación con la contrarrecíproca $(p \to q \Leftrightarrow (\neg q \to \neg p))$:
   i. $\overline{L} \notin R \implies L \notin R$
   ii. $\neg(\overline{L} \in R) \implies \neg(L \in R)$
   iii. $\neg\neg(L \in R) \implies \neg\neg(\overline{L} \in R)$
   iv. $L \in R \implies \overline{L} \in R$

2. Por lo tanto se quiere demostrar que $L \in R \implies \overline{L} \in R$.
3. Como $L \in R$, existe una máquina de Turing M que decide L, es decir:
    i. Si $w \in L$ entonces M se detiene en el estado de aceptación q_A.
    ii. Si $w \notin L$ entonces M se detiene en el estado de rechazo q_R.
4. Se puede construir una máquina de Turing M' que decide $\overline{L}$ a partir de M, simplemente intercambiando todas las transiciones que llevan al estado de aceptación q_A por transiciones que llevan al estado de rechazo q_R, y todas las transiciones que llevan al estado de rechazo q_R por transiciones que llevan al estado de aceptación q_A.
    i. Claramente esta máquina M' también se detiene siempre, por lo que decide $\overline{L}$.
5. Por lo tanto, si $L \in R$ entonces $\overline{L} \in R$.

b. $(L_1 \in RE) \land (L_2 \in RE) \implies L_1 \cap L_2 \in RE$

1. Como $L_1 \in RE$, existe una máquina de Turing M_1 tal que:
    i. Si $w \in L_1$ entonces M_1 se detiene en el estado de aceptación q_A.
    ii. Si $w \notin L_1$ entonces M_1 se detiene en el estado de rechazo q_R o loopea infinitamente.
2. Como $L_2 \in RE$, existe una máquina de Turing M_2 tal que:
    i. Si $w \in L_2$ entonces M_2 se detiene en el estado de aceptación q_A.
    ii. Si $w \notin L_2$ entonces M_2 se detiene en el estado de rechazo q_R o loopea infinitamente.
3. Se puede construir una MT M_3 que reconozca el lenguaje $L_1 \cap L_2$, es decir, las cadenas que pertenecen a ambos lenguajes, de la siguiente manera:
    i. Dada una cadena de entrada w, M_3 simula a M_1 con la entrada w.
    ii. Si M_1 se detiene en el estado de rechazo q_R, entonces M_3 también se detiene en el estado de rechazo q_R.
    iii. Si M_1 se detiene en el estado de aceptación q_A, entonces M_3 simula a M_2 con la misma entrada w.
    iv. Si M_2 se detiene en el estado de rechazo q_R, entonces M_3 se detiene en el estado de rechazo q_R.
    v. Si M_2 se detiene en el estado de aceptación q_A, entonces M_3 se detiene finalmente en el estado de aceptación q_A.
    vi. Si cualquiera de las máquinas loopea, entonces M_3 también loopea.
4. Como M_3 acepta todas las cadenas que pertenecen a $L_1 \cap L_2$ y loopea o rechaza las demás, entonces $L_1 \cap L_2 \in RE$.

c. $(L_1 \in RE) \land (L_2 \in RE) \implies L_1 \cup L_2 \in RE$

1. Como $L_1 \in RE$, existe una máquina de Turing M_1 tal que:
    i. Si $w \in L_1$ entonces M_1 se detiene en el estado de aceptación q_A.
    ii. Si $w \notin L_1$ entonces M_1 se detiene en el estado de rechazo q_R o loopea infinitamente.
2. Como $L_2 \in RE$, existe una máquina de Turing M_2 tal que:

   i. Si $w \in L_2$ entonces M_2 se detiene en el estado de aceptación q_A.
   ii. Si $w \notin L_2$ entonces M_2 se detiene en el estado de rechazo q_R o loopea infinitamente.
3. Se puede construir una MT M_3 que reconozca el lenguaje $L_1 \cup L_2$, es decir, las cadenas que pertenecen a al menos uno de los dos lenguajes, de la siguiente manera:
   i. Dada una cadena de entrada w, M_3 simula a M_1 con la entrada w.
   ii. Si M_1 se detiene en el estado de aceptación q_A, entonces M_3 también se detiene en el estado de aceptación q_A, ya que no necesita chequear M_2, basta con que $w \in L_1$.
   iii. Si M_1 se detiene en el estado de rechazo q_R, entonces M_3 simula a M_2 con la misma entrada w.
   iv. Si M_2 se detiene en el estado de aceptación q_A, entonces M_3 se detiene en el estado de aceptación q_A ya que por más que $w \notin L_1$, basta con que $w \in L_2$.
   v. Si M_2 se detiene en el estado de rechazo q_R, entonces M_3 se detiene en el estado de rechazo q_R, es decir, la cadena w no estaba en ninguno de los dos lenguajes.
   vi. Si cualquiera de las máquinas loopea, entonces M_3 también loopea.
4. Como M_3 acepta todas las cadenas que pertenecen a $L_1 \cup L_2$ y loopea o rechaza las demás, entonces $L_1 \cup L_2 \in RE$.

d. La unión de un número finito de lenguajes recursivamente enumerables es un lenguaje recursivamente enumerable.

La demostración es análoga a la del inciso c), pero en lugar de dos lenguajes se tiene un número finito n de lenguajes recursivamente enumerables $L_1, L_2, \ldots, L_n$.

Lo que se haría es construir una MT M que simule secuencialmente a cada una de las máquinas de Turing $M_1, M_2, \ldots, M_n$ que reconocen a cada uno de los lenguajes $L_1, L_2, \ldots, L_n$ respectivamente. Si al menos una de ellas para en q_A, entonces M también para en q_A. Si todas paran en q_R, entonces M para en q_R. Si alguna loopea, entonces M también loopea.

Por lo tanto la unión de un número finito de lenguajes recursivamente enumerables es un lenguaje recursivamente enumerable.

7. Para los casos a, b y c del punto anterior ¿valen las recíprocas? Justifique.

1. Recíproca del inciso a: $L \notin R \implies \overline{L} \notin R$
   i. Se reescribe la implicación con la contrarrecíproca $(p \to q \Leftrightarrow (\neg q \to \neg p))$:
      a. $L \notin R \implies \overline{L} \notin R$
      b. $\neg(\overline{L} \notin R) \implies \neg(L \notin R)$
      c. $\neg\neg(\overline{L} \in R) \implies \neg\neg(L \in R)$
      d. $\overline{L} \in R \implies L \in R$

e. Por lo tanto se quiere demostrar que $\overline{L} \in R \implies L \in R$.

f. Como $\overline{L} \in R$, existe una máquina de Turing M que decide $\overline{L}$.

a. Si $w \in \overline{L}$ entonces M se detiene en el estado de aceptación q_A.

b. Si $w \notin \overline{L}$ entonces M se detiene en el estado de rechazo q_R.

g. Se puede construir una máquina de Turing M' que decide L a partir de M, simplemente intercambiando todas las transiciones que llevan al estado de aceptación q_A por transiciones que llevan al estado de rechazo q_R, y todas las transiciones que llevan al estado de rechazo q_R por transiciones que llevan al estado de aceptación q_A.

a. Claramente esta máquina M' también se detiene siempre, por lo que decide L.

h. Por lo tanto, si $\overline{L} \in R$ entonces $L \in R$.

2. Recíproca del inciso b: $L_1 \cap L_2 \in RE \implies (L_1 \in RE) \wedge (L_2 \in RE)$

i. Esto no se cumple en general.

ii. Contraejemplo:

a. Sea $L_1 = L_D$ el lenguaje diagonal.

b. Sea $L_2 = \overline{L_D}$ el complemento del lenguaje diagonal.

c. Se sabe que $L_D \notin RE$ y que $\overline{L_D} \in RE$.

d. Se tiene que $L_1 \cap L_2 = L_D \cap \overline{L_D} = \emptyset$.

e. Se sabe que $\emptyset \in RE$.

f. Por lo tanto, como se ha encontrado un contraejemplo donde $L_1 \in RE$, $L_2 \notin RE$ y $L_1 \cap L_2 \in RE$, entonces la recíproca no se cumple en general.

3. Recíproca del inciso c: $L_1 \cup L_2 \in RE \implies (L_1 \in RE) \wedge (L_2 \in RE)$

i. Esto no se cumple en general.

ii. Contraejemplo:

a. Sea $L_1 = L_D$ el lenguaje diagonal.

b. Sea $L_2 = \overline{L_D}$ el complemento del lenguaje diagonal.

c. Se sabe que $L_D \notin RE$ y que $\overline{L_D} \in RE$.

d. Se tiene que $L_1 \cup L_2 = L_D \cup \overline{L_D} = \Sigma^*$.

e. Se sabe que $\Sigma^* \in RE$.

f. Por lo tanto, como se ha encontrado un contraejemplo donde $L_1 \in RE$, $L_2 \notin RE$ y $L_1 \cup L_2 \in RE$, entonces la recíproca no se cumple en general.

8. Si L es un subconjunto de un lenguaje recursivamente enumerable, ¿Puede afirmarse entonces que L es recursivamente enumerable? Justifique.

1. Sea $L' \in RE$.
2. Sea $L \subseteq L'$.

3. $L \in RE$?
4. No es posible afirmar que L es recursivamente enumerable, ya que existen subconjuntos de lenguajes recursivamente enumerables que no son recursivamente enumerables:
5. Contraejemplo:
    i. $\Sigma^* \in RE$
    ii. $L_D \subseteq \Sigma^*$
    iii. $L_D \notin RE$

9. Dado L_1, un lenguaje recursivo cualquiera

$$L_2 = \{\langle M \rangle \mid L(M) = \overline{L_1}\}$$

$$L_3 = \{\langle M \rangle \mid L(M) = \overline{L_1} \text{ y } M \text{ siempre se detiene}\}$$

Determine si $(L_2 - L_3) = \emptyset$. Justifique su respuesta.

1. $L_1 \in R$.
2. Por lo tanto $\overline{L_1} \in R$.
3. L_2 es el lenguaje cuyos elementos son las codificaciones de máquinas de Turing tal que el lenguaje que reconoce cada máquina es el complemento de L_1.
4. L_3 es el lenguaje cuyos elementos son las codificaciones de máquinas de Turing tal que el lenguaje que reconoce cada máquina es el complemento de L_1 y además la máquina siempre se detiene.
5. La diferencia $L_2 - L_3$ son las codificaciones de máquinas de Turing tal que el lenguaje que reconoce cada máquina es el complemento de L_1 pero la máquina no siempre se detiene.
6. Este conjunto NO es vacío porque existen máquinas de Turing que reconocen a $\overline{L_1}$ pero no siempre se detienen.
7. Es decir, para cualquier lenguaje recursivo, siempre se puede construir una MT que reconozca las cadenas del lenguaje pero loopee en las cadenas que no pertenecen al lenguaje, en lugar de detenerse y rechazarlas. Esto se logra fácilmente tomando cualquier MT que decida el lenguaje (es decir que nunca loopea), y cambiando sus transiciones hacia q_R por transiciones hacia un estado de loopeo q_L.

Por lo tanto la afirmación es falsa, los lenguajes L_2 y L_3 son distintos.

10. Sean los lenguajes $L = \{\langle M \rangle \mid M \text{ siempre se detiene}\}$ y $L_R = \{\langle M \rangle \mid L(M) \in R\}$.

¿Cuál es la afirmación correcta?

Para este ejercicio, es clave recordar que $R \subseteq RE$.

a. $L \subset L_R$

1. L es el lenguaje cuyos elementos son las codificaciones de máquinas de Turing que siempre se detienen.
2. L_R es el lenguaje cuyos elementos son las codificaciones de máquinas de Turing cuyo lenguaje es decidible.
3. Todo elemento de L está también en L_R, porque si una máquina de Turing siempre se detiene, entonces el lenguaje que reconoce es decidible.

Por lo tanto la afirmación es verdadera. ✅

b. $L_R \subset L$

1. L_R es el lenguaje cuyos elementos son las codificaciones de máquinas de Turing cuyo lenguaje es decidible.
2. L es el lenguaje cuyos elementos son las codificaciones de máquinas de Turing que siempre se detienen.
3. En general, no todo elemento de L_R está en L, porque existen máquinas de Turing cuyo lenguaje es decidible pero que no siempre se detienen (justificado en el punto 9).

Por lo tanto la afirmación es falsa. ❌

c. $L = L_R$

1. Como se tiene que $L \subset L_R$ y $L_R \not\subset L$, entonces trivialmente $L \neq L_R$.

Por lo tanto la afirmación es falsa. ❌

11. Encuentre una justificación para cada una de las siguientes afirmaciones:

a. $\emptyset \in RE$

1. Como $\emptyset \subseteq \Sigma^*$, $\emptyset$ es un lenguaje.
2. $\emptyset \in R$ porque existe una máquina de Turing que rechaza todas las cadenas posibles del lenguaje y además siempre termina, es decir, $L(M) = \emptyset$:
   i. $\delta(q_0, B) = (q_R, B, S)$
   ii. $\delta(q_0, x) = (q_R, x, S) \quad \forall x \in \Sigma$

3. Como $R \subseteq RE$, entonces $\emptyset \in RE$.

b. Si L es un lenguaje formado por una sola palabra, entonces $L \in R$

1. $L = \{w\}$, con $w \in \Sigma^*$.
2. $L \in R$ si existe una máquina de Turing que acepta la palabra w, rechaza todas las demás palabras, y además siempre se detiene.
3. Toda palabra w es una secuencia finita de símbolos de Σ, es decir $w = s_1, s_2, s_3, \ldots, s_n$.
4. Se puede construir una MT M que pase por una cantidad finita de estados para aceptar la palabra w de la siguiente manera:
    i. M comienza en el estado inicial q_0.
    ii. La máquina itera sobre cada símbolo s_i de la palabra w uno por uno asegurándose de que la cadena de entrada coincida exactamente con w.
    iii. Si en algún punto el símbolo de la cadena de entrada no coincide con el símbolo esperado s_i, la máquina para en q_R.
    iv. Luego de leer el último símbolo s_n, se lee un símbolo en blanco B que indica el final de la cadena, y la máquina se detiene en el estado de aceptación q_A.

c. Si L es un lenguaje finito, entonces $L \in R$

1. Sea $L = \{w_1, w_2, \ldots, w_n\}$ con $w_i \in \Sigma^*$.
2. $L \in R$ si existe una máquina de Turing que acepta cada palabra w_i, rechaza todas las demás palabras, y además siempre se detiene.
3. Toda palabra w es una secuencia finita de símbolos de Σ, es decir $w = s_1, s_2, s_3, \ldots, s_n$.
4. Como L es finito, existe una cantidad finita de palabras w_i.
5. Se puede construir una MT M que pase por una cantidad finita de estados para aceptar cada palabra w_i de la siguiente manera:
    i. M comienza en el estado inicial q_0.
    ii. Para cada palabra w_i, la máquina itera sobre cada símbolo s_j de la palabra uno por uno asegurándose de que la cadena de entrada coincida exactamente con w_i.
    iii. Si en algún punto el símbolo de la cadena de entrada no coincide con el símbolo esperado s_j, la máquina pasa a chequear la siguiente palabra w_{i+1}, para lo cual debe volver al inicio de la cinta.
    iv. Luego de leer el último símbolo s_n de alguna palabra w_i, se lee un símbolo en blanco B que indica el final de la cadena, y la máquina se detiene en el estado de aceptación q_A.
    v. Si ninguna de las palabras coincide, la máquina se detiene en el estado de rechazo q_R.

12. Demuestre que si el $\text{Halting Problem (HP)}$ es un lenguaje recursivo entonces podría construirse una máquina de Turing que acepte el lenguaje universal L_u, y que se detenga para todo $w \in \Sigma^*$ ¿Qué puede decir entonces sobre la recursividad de HP?

$$HP = \{(\langle M \rangle, w) \mid \text{M se detiene con input w}\}$$

$$L_u = \{(\langle M \rangle, w) \mid \text{M acepta w}\}$$

Tenemos una proposición del estilo $(HP \in R \rightarrow L_u \in R)$. Para probar que esto es falso, se puede asumir que $HP \in R$ es verdadero y luego llegar a que $L_u \in R$ es falso.

Por premisa, se asume que HP es un lenguaje decidible $(HP \in R)$. Por lo tanto, existe una máquina de Turing M_{HP} que decide HP, es decir, que hace lo siguiente:

- **Recibe** un input $(\langle M \rangle, w)$, es decir, un código binario de una MT M y una cadena w que es la entrada de M.
- **Acepta** si M se detiene con input w.
- **Rechaza** si M no se detiene (loopea) con input w.
- Como M_{HP} es decidible, **siempre se detiene (nunca loopea)**.

Luego se puede construir una máquina de Turing M_{L_u} que decide L_u utilizando a M_{HP} de la siguiente manera:

- Recibe un input $(\langle M \rangle, w)$.
- Ejecuta a M_{HP} con el input $(\langle M \rangle, w)$.
  - Como M_{HP} siempre se detiene, M_{L_u} espera a que M_{HP} termine.
- Si M_{HP} **rechaza**:
  - Esto equivale a que M no se detiene con input w.
  - Como M no se detiene, entonces M no puede aceptar w.
  - Por lo tanto $(\langle M \rangle, w) \notin L_u$.
  - M_{L_u} rechaza, y luego se detiene.
- Si M_{HP} **acepta**:
  - Esto equivale a que M se detiene con input w.

- Como sabemos que M se detiene con input w, entonces podemos simular su ejecución de forma segura sin riesgo de loopear.
- M_{L_u} simula a M con el input w:
- Si M acepta w, entonces M_{L_u} acepta.
- Si M rechaza w, entonces M_{L_u} rechaza.

Como M_{HP} siempre se detiene, y la simulación de M con input w también se detiene (porque en este caso M se detiene), entonces M_{L_u} siempre se detiene.

Por lo tanto, se ha construido una máquina de Turing M_{L_u} que decide L_u, lo que implica que $L_u \in R$.

Sin embargo, se **sabe por teorema que L_u no es un lenguaje decidible** $(L_u \notin R)$. Esto lleva a una contradicción, lo que implica que la suposición inicial $(HP \in R)$ es falsa.

Se puede decir entonces que el Halting Problem HP **no es un lenguaje recursivo** $(HP \notin R)$.

13. Demuestre que $L_{NV} \in RE$

$$L_{NV} = \{\langle M \rangle \mid L(M) \neq \emptyset\}$$

$L_{NV} \in RE$ si existe una máquina de Turing M tal que $L(M) = L_{NV}$. Es decir, si existe una MT M que acepte todas las cadenas que son códigos binarios de máquinas de Turing que reconocen al menos una cadena, y rechaze todas las demás cadenas ya sea loopeando o yendo al estado de rechazo q_R.

Esta MT existe y se puede construir de la siguiente manera:

- Recibe un input $\langle M' \rangle$, que es el código binario de una máquina de Turing M'.
- Si $\langle M' \rangle$ no es un código válido de una máquina de Turing:
  - Entonces M' no reconoce ninguna cadena, es decir, $L(M') = \emptyset$.
  - M rechaza el input $\langle M' \rangle$ y se detiene.
- Si $\langle M' \rangle$ es un código válido de una máquina de Turing:
  - M simula la ejecución de M' con todas las posibles cadenas de entrada $w \in \Sigma^*$, una por una.
  - Si durante la simulación, M' acepta alguna cadena w:
    - Entonces $L(M') \neq \emptyset$.
    - M acepta el input $\langle M' \rangle$ y se detiene.
  - Si durante la simulación, M' no acepta ninguna cadena w:
    - Entonces $L(M') = \emptyset$.
    - M rechaza el input $\langle M' \rangle$ y se detiene.
    - O bien, M puede loopear en este caso, si M' también loopea.

Por lo tanto, se ha construido una máquina de Turing M que acepta todas las cadenas que son códigos binarios de máquinas de Turing que reconocen al menos una cadena, y rechaza todas las demás cadenas. Esto implica que $L_{NV} \in RE$.