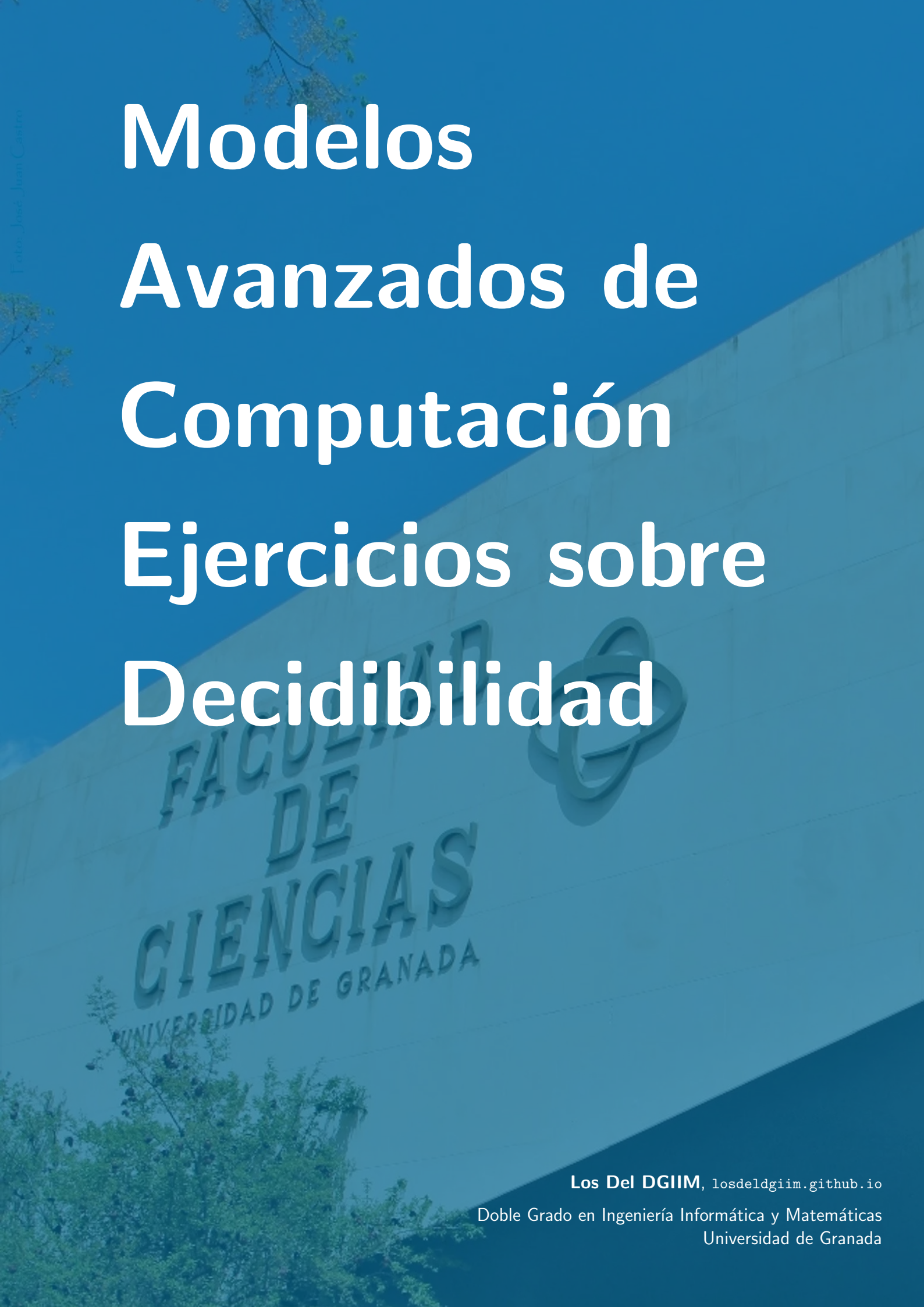
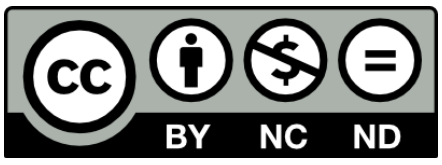
Modelos Avanzados de Computación Ejercicios sobre Decidibilidad



Los Del DGIIM, losdeldgiim.github.io

Doble Grado en Ingeniería Informática y Matemáticas
Universidad de Granada



Modelos Avanzados de Computación Ejercicios sobre Decidibilidad

Los Del DGIIM, losdeldgiim.github.io

José Manuel Sánchez Varbas

Granada, 2026

Asignatura Modelos Avanzados de Computación.

Curso Académico 25-26.

Grado Grado en Ingeniería Informática.

Grupo 1.

Profesor Serafín Moral Callejón.

Descripción Ejercicios sobre Decidibilidad de la Relación 2.

Fecha Abril 2026.

Ejercicio 8. Demostrar que el problema de la parada, determinar el conjunto de parejas (M, w) tales que la MT M para cuando tiene a w como entrada, es semidecidible pero no decidible.

Ejercicio 9. (MANUAL) Demostrar que determinar si una MT acepta al menos un palíndromo no es decidible pero sí semidecidible.

Ejercicio 10. Supongamos que tenemos MT con dos tipos de estados: *campana* y *silbato*. Determinar que saber si una MT entrará alguna vez en un estado *silbato* es indecidible. ¿Es semidecidible?

Ejercicio 11. (MANUAL) Demostrar que es indecidible (no recursivo) saber si una MT termina escribiendo un 1 cuando comienza con una cinta completamente en blanco.

Ejercicio 12. Determinar si los siguientes problemas son decidibles, semidecidibles o no semidecidibles:

a) Determinar si el lenguaje de una MT contiene, al menos, dos palabras distintas.

b) Determinar si el lenguaje reconocido por una MT es finito.

c) (MANUAL) Determinar si el lenguaje reconocido por una MT es infinito.

d) Determinar si el lenguaje de una MT es independiente del contexto.

Ejercicio 13. Sea L el lenguaje formado por pares de códigos de MT más un entero (M_1, M_2, k) tales que $L(M_1) \cap L(M_2)$ contiene, al menos, k palabras. Demostrar que L es semidecidible pero no decidible.

Ejercicio 14. Demostrar que los siguientes lenguajes son recursivos:

a) El conjunto de las MT M tales que al comenzar con la cinta en blanco, en algún momento escribirán un símbolo no blanco en la cinta.

b) El conjunto de las MT que nunca se mueven a la izquierda.

c) El conjunto de los pares (M, w) tales que M, al actuar sobre la entrada w, nunca lee una casilla de la cinta más de una vez.

Ejercicio 15. Indicar si los siguientes problemas son decidibles, semidecidibles o no semidecidibles:

a) (MANUAL) Determinar si una MT se para para cualquier entrada.

b) Determinar si una MT no se para para ninguna entrada.

c) Determinar si una MT se para, al menos, para una entrada.

d) Determinar si una MT no se para, al menos, para una entrada.

Ejercicio 16. Supongamos que tenemos 4 lenguajes L_1, L_2, L_3, L_4 de tal manera que existe una reducción de L_1 a L_2, de L_2 a L_3 y de L_4 a L_3. Para cada una de las siguientes afirmaciones, indicar si para todos los casos son CIERTAS, para todos los casos son FALSAS o si son POSIBLES (en algunos casos son ciertas y, en otros, son falsas).

a) L_1 es recursivamente enumerable pero no recursivo, y L_3 es recursivo.

b) L_1 no es recursivo y L_4 no es recursivamente enumerable.

c) El complementario de L_1 no es recursivamente enumerable, pero el complementario de L_2 es recursivamente enumerable.

d) El complementario de L_2 no es recursivo, pero el complementario de L_3 es recursivo.

e) Si L_1 es recursivo, entonces el complementario de L_2 es recursivo.

f) Si L_3 es recursivo, entonces el complementario de L_4 es recursivo.

g) Si L_3 es recursivamente enumerable, entonces la unión de L_2 y L_4 es recursivamente enumerable.

h) Si L_3 es recursivamente enumerable, entonces la intesección de L_2 y L_4 es recursivamente enumerable.

Ejercicio 17. Sea el problema de determinar si una máquina de Turing acepta a lo más 100 palabras. Determinar si es decidible, semidecidible o no semidecidible. Justificar la respuesta.

Ejercicio 18. Determinar cuáles de los siguientes problemas son decidibles, semidecidibles o no semidecidibles:

a) Saber si una MT para una entrada 0011 no va a usar más de 10 casillas de la cinta.

b) Dadas dos MT saber si aceptan el mismo lenguaje

Ejercicio 19. Determinar cuáles de los siguientes problemas son decidibles, semidecidibles o no semidecidibles:

a) Dada una MT M y una palabra u, saber si la MT acepta la palabra u en un número de pasos menor o igual a $|u|$.

b) Dada una MT determinar si no acepta la palabra vacía.

Ejercicio 20. Determinar cuáles de los siguientes problemas son decidibles, semidecidibles o no semidecidibles (se supone que las MTs tienen a $\{0, 1\}$ como alfabeto de entrada):

a) Dadas dos máquinas de Turing, M_1 y M_2, determinar si existe, al menos, una palabra aceptada simultáneamente por ambas máquinas.

b) (MANUAL) Dada una MT, determinar si para toda palabra de entrada no realiza más de 5 movimientos.

c) Determinar si una MT no es determinista.

d) (MANUAL) Dada una MT, determinar si ninguna de las palabras que acepta es un palíndromo.

Justifica las respuestas.

Ejercicio 21. Se considera el siguiente problema: dadas dos máquinas de Turing, determinar si aceptan el mismo lenguaje. Razonar si este problema es decidible, semidecidible o no semidecidible.

Ejercicio 22. Determinar, justificando las respuestas, cuáles de los siguientes problemas son decidibles, semidecidibles o no semidecidibles:

a) Dada una máquina de Turing y una palabra de entrada determinar si visita menos de 10 casillas distintas de la cinta de lectura.

b) Dadas dos Máquinas de Turing, determinar si el conjunto de las palabras aceptadas a la vez por ambas máquinas de Turing es finito.

Ejercicio 23. Indica cuáles de los siguientes problemas son decidibles, semidecidibles o no semidecidibles:

a) Determinar si el lenguaje aceptado por una MT es regular.

b) Dadas dos MTs determinar si hay una palabra que es aceptada por las dos MT y además es un palíndromo.

Ejercicio 24. Indica cuáles de los siguientes problemas son decidibles, semidecidibles o no semidecidibles:

a) Dada una MT determinar si todos sus estados son accesibles (se puede llegar a ellos para un cálculo para alguna entrada).

b) Dada una MT determinar si su número de estados es menor o igual a 5.

Ejercicio 25. Demuestra que el Problema de la Correspondencia de Post con un alfabeto A que tiene un sólo elemento es decidible.

Ejercicio 8. Demostrar que el problema de la parada, determinar el conjunto de parejas (M, w) tales que la MT M para cuando tiene a w como entrada, es semidecidible pero no decidible.

Definimos PARADA$(M, w) = \{\langle M, w\rangle : M \text{ para con entrada } w\}$. Para ver que es semidecidible, la idea es construir un simulador de MT M_p, que lea la codificación $\langle M, w\rangle$, simule la MT M sobre la entrada w y si en algún momento de la simulación M para, M_p acepta, de manera que M_p acepte exactamente las entradas $\langle M, w\rangle$ tal que M para con w.

Algoritmo 1 Máquina de Turing M_p

Entrada: Una palabra $v \in \{0, 1\}^*$
Comprobar si v es de la forma $\langle M, w\rangle$
Si v no es una codificación válida **entonces**
    Rechazar
Fin Si
Simular paso a paso la ejecución de M sobre la entrada w
Si en algún momento la simulación de M para **entonces**
    Aceptar
Fin Si

Esta MT M_p acepta exactamente los casos positivos de las palabras del lenguaje L_p asociado:

- Si $\langle M, w\rangle \in$ PARADA, la simulación de M sobre w termina en un número finito de pasos, y M_p aceptará.
- Si $\langle M, w\rangle \notin$ PARADA, entonces M cicla indefinidamente sobre w, y la simulación hace continúa también indefinidamente, por tanto M_p no aceptará.

Por lo tanto, PARADA es semidecidible.

Para ver que no es decidible, la idea será hacer una reducción de universal a parada, que sabemos que es semidecidible, pero no decidible por teoría. Consideramos los problemas:

$$\begin{cases} P1 = & \text{UNIVERSAL : dada una MT } M \text{ y una palabra } w, \text{ determinar si } M \text{ acepta } w \\ P2 = & \text{PARADA : dada una MT } N \text{ y una palabra } u, \text{ determinar si } N \text{ para con } u \end{cases}$$

Construimos un algoritmo F que transforma cada instancia (M, w) de UNIVERSAL en una instancia (N, w) de PARADA, donde $N = F(M)$ es la MT siguiente:

Algoritmo 2 Máquina de Turing $N = F(M)$

Entrada: Una palabra $u \in \{0, 1\}^*$
1: Simular la ejecución de M sobre la entrada u
2: Si M acepta u, entonces N para
3: En otro caso, N no para (cicla).

Esta construcción es algorítmica, ya que a partir de M puede construirse una MT N que simule a M sobre su misma entrada u y que solo pare cuando la simulación llegue a aceptación.

Veamos la equivalencia:

- Si (M, w) es un caso positivo de UNIVERSAL, es decir, si M acepta w, entonces al ejecutar N sobre w la simulación llega a aceptación y N para. Por tanto, $(N, w) \in$ PARADA.
- Si (M, w) es un caso negativo de UNIVERSAL, es decir, si M no acepta w, entonces la máquina N, sobre w, no para. Por tanto, $(N, w) \notin$ PARADA. Así

$$(M, w) \in \text{UNIVERSAL} \iff (N, w) \in \text{PARADA}$$

Por tanto, tenemos que UNIVERSAL$(M, w) \propto$ PARADA(N, w). Si PARADA fuera decidible, tendríamos que UNIVERSAL también lo sería porque UNIVERSAL se reduce a PARADA. Sin embargo, sabemos que esto no es cierto, por lo que PARADA no es decidible.

Ejercicio 9. (MANUAL) Demostrar que determinar si una MT acepta al menos un palíndromo no es decidible pero sí semidecidible.

Dada una MT M, definimos $\text{PAL}(M) = \{\langle M\rangle : \exists w \in \{0,1\}^* \text{ palíndromo tal que } M \text{ acepta } w\}$.

Para la no decidibilidad, vemos que la propiedad "aceptar al menos un palíndromo" depende solo del lenguaje aceptado por la máquina, $L(M)$, y no de la máquina concreta, es decir, es una propiedad de los lenguajes r.e.. Además, es una propiedad no trivial, ya que el lenguaje $\{11\}$ la cumple, pero $\{10\}$ no. Así, por el Teorema de Rice, PAL no es decidible.

Para la semidecidibilidad, consideramos una MTND M_{PAL} que funciona como sigue:

Algoritmo 3 MTND M_{PAL}

Entrada: Una codificación $\langle M\rangle$ de una MT M
Selecciona de forma no determinista una palabra $w \in \{0,1\}^*$.
Comprueba si w es un palíndromo.
Si w es palíndromo, simula M con entrada w
Si M acepta w, entonces M_{PAL} acepta.

Tenemos entonces que, si M acepta al menos un palíndromo w_0, existe una rama de cómputo de M_{PAL} que precisamente toma w_0, comprueba que es palíndromo y acepta. Sin embargo, si M no acepta ningún palíndromo, entonces ninguna rama puede aceptar. Por lo tanto, M_{PAL} acepta exactamente las codificaciones $\langle M\rangle$ tales que M acepta al menos un palíndromo, de donde $\text{PAL}(M)$ es semidecidible.

Ejercicio 10. Supongamos que tenemos MT con dos tipos de estados: *campana* y *silbato*. Determinar que saber si una MT entrará alguna vez en un estado *silbato* es indecidible. ¿Es semidecidible?

Definimos

$$\text{SILBATO}(M) = \{\langle M\rangle : \exists w \in \{0,1\}^* : M \text{ entra alguna vez en un estado silbato con entrada } w\}$$

Para ver que no es decidible, dada una MT M, como la propiedad "entrar en un estado silbato" no es una propiedad del lenguaje $L(M)$, tampoco es una propiedad de los lenguajes r.e., y no puede aplicarse el Teorema de Rice.

Haremos entonces una reducción desde un problema no decidible de tal manera que sea sencilla. Por ejemplo, PARADA. Consideramos entonces el problema

$$\text{PARADA}(M, w) = \{\langle M, w\rangle : M \text{ para con entrada } w\}$$

Vamos a construir un proceso algorítmico F para pasar de una instancia de PARADA a una instancia de SIL. Sea (M, w) una instancia de PARADA, construimos las siguiente máquina de Turing $F(M, w)$ (que es una instancia de SIL)

Algoritmo 4 MT $F(M, w)$

Entrada: $v \in \{0,1\}^*$
Borramos v de la cinta.
Copiamos $\langle M\rangle 111w$ en la cinta.
Simulamos M con entrada w.
    Si M para **entonces**
        $F(M, w)$ entra en un estado *silbato*.
    Si M no para **entonces**
        $F(M, w)$ cicla y nunca entra en un estado *silbato*.

Vemos que la máquina $F(M, w)$ se construye de tal manera que todos los estados usados para las tareas previas a que M termine sean de tipo *campana*. Si M termina, entonces añadimos un nuevo estado especial q_s, de tipo *silbato*, al que se llega únicamente cuando la simulación termina. Por lo tanto

$$\langle M, w\rangle \in \text{PARADA} \iff \langle F(M, w)\rangle \in \text{SILBATO}$$

Puesto que:

1. Si M para con w (caso positivo de PARADA), entonces $F(M, w)$ termina la simulación y entra en q_s (caso positivo de SILBATO).
2. Si M no para con w (caso negativo de PARADA), entonces $F(M, w)$ ciclará y nunca entrará en estado silbato (caso negativo de SILBATO).

Por lo tanto, PARADA $\propto$ SILBATO. Si SILBATO fuera decidible, también lo sería PARADA, cosa que sería una contradicción. Así, SILBATO no es decidible.

La semidecidibilidad puede comprobarse con una MTND que seleccione de forma no determinista una entrada u y simule M con dicha entrada y acepte en caso de que durante la simulación M entre en un estado *silbato*. Más en detalle:

Algoritmo 5 MTND $M_{SILBATO}$

Entrada: Una codificación $\langle M \rangle$ de una MT M
Selecciona de forma no determinista una palabra $u \in \{0,1\}^*$.
Simulamos M con entrada u.
Si en algún momento de la simulación M entra en un estado *silbato*, entonces $M_{SILBATO}$ acepta.

Ejercicio 11. (MANUAL) Demostrar que es indecidible (no recursivo) saber si una MT termina escribiendo un 1 cuando comienza con una cinta completamente en blanco.

Hecho en el manual. Para la no decidiblidad hay que hacer una reducción desde UNIVERSAL al problema, y para la semidecidibilidad basta ejecutar un bucle $i = 1, \ldots, \infty$ tal que se simule la MT con entrada ε en i pasos, y si en alguno de esos i pasos escribe un 1 acepta la entrada.

Ejercicio 12. Determinar si los siguientes problemas son decidibles, semidecidibles o no semidecidibles:

a) Determinar si el lenguaje de una MT contiene, al menos, dos palabras distintas.

Definimos DOSPal$(M) = \{\langle M\rangle : \exists u, v \in L(M) \text{ con } u \neq v\}$. Vemos que no es decidible, ya que la propiedad "$L(M)$ contiene al menos dos palabras distintas" depende solo del lenguaje $L(M)$, no de la MT M concreta, luego es una propiedad de los lenguajes r.e.. Además, es no trivial, ya que $\{0\}$ no cumple la propiedad, pero $\{0, 1\}$ sí la cumple. Por el Teorema de Rice, DOSPal(M) es no decidible. Veamos que es semidecidible. Para ello, consideramos una MTND M_{DOSPal} que funciona como sigue:

Algoritmo 6 MTND M_{DOSPAL}

Entrada: Una codificación $\langle M\rangle$ de una MT M
Selecciona de forma no determinista dos palabras $u, v \in \{0, 1\}^*$.
Si $u = v$, entonces M_{DOSPal} rechaza.
Simulamos M con entrada u y con entrada v.
Si ambas simulaciones aceptan, entonces M_{DOSPal} acepta.

Tenemos entonces que, si M acepta al menos dos palabras distintas u_0, v_0, existe una rama de cómputo de M_{DOSPal} que precisamente toma u_0 y v_0, simula $M(u_0)$ y $M(v_0)$, y como ambas aceptan, esa rama acepta. Sin embargo, si M no acepta dos palabras distintas, entonces para cualquier par distinto (u, v), al menos una de las dos simulaciones no aceptará y consecuentemente esa rama no aceptará. Por lo tanto, M_{DOSPal} acepta exactamente las codificaciones $\langle M\rangle$ tales que M acepta al menos dos palabras distintas, de donde DOSPal(M) es semidecidible.

b) Determinar si el lenguaje reconocido por una MT es finito.

Definimos FINITO$(M) = \{\langle M\rangle : L(M)$ es finito$\}$. Vemos que no es decidible, ya que la propiedad "$L(M)$ es finito" depende solo del lenguaje $L(M)$, no de la MT M concreta, luego es una propiedad de los lenguajes r.e.. Además, es no trivial, ya que cumple la propiedad (una MT sin estado final), pero $\{0,1\}^*$ no la cumple (una MT con el estado inicial también final que acepte todas las palabras). Por el Teorema de Rice, FINITO(M) es no decidible.

Para lo no semidecidibilidad se puede hacer una reducción desde

$$\text{DIAGONAL}(M) = \{\langle M\rangle : M \text{ no acepta } \langle M\rangle\}$$

(que sabemos por teoría que no es semidecidible), igual que en el apartado c), pero permutando los casos en que se rechaza por lo que se acepta. El proceso algorítmico F pasará de una instancia de DIAGONAL a una instancia de FINITO. Sea entonces M una instancia de DIAGONAL, y construimos la siguiente máquina de Turing $F(M)$:

Algoritmo 7 MT $F(M)$

Entrada: Una palabra $v \in \{0,1\}^*$.
Si v no es de la forma 0^n con $n > 0$ **entonces**
    $F(M)$ rechaza.
Si $v = 0^n$ para cierto $n > 0$ **entonces**
    Simulamos M con entrada $\langle M\rangle$ durante n pasos.
    Si M acepta $\langle M\rangle$ **entonces**
        $F(M)$ acepta.
    Si no entonces
        $F(M)$ rechaza.

c) (MANUAL) Determinar si el lenguaje reconocido por una MT es infinito.

Hecho en el manual.

d) Determinar si el lenguaje de una MT es independiente del contexto.

Definimos INDCONT$(M) = \{\langle M\rangle : L(M)$ es independiente del contexto$\}$. Al menos es no decidible por el Teorema de Rice, ya que, para una MT M, la propiedad "ser $L(M)$ independiente del contexto" es una propiedad que depende solo de $L(M)$, no de M, luego es una propiedad no trivial de los lenguajes r.e. (no trivial porque sabemos que existe una MT M_1 tal que $L(M_1)$ es independiente del contexto y M_2 tal que $L(M_2)$ no es independiente del contexto (por Modelos de Computación)). De hecho, podemos ver que es no semidecidible por medio de una reducción desde

$$\text{DIAGONAL}(M) = \{\langle M\rangle : M \text{ no acepta } \langle M\rangle\}$$

Sea M'' una MT fija tal que

$$L(M'') = \{0^n 1^n 0^n : n \geqslant 0\},$$

que no es un lenguaje independiente del contexto (puede comprobarse con el Lema de bombeo). Dada una MT M, construimos una MT M' con el siguiente comportamiento:

Algoritmo 8 MT M'

Entrada: Una palabra $x \in \{0,1\}^*$
Simulamos M con entrada $\langle M \rangle$.
Si M acepta $\langle M \rangle$ **entonces**
    Simulamos M'' con entrada x.
    Si M'' acepta x **entonces**
        M' acepta.
    Si no
        M' rechaza.
    Fin Si
Fin Si

Tenemos entonces que:

- Si $\langle M \rangle \in$ DIAGONAL, entonces M no acepta $\langle M \rangle$. Por tanto, al simular M con entrada $\langle M \rangle$, pueden pasar dos cosas: o bien M rechaza, o bien M cicla. En ambos casos, M' no acepta ninguna palabra. Luego

  $$L(M') = \emptyset.$$

  Como $\emptyset$ es independiente del contexto, se tiene que

  $$\langle M' \rangle \in \text{INDCONT}.$$

- Si $\langle M \rangle \notin$ DIAGONAL, entonces M acepta $\langle M \rangle$. Por construcción, en ese caso M' pasa a comportarse como M'' sobre su entrada x. Por tanto,

  $$L(M') = L(M'') = \{0^n 1^n 0^n : n \geqslant 0\}.$$

  Como este lenguaje no es independiente del contexto, se tiene que

  $$\langle M' \rangle \notin \text{INDCONT}.$$

Así,

$$\langle M \rangle \in \text{DIAGONAL} \iff \langle M' \rangle \in \text{INDCONT}.$$

Por tanto,

$$\text{DIAGONAL} \propto \text{INDCONT}.$$

Como DIAGONAL es no semidecidible, INDCONT es no semidecidible.

Ejercicio 13. Sea L el lenguaje formado por pares de códigos de MT más un entero (M_1, M_2, k) tales que $L(M_1) \cap L(M_2)$ contiene, al menos, k palabras. Demostrar que L es semidecidible pero no decidible.

Vemos que

$$L = \{(M_1, M_2, k) : |L(M_1) \cap L(M_2)| \geqslant k\}$$

Para ver que es semidecidible, construimos la siguiente $MTND$:

Algoritmo 9 MTND M_L

Entrada: Una codificación $\langle M_1, M_2, k\rangle$, donde M_1 y M_2 son MT y $k \in \mathbb{Z}$.
Si $k = 0$ **entonces**
    M_L acepta.
Fin Si
Selecciona de forma no determinista k palabras $u_1, \ldots, u_k \in \{0,1\}^*$.
Si $\exists i, j \in \{1, \ldots, k\} : i \neq j \quad \wedge \quad u_i = u_j$ **entonces**
    M_L rechaza.
Fin Si
Para $i = 1, \ldots, k$ **hacer**
    Simulamos M_1 con entrada u_i.
    Simulamos M_2 con entrada u_i.
    Si alguna de las dos simulaciones rechaza **entonces**
        M_L rechaza.
    Fin Si
Fin Para
Si todas las simulaciones aceptan **entonces**
    M_L acepta.
Fin Si

Vemos que funciona, ya que $(M_1, M_2, k) \in L$ si y solo si existen k palabras distintas $u_1, \ldots, u_k$ tales que $u_i \in L(M_1) \cap L(M_2) \quad \forall i = 1, \ldots, k$. Por lo tanto, M_L semidecide L, luego L es semidecidible.

Para ver que L es no decidible, hacemos una reducción desde

$$\text{C-VACIO}(M) = \{\langle M\rangle : L(M) \neq \emptyset\}$$

que sabemos que no es decidible por teoría.

Dada una MT M, construimos la instancia $(M, U, 1)$, con U es una MT que acepta todas las palabras, es decir, $L(U) = \{0,1\}^*$ (por ejemplo, U tiene el estado inicial también final). Entonces

$$(M, U, 1) \in L \iff |L(M) \cap L(U)| \geqslant 1 \iff L(M) \neq \emptyset$$

donde en la última equivalencia se ha usado que $L(M) \cap L(U) = L(M)$, y $|L(M)| \geqslant 1 \iff L(M) \neq \emptyset$. Tenemos entonces que

$$M \in \text{C-VACIO} \iff (M, U, 1) \in L$$

Así, C-VACIO $\propto L$, y si L fuera decidible también lo sería C-VACIO, cosa que es falsa, luego L no es decidible.

Ejercicio 14. Demostrar que los siguientes lenguajes son recursivos:

a) El conjunto de las MT M tales que al comenzar con la cinta en blanco, en algún momento escribirán un símbolo no blanco en la cinta.

Sea

$$L_a = \{\langle M\rangle : M \text{ empezando con la cinta en blanco escribe alguna vez un símbolo distinto de } \#\}$$

Teniendo en cuenta que, si la cinta comienza entera en blanco (entonces M se ejecuta sobre la palabra vacía), seguirá estándolo mientras no se escriba ningún símbolo no blanco. Por tanto, en cada paso M lee siempre #. El comportamiento de M se puede determinar entonces por transiciones de la forma $\delta(q, \#)$. Como una MT tiene un conjunto finito de estados y transiciones de la forma $\delta(q, b) = (p, c, M)$, con $M \in \{I, S, D\}$, puede decidirse algorítmicamente en una cantidad finita de pasos, y por tanto en tiempo finito.

Algoritmo 10 MT M_{L_a}

Entrada: Una codificación $\langle M\rangle$ de una MT M
$q \leftarrow q_0$
$V \leftarrow \emptyset$
Mientras $q \notin V$ **hacer**
  Añadir q a V
  Si $\delta(q, \#)$ no está definida **entonces**
    Rechazar
  Fin Si
  Sea $\delta(q, \#) = (p, c, m)$
  Si $c \neq \#$ **entonces**
    Aceptar
  Fin Si
  $q \leftarrow p$
Fin Mientras
Rechazar

Si aparece una transición que escribe un símbolo $c \neq \#$, M escribe un símbolo no blanco, luego aceptamos. En otro caso, la máquina nunca escribirá un símbolo no blanco. Como los estados son finitos, el algoritmo siempre termina, y decide correctamente.

b) El conjunto de las MT que nunca se mueven a la izquierda.

Sea

$$L_b = \{\langle M\rangle : M \text{ nunca se mueve a la izquierda}\}$$

Para ver que es decidible, basta con buscar si hay alguna transición con movimiento I que pueda ejecutarse. Como antes de la primera vez la máquina se mueve a la izquierda, todos los movimientos son estáticos o a la derecha, la máquina nunca vuelve a leer una casilla anterior. Por tanto, antes de ese posible primer movimiento a la izquierda, lo único que puede leer es, o bien, símbolos de la entrada, o bien, si los lee todos, blancos #.

Así, podemos construir un grafo finito de configuraciones resumidas, con tripletas (q, t, b), con $q \in Q$ el estado actual, $t \in \{\text{entrada, blancos}\}$ que indica si aún estamos leyendo parte de la entrada, o ya hemos terminado con la entrada y hemos llegado a blanco, y $b \in B$ que indica el símbolo leído en la casilla actual.

Algoritmo 11 Decisor para L_b

Entrada: Una codificación $\langle M\rangle$ de una MT M.
Construimos un grafo finito cuyos nodos son ternas (q, t, b).
Los nodos iniciales son $(q_0, \text{entrada}, a)$, para cada $a \in A$, y $(q_0, \text{blancos}, \#)$.
Para cada nodo (q, t, b) **hacer**
  Si $\delta(q, b) = (p, c, I)$ **entonces**
    Marcamos (q, t, b) con (*).
  Si no, si $\delta(q, b) = (p, c, S)$ **entonces**
    Añadimos una arista dirigida de (q, t, b) a (p, t, c).
  Si no, si $\delta(q, b) = (p, c, D)$ **entonces**
    Si $t =$ entrada **entonces**
      Añadimos aristas dirigidas a $(p, \text{entrada}, a)$, para cada $a \in A$.
    Fin Si
    Añadimos una arista dirigida a $(p, \text{blancos}, \#)$.
  Fin Si
Fin Para
Si algún nodo marcado con (*) es alcanzable desde algún nodo inicial, rechazar.
En caso contrario, aceptar.

Veamos que decide correctamente L_b. Si algún nodo marcado es alcanzable, entonces existe una entrada y una ejecución de M que llega a una configuración en la que se ejecuta una transición con movimiento I. Por tanto, M sí se mueve a la izquierda, y debemos rechazar $\langle M\rangle$.

Recíprocamente, si ningún marcado es alcanzable, entonces ninguna transición con movimiento I puede ejecutarse antes del primer movimiento a la izquierda. Pero todo movimiento a la izquierda tendría que ser precisamente el primero en algún instante. Por tanto, M nunca se mueve a la izquierda en ninguna ejecución, y debemos aceptar $\langle M\rangle$.

c) El conjunto de los pares (M, w) tales que M, al actuar sobre la entrada w, nunca lee una casilla de la cinta más de una vez.

Sea

$L_c = \{\langle M, w\rangle : M \text{ actuando sobre } w \text{ nunca lee una casilla de la cinta más de una vez}\}$

Como estamos hablando de "una" cinta, si el cabezal de dicha cinta cambia de sentido (iba hacia la derecha, y vuelve a la izquierda, o iba hacia la izquierda y vuelve a la derecha), necesariamente vuelve a una casilla ya leída.

Por tanto, mientras no se repita ninguna caislla, el movimiento tiene que ser siempre el mismo, o hacia la derecha, o hacia la izquierda (tampoco puede pararse).

Algoritmo 12 MT M_{L_c}

Entrada: Una codificación $\langle M, w\rangle$
Simulamos M sobre w.
Guardamos la posición actual del cabezal y el conjunto de casillas ya leídas.
También guardamos la dirección del primer movimiento, si ya se ha producido.
Mientras la simulación no haya terminado **hacer**
Si la casilla actual ya había sido leída **entonces**
Rechazar
Fin Si
Marcamos la casilla actual como leída.
Si M se para **entonces**
Aceptar
Fin Si
Ejecutamos un paso de M.
Si el movimiento cambia respecto al sentido anterior **entonces**
Rechazar
Fin Si
Si la cabeza está fuera de la zona inicial de la palabra w, leyendo blancos nuevos, y se repite un estado moviéndose siempre en la misma dirección **entonces**
Aceptar
Fin Si
Fin Mientras

El algoritmo siempre termina porque:

1. Si M vuelve a leer una casilla, lo detectamos y rechazamos.
2. Si M se para antes de repetir casilla, aceptamos.
3. Si M sigue indefinidamente sin repetir casillas, entonces necesariamente avanza siempre en una dirección leyendo blancos nuevos, y como hay finitos estados, algún estado debe repetirse y entonces aceptamos.

Ejercicio 15. Indicar si los siguientes problemas son decidibles, semidecidibles o no semidecidibles:

a) (MANUAL) Determinar si una MT se para para cualquier entrada.

Hecho en el manual, sabemos que es no semidecidible. Se usa una reducción desde C-UNIVERSAL.

b) Determinar si una MT no se para para ninguna entrada.

El complementario de este problema es semidecidible con una MTND que seleccione no deterministícamente una entrada y simule la MT sobre la palabra. Si alguna para, acepta. Dicho complementario no es decidible porque puede reducirse desde PARADA(M, w) (simulamos M sobre w. Si M para, entonces la salida para también. Si M no para, tampoco lo hace la salida por el algoritmo), que no es decidible. Por tanto, el problema será no semidecidible.

c) Determinar si una MT se para, al menos, para una entrada.

Discutido en el apartado anterior. Es el complementario mencionado (semidecidible pero no decidible).

d) Determinar si una MT no se para, al menos, para una entrada.

Sabemos que PARADA es semidecidible pero no decidible, luego su complementario es no semidecidible. Podemos reducir desde C-PARADA(M, w) para ver que este problema no es semidecidible (simulamos M sobre w. Si para, la transformación algorítmica para, y si no para, tampoco para la salida del algoritmo).

Ejercicio 16. Supongamos que tenemos 4 lenguajes L_1, L_2, L_3, L_4 de tal manera que existe una reducción de L_1 a L_2, de L_2 a L_3 y de L_4 a L_3. Para cada una de las siguientes afirmaciones, indicar si para todos los casos son CIERTAS, para todos los casos son FALSAS o si son POSIBLES (en algunos casos son ciertas y, en otros, son falsas).

a) L_1 es recursivamente enumerable pero no recursivo, y L_3 es recursivo.

FALSA. Si L_3 fuera recursivo, entonces también lo sería L_2, ya que $L_2 \propto L_3$, y por el mismo argumento, también sería recursivo L_1, pues $L_1 \propto L_2$, y en particular L_1 sería r.e., lo cual es una contradicción.

b) L_1 no es recursivo y L_4 no es recursivamente enumerable.

POSIBLE. Consideramos L_u (lenguaje asociado a UNIVERSAL), que es r.e. pero no recursivo, y L_d (lenguaje asociado a DIAGONAL), que no es r.e., luego tampoco recursivo.

Ejemplo: $L_i = L_d \quad i = 1, \ldots, 4$. Todas las reducciones se dan por la identidad, y $L_1 = L_d$ no es recursivo, y $L_4 = L_d$ no es r.e.

Contraejemplo: $L_i = L_u \quad i = 1, \ldots, 4$. Todas las reducciones se dan por la identidad, pero ahora $L_1 = L_u$ no es recursivo, pero $L_4 = L_u$ sí es r.e.

c) El complementario de L_1 no es recursivamente enumerable, pero el complementario de L_2 es recursivamente enumerable.

FALSA. Sabemos que $L_1 \propto L_2 \iff \overline{L_1} \propto \overline{L_2}$. Entonces, si $\overline{L_2}$ fuera r.e., también lo sería $\overline{L_1}$. Contradicción.

d) El complementario de L_2 no es recursivo, pero el complementario de L_3 es recursivo.

FALSA. Igual que en el apartado anterior, $L_2 \propto L_3 \iff \overline{L_2} \propto \overline{L_3}$. Entonces, si $\overline{L_3}$ fuera recursivo, también lo sería $\overline{L_2}$. Contradicción.

e) Si L_1 es recursivo, entonces el complementario de L_2 es recursivo.

POSIBLE. Consideramos $L_V = \emptyset$ (lenguaje vacío), que es recursivo, luego r.e., y A^* (todas las palabras del alfabeto $A = \{0, 1\}$), que es recursivo.

Ejemplo: $L_i = L_V \quad i = 1, \ldots, 4$. Todas las reducciones se dan por la identidad, y $L_1 = L_V$ es recursivo, y $\overline{L_2} = A^*$ es recursivo.

Contraejemplo: $L_1 = A^*$ y $L_i = L_u \quad i = 2, \ldots, 4$. Se reduce $A^* = L_1 \propto L_2 = L_u$ enviando toda palabra de L_1 a una palabra fija de L_2. Las reducciones

$L_2 \propto L_3$ y $L_4 \propto L_3$ se dan por la identidad, pero ahora $L_1 = A^*$ es recursivo, pero $\overline{L_2} = \overline{L_u}$, que no es recursivo (porque está asociado a C-UNIVERSAL, que no es semidecidible, luego $\overline{L_u}$ no es r.e., y tampoco recursivo).

f) Si L_3 es recursivo, entonces el complementario de L_4 es recursivo.

CIERTA. Si L_3 es recursivo, como $L_4 \propto L_3$, L_4 es recursivo, y entonces, por un teorema visto en teoría, $\overline{L_4}$ también.

g) Si L_3 es recursivamente enumerable, entonces la unión de L_2 y L_4 es recursivamente enumerable.

CIERTA. Si L_3 es r.e., como $L_2 \propto L_3$ y $L_4 \propto L_3$, se tiene que L_2 y L_4 son r.e., y como la unión de dos r.e. es r.e., entonces la afirmación es cierta.

h) Si L_3 es recursivamente enumerable, entonces la intesección de L_2 y L_4 es recursivamente enumerable.

CIERTA. Ya sabemos que si L_3 es r.e. entonces L_2 y L_4 son r.e., y como la intersección de dos r.e. también es r.e., la afirmación es cierta.

Ejercicio 17. Sea el problema de determinar si una máquina de Turing acepta a lo más 100 palabras. Determinar si es decidible, semidecidible o no semidecidible. Justificar la respuesta.

Sea $P(M) = \{\langle M\rangle : |L(M)| \leqslant 100\}$. Veamos que es no semidecidible. Para ello, consideramos el problema complementario $\overline{P}(M) = \{\langle M\rangle : |L(M)| > 100\}$.

La propiedad "$|L(M)| > 100$" solo depende de $L(M)$, no de la MT M concreta. Por tanto, es una propiedad de los lenguajes r.e.. Además, existe una MT que no la cumple (por ejemplo, una MT sin estados finales), y existe una MT que la cumple (por ejemplo, una MT con mismo estado inicial y final). Por lo tanto, es no trivial, y entonces, por el Teorema de Rice, $\overline{P}$ es no decidible.

Para la semidecidibilidad, usamos una MTND descrita como sigue:

Algoritmo 13 MTND M_P

Entrada: Una codificación $\langle M\rangle$ de una MT M
Selecciona de forma no determinista 101 palabras $u_1, \ldots, u_{101} \in \{0,1\}^*$.
Si $\exists i, j \in \{1, \ldots, 101\} : i \neq j \quad \wedge \quad u_i = u_j$ **entonces**
    M_P rechaza.
Fin Si
Para $i = 1, \ldots, 101$ **hacer**
    Simulamos M con entrada u_i.
Fin Para
Si todas las simulaciones aceptan **entonces**
    M_P acepta.
Fin Si

Tenemos entonces que, si M acepta al menos 101 palabras distintas $u_1, \ldots, u_{101}$, existe una rama de cómputo de M_P que precisamente toma $u_1, \ldots, u_{101}$, comprueba que son distintas dos a dos, simula $M(u_1), \ldots, M(u_{101})$, y como todas aceptan, esa rama acepta.

Por otro lado, si M no acepta al menos 101 palabras distintas, entonces toda rama de cómputo de M_P o bien selecciona dos palabras iguales, en cuyo caso rechaza, o bien selecciona 101 palabras distintas. En este último caso, al menos una de ellas no pertenece a $L(M)$, luego al menos una simulación no aceptará y, consecuentemente, esa rama no aceptará. Por lo tanto, M_P acepta exactamente las codificaciones $\langle M\rangle$ tales que M acepta al menos 101 palabras distintas, de donde $\overline{P}$ es semidecidible.

Como $\overline{P}$ es semidecidible pero no decidible, su complementario $\overline{\overline{P}} = P$ es no semidecidible.

Ejercicio 18. Determinar cuáles de los siguientes problemas son decidibles, semidecidibles o no semidecidibles:

a) Saber si una MT para una entrada 0011 no va a usar más de 10 casillas de la cinta.

Veamos que es decidible, para ello, siguiendo el manual, basta ver que "el problema se reduce a realizar un número finito de comprobaciones, cada una de ellas en un número finito de pasos." Definimos

$$\mathrm{P}(M) = \{\langle M\rangle : \text{ con entrada 0011, no usa más de 10 casillas}\}$$

Primero, la propiedad "con entrada 0011 no usa más de 10 casillas" no es una propiedad del lenguaje $L(M)$, luego no es una propiedad de los lenguajes r.e., y no puede aplicarse el Teorema de Rice. Para ver que es decidible, consideramos una MT M_P que funciona como sigue:

Algoritmo 14 MT M_P

Entrada: Una codificación $\langle M\rangle$ de una MT M
Simulamos M con entrada 0011, partiendo de su configuración inicial.
En una segunda cinta vamos guardando las casillas distintas visitadas por M y su número (contador).
Si contador > 10 **entonces**
M_P rechaza.
Si M se para y contador $\leqslant 10$ **entonces**
M_P acepta.
Si se repite una configuración y contador $\leqslant 10$ **entonces**
M_P acepta.

Al simular M_p, pueden pasar tres cosas:

1. Que intente usar más de 10 casillas, y entonces M_P debe rechazar.
2. Que pare sin superar 10 casillas, y entonces M_P debe aceptar.
3. Que no pare (cicle), pero entonces, como hay finitas configuraciones posibles, necesariamente se repetirá alguna en tiempo finito, y entonces ciclará siempre sin salir de esas 10 casillas, y entonces M_P debe aceptar.

Por lo tanto, M_P es un algoritmo que resuelve el problema, lo cual lo hace decidible.

b) Dadas dos MT saber si aceptan el mismo lenguaje.

Definimos IGUAL$(M_1, M_2) = \{(\langle M_1 \rangle, \langle M_2 \rangle) : L(M_1) = L(M_2)\}$. Veamos que no es semidecidible. Podría hacerse directamente, pero primero, para ver que no es decidible, fijamos una MT $M_\emptyset$ tal que

$$L(M_\emptyset) = \emptyset.$$

Consideramos el problema

$$P_{M_\emptyset} = \{\langle M \rangle : L(M) = L(M_\emptyset)\}.$$

Como $L(M_\emptyset) = \emptyset$, entonces

$$L(M) = L(M_\emptyset) \iff L(M) = \emptyset.$$

Por tanto,

$$P_{M_\emptyset} = \{\langle M \rangle : L(M) = \emptyset\}.$$

Si IGUAL fuera decidible, también lo sería $P_{M_\emptyset}$, lo cual contradice el Teorema de Rice, ya que la propiedad "$L(M) = \emptyset$" es una propiedad de los lenguajes r.e. (depende solo de $L(M)$) y además es no trivial: la cumple una MT sin estados finales, pero no la cumple una MT que acepte, por ejemplo, la palabra 0. Por tanto, IGUAL no es decidible.

Veamos que, además, no es semidecidible por medio de una reducción desde

$$\text{VACIO}(M) = \{\langle M \rangle : L(M) = \emptyset\}$$

Fijada una MT M, tal que $L(M) = \emptyset$, es decir, $M \in$ VACIO, definimos la reducción

$$F(\langle N \rangle) = (\langle N \rangle, \langle M \rangle)$$

Intuitivamente, comparar una MT fijada vacía convierte VACIO en IGUAL. Entonces:

1. Si $L(N) = \emptyset$, se tiene que $L(N) = \emptyset = L(M)$, luego $F(\langle N \rangle) \in$ IGUAL.
2. Si $L(N) \neq \emptyset$, se tiene que $L(N) \neq L(M)$, luego $F(\langle N \rangle) \notin$ IGUAL.

Se tiene entonces que VACIO $\propto$ IGUAL. Como VACIO no es semidecidible, tampoco lo es IGUAL.

Ejercicio 19. Determinar cuáles de los siguientes problemas son decidibles, semidecidibles o no semidecidibles:

a) Dada una MT M y una palabra u, saber si la MT acepta la palabra u en un número de pasos menor o igual a $|u|$.

Definimos

$P(M,u) = \{\langle M,u\rangle : M \text{ acepta } u \text{ en un número de pasos menor o igual a } |u|\}$.

Veamos que es decidible. Para ello, consideramos una MT M_P que funciona como sigue, donde la configuración inicial de una MT con entrada u es (q_0, ε, u):

Algoritmo 15 MT M_P

Entrada: Una codificación $\langle M,u\rangle$ de una MT M y una palabra u
Para $i = 0, \ldots, |u|$ **hacer**
    Simulamos i pasos de M con entrada u, partiendo de su configuración inicial.
    Si M entra en un estado de aceptación **entonces**
        M_P acepta.
    Si M para sin aceptar **entonces**
        M_P rechaza.
    Si $i = |u|$ y M no acepta **entonces**
        M_P rechaza.

Al simular M_P, pueden pasar tres cosas:

1. Que M acepte en k pasos con $k \leq |u|$, y entonces M_P debe aceptar.
2. Que M se pare antes o en $|u|$ pasos, pero sin aceptar, y entonces M_P debe rechazar.
3. Que M no haya aceptado tras $|u|$ pasos. En ese caso, aunque después aceptara o ciclara, ya no cumpliría la condición pedida, y entonces M_P debe rechazar.

Por tanto, M_P siempre termina, ya que sólo necesita simular, como mucho, $|u|$ pasos de cálculo, y decide correctamente si M acepta u en un número de pasos menor o igual a $|u|$, y el problema es decidible.

b) Dada una MT determinar si no acepta la palabra vacía.

Deberíamos intuir, al ser de la forma "determinar si no acepta (algo)" que no es semidecidible. Definimos $P_\varepsilon = \{\langle M\rangle : M \text{ no acepta } \varepsilon\}$. Vemos que es el complementario de $C - P_\varepsilon = \{\langle M\rangle : M \text{ acepta } \varepsilon\}$. Sin embargo, $C - P_\varepsilon$ es semidecidible, ya que podemos construir una MT M_{P_ε} que, con entrada $\langle M\rangle$, simule a M sobre la palabra vacía ε, y acepte si M acepta. Si además P_ε fuera semidecidible entonces P_ε y $C - P_\varepsilon$ serían semidecidibles, lo que haría que P_ε fuera decidible, y a su vez, que $C - P_\varepsilon$, al ser complementario de decidible, también lo fuera. Sin embargo, $C - P_\varepsilon$ no es decidible, ya que "aceptar ε" depende solo de $L(M)$, porque equivale a ver si $\varepsilon \in L(M)$. Entonces es una propiedad de los lenguajes r.e., y además es no trivial, ya que hay una MT cuyo lenguaje aceptado es $\{\varepsilon\}$ (por ejemplo, una transición que lea blanco y pase a estado final), que sí la cumple, pero también otra MT cuyo lenguaje aceptado es $\{0\}$, que no la cumple. Por tanto, $C - P_\varepsilon$ no es decidible por el Teorema de Rice, lo cual sería una contradicción. Necesariamente entonces P_ε no es semidecidible.

Ejercicio 20. Determinar cuáles de los siguientes problemas son decidibles, semidecidibles o no semidecidibles (se supone que las MTs tienen a $A = \{0, 1\}$ como alfabeto de entrada):

a) Dadas dos máquinas de Turing, M_1 y M_2, determinar si existe, al menos, una palabra aceptada simultáneamente por ambas máquinas.

Sea $P = \{\langle M_1, M_2\rangle : L(M_1) \cap L(M_2) \neq \emptyset\}$. Veamos que es semidecidible pero no decidible. La no decidibilidad se deduce del Teorema de Rice, ya que, fijada una MT M_* que acepte todas las palabras, $L(M_*) = A^*$, consideramos el problema siguiente

$$P_{M_*} = \{\langle M\rangle : L(M) \cap L(M_*) \neq \emptyset\}$$

Como $L(M_*) = A^*$, entonces

$$L(M) \cap L(M_*) \neq \emptyset \iff L(M) \neq \emptyset$$

Por tanto, $P_{M_*} = \{\langle M\rangle : L(M) \neq \emptyset\}$. Si P fuera decidible, también lo sería P_{M_*}, lo cual es una contradicción con el Teorema de Rice, ya que la propiedad "$L(M) \neq \emptyset$" es una propiedad de los lenguajes r.e. (solo depende de $L(M)$) y además es no trivial, ya que si M es una MT sin estados finales, entonces $L(M) = \emptyset$, y si M es una MT que acepta todas las palabras, se tiene que $L(M) = A^* \neq \emptyset$. Por tanto, P es no decidible.

La semidecidibilidad se obtiene con la siguiente MTND:

Algoritmo 16 MTND M_P

Entrada: Un par de codificaciones $\langle M_1, M_2\rangle$ de dos MT M_1 y M_2
Selecciona de forma no determinista una palabra $w \in A^*$.
Para $i = 1, 2$ **hacer**
  Simulamos M_i con entrada w.
Fin Para
Si ambas simulaciones aceptan **entonces**
  M_P acepta.
Fin Si

b) (MANUAL) Dada una MT, determinar si para toda palabra de entrada no realiza más de 5 movimientos.

Hecho en el manual con 10 movimientos (pasos de cálculo). La clave está en ver que en 5 movimientos solo se pueden leer cinco casillas de la cinta como máximo. Entonces las palabras se identifican para este problema con los prefijos de longitud, a lo sumo, 5. Así, dada una palabra $u \in A^*$ tal que $|u| \leqslant 5$, y dada una palabra $v \in A^*$, entonces, para una MT M cualquiera,

M no realiza más de 5 movimientos con $u \iff M$ no realiza más de 5 movimientos con uv

El conjunto $\{u \in A^* : |u| \leqslant 5\}$ es finito. De hecho, podemos obtener su cardinal:

$$\sum_{k=0}^{5} 2^k = \frac{2^6 - 1}{2 - 1} = 63$$

y basta ejecutar 5 pasos de cálculo para cada una de estas 63 palabras. Si con todas para, la respuesta es que sí, y si con alguna no ha parado, la respuesta es que no.

c) Determinar si una MT no es determinista.

Sea $P = \{\langle M \rangle : M \text{ es una MTND}\}$. Podemos utilizar el siguiente algoritmo para resolver el problema:

Algoritmo 17 MT M_P

Entrada: Una codificación $\langle M \rangle$ de una MT M
Leemos la codificación de M, en particular su conjunto de estados Q, su alfabeto de cinta B y su tabla de transiciones.
Para cada par $(q, a) \in Q \times B$ **hacer**
  Contamos cuántas transiciones están definidas desde (q, a).
  Si hay dos o más transiciones definidas desde (q, a) **entonces**
    M_P acepta.
  Fin Si
Fin Para
M_P rechaza.

Como Q, B y la tabla de transiciones de M son finitos, el algoritmo siempre termina, y el algoritmo anterior responde correctamente en todos los casos. Por tanto, P es decidible y, en particular, semidecidible.

d) (MANUAL) Dada una MT, determinar si ninguna de las palabras que acepta es un palíndromo.

Es el complementario del problema del Ejercicio 9, que es semidecidible pero no decidible, luego este problema es no semidecidible.

Ejercicio 21. Se considera el siguiente problema: dadas dos máquinas de Turing, determinar si aceptan el mismo lenguaje. Razonar si este problema es decidible, semidecidible o no semidecidible.

Es igual que el Ejercicio 18 b).

Ejercicio 22. Determinar, justificando las respuestas, cuáles de los siguientes problemas son decidibles, semidecidibles o no semidecidibles:

a) Dada una máquina de Turing y una palabra de entrada determinar si visita menos de 10 casillas distintas de la cinta de lectura.

Es igual que el Ejercicio 18 a), pero con una entrada $w \in \{0,1\}^*$ cualquiera.

Solo cambiaría la definición del problema

$$P(M,w) = \{\langle M, w\rangle : M \text{ con entrada } w \text{ visita menos de 10 casillas distintas}\}$$

y la entrada del algoritmo sería una codificación $\langle M, w\rangle$ de una MT M y una palabra w, y habría que simular M sobre w y seguir la misma lógica que en dicho ejercicio.

Este problema también es decidible.

b) Dadas dos Máquinas de Turing, determinar si el conjunto de las palabras aceptadas a la vez por ambas máquinas de Turing es finito.

Podríamos ver en dos pasos no decidibilidad con Rice, y luego no semidecidibilidad, pero veamos directamente que no es semidecidible por medio de una reducción desde VACIO (que sabemos por teoría que no es semidecidible).

Definimos $P(M_1, M_2) = \{\langle M_1, M_2\rangle : L(M_1) \cap L(M_2) \text{ es finito}\}$. Reducimos desde

$$\text{VACIO}(M) = \{\langle M\rangle : L(M) = \emptyset\}$$

Dada una MT M, construimos M' dada como sigue:

Algoritmo 18 MT M'

Entrada: Una palabra $x \in \{0,1\}^*$
Ignoramos la entrada x.
Para $t = 1, \ldots, \infty$ **hacer**
  Para cada palabra $w \in \{0,1\}^*$ con $|w| \leqslant t$ **hacer**
    Simulamos M con entrada w durante t pasos.
    Si M acepta w en esos t pasos **entonces**
      M' acepta x.
    Fin Si
  Fin Para
Fin Para

Es decir, se tiene que

$$L(M') = \begin{cases} \emptyset & \text{si } L(M) = \emptyset \\ \{0,1\}^* & \text{si } L(M) \neq \emptyset \end{cases}$$

Fijamos ahora una MT M_0 tal que $L(M_0) = \{0,1\}^*$. Definimos la reducción

$$F(\langle M \rangle) = (\langle M' \rangle, \langle M_0 \rangle)$$

Entonces $L(M') \cap L(M_0) = L(M')$, y

$$L(M) = \emptyset \iff L(M') = \emptyset \iff L(M') \cap L(M_0) \text{ es finito}$$

Por tanto, VACIO $\propto P$. Como VACIO no es semidecidible, tampoco lo es P.

Ejercicio 23. Indica cuáles de los siguientes problemas son decidibles, semidecidibles o no semidecidibles:

a) Determinar si el lenguaje aceptado por una MT es regular.

Sea REGULAR = $\{\langle M\rangle : L(M) \text{ es regular}\}$. Podemos ver que no es decidible y luego que no es semidecidible en dos pasos, o bien directamente que no es semidecidible por medio de una reducción desde VACIO. Para ver que es no decidible usamos el Teorema de Rice, ya que la propiedad "$L(M)$ es regular" depende solo de $L(M)$, luego es una propiedad de los lenguajes r.e. y es no trivial, porque existe una MT M_1 con $L(M_1) = \emptyset$ (sin estados finales por ejemplo), y $L(M_1)$ cumple la propiedad, y existe una MT M_2 con $L(M_2) = \{0^n1^n : n \geqslant 0\}$ (por ejemplo, multicinta que cuente cuántos 0's aparecen y ponga ese mismo número de 1's a continuación), y $L(M_2)$ no cumple la propiedad.

La no semidecidibilidad puede obtenerse con una reducción desde

$$\text{VACIO}(M) = \{\langle M\rangle : L(M) = \emptyset\}$$

Dada una MT M, construimos una MT M' con el siguiente algoritmo:

Algoritmo 19 MT M'

Entrada: Una palabra $x \in \{0,1\}^*$
Comprobamos si $x \in \{0^n1^n : n \geqslant 0\}$.
Si $x \notin \{0^n1^n : n \geqslant 0\}$ **entonces**
  M' rechaza.
Fin Si
Para $t = 1, \ldots, \infty$ **hacer**
  Para cada palabra $w \in \{0,1\}^*$ con $|w| \leqslant t$ **hacer**
    Simulamos M con entrada w durante t pasos.
    Si M acepta w en esos t pasos **entonces**
      M' acepta x.
    Fin Si
  Fin Para
Fin Para

Tendremos entonces que

$$L(M') = \begin{cases} \emptyset & \text{si } L(M) = \emptyset \\ \{0^n1^n : n \geqslant 0\} & \text{si } L(M) \neq \emptyset \end{cases}$$

Como $\emptyset$ es regular, pero $\{0^n1^n : n \geqslant 0\}$ no lo es, se tiene que

$$L(M) = \emptyset \iff L(M') = \emptyset \iff L(M') \text{ es regular}$$

Definimos la reducción $F(\langle M\rangle) = \langle M'\rangle$. Viendo que

$$\langle M\rangle \in \text{VACIO} \iff F(\langle M\rangle) = \langle M'\rangle \in \text{REGULAR}$$

deducimos que VACIO $\propto$ REGULAR. Como VACIO no es semidecidible, tampoco lo es REGULAR.

b) Dadas dos MTs determinar si hay una palabra que es aceptada por las dos MT y además es un palíndromo.

Sea $P = \{\langle M_1, M_2\rangle : \exists w \in \{0,1\}^* \text{ palíndromo} : w \in L(M_1) \cap L(M_2)\}$. Veamos que es semidecidible pero no decidible. La no decidibilidad se deduce del Teorema de Rice, ya que, fijada una MT M_* que acepte todas las palabras, $L(M_*) = A^*$, consideramos el problema siguiente

$$P_{M_*} = \{\langle M\rangle : \exists w \in \{0,1\}^* \text{ palíndromo} : w \in L(M) \cap L(M_*)\}$$

Como $L(M_*) = A^*$, entonces, considerando $\text{PAL} = \{w \in \{0,1\}^* : w \text{ es un palíndromo}\}$:

$$\exists w \in \{0,1\}^* \text{ palíndromo} : w \in L(M) \cap L(M_*) \iff L(M) \cap \text{PAL} \neq \emptyset$$

Por tanto, $P_{M_*} = \{\langle M\rangle : L(M) \cap \text{PAL} \neq \emptyset\}$. Si P fuera decidible, también lo sería P_{M_*}, lo cual es una contradicción con el Teorema de Rice, ya que la propiedad "$L(M) \cap \text{PAL} \neq \emptyset$" es una propiedad de los lenguajes r.e. (solo depende de $L(M)$) y además es no trivial, ya que si M es una MT sin estados finales, entonces $L(M) = \emptyset$, y no cumple la propiedad, y si M es una MT que acepta todas las palabras, se tiene que $L(M) = A^* \neq \emptyset$, que sí cumple la propiedad (contiene a todos los palídromos de hecho). Por tanto, P es no decidible.

La semidecidibilidad se obtiene con la siguiente MTND:

Algoritmo 20 MTND M_P

Entrada: Un par de codificaciones $\langle M_1, M_2\rangle$ de dos MT M_1 y M_2
Selecciona de forma no determinista una palabra $w \in A^*$.
Comprueba si w es un palíndromo.
Si w no es un palíndromo **entonces**
    M_p rechaza.
Fin Si
Para $i = 1, 2$ **hacer**
    Simulamos M_i con entrada w.
Fin Para
Si ambas simulaciones aceptan **entonces**
    M_P acepta.
Fin Si

Ejercicio 24. Indica cuáles de los siguientes problemas son decidibles, semidecidibles o no semidecidibles:

a) Dada una MT determinar si todos sus estados son accesibles (se puede llegar a ellos para un cálculo para alguna entrada).

Sea

$$P = \{\langle M\rangle : \forall q \in Q_M, \exists w \in \{0,1\}^* : \exists t \in \mathbb{N} : M(w) \text{ accede al estado } q \text{ en } t \text{ pasos}\}$$

El problema es semidecidible pero no decidible. Podemos ver la semidecidibilidad con la siguiente MTND:

Algoritmo 21 MTND M_P

Entrada: Una codificación $\langle M\rangle$ de una MT M
Sea $Q_M = \{q_1, \ldots, q_k\}$ el conjunto de estados de M.
Para $i = 1, \ldots, k$ **hacer**
  Selecciona de forma no determinista una palabra $w_i \in \{0,1\}^*$.
  Selecciona de forma no determinista un número $t_i \in \mathbb{N}$.
Fin Para
Para $i = 1, \ldots, k$ **hacer**
  Simula M con entrada w_i durante, a lo sumo, t_i pasos.
  Si durante esa simulación no se visita el estado q_i **entonces**
    M_p rechaza.
  Fin Si
Fin Para
M_p acepta.

Si todos los estados son accesibles, existirá, para cada $i = 1, \ldots, k$, un estado $q_i \in Q_M$, una palabra $w_i \in \{0,1\}^*$ y un número t_i tales que la ejecución de M sobre w_i visita q_i en, a lo sumo, t_i pasos.

Para probar que P no es decidible, reducimos desde

$$\text{UNIVERSAL}(M, w) = \{\langle M, w\rangle : M \text{ acepta } w\},$$

que sabemos que no es decidible.

Dada una instancia $\langle M, w\rangle$ de UNIVERSAL, construimos una MT M' tal que todos sus estados sean accesibles salvo, quizá, un estado especial q_*. La máquina M' ignora su entrada y funciona como sigue:

Algoritmo 22 MT M'

Entrada: Una palabra $x \in \{0,1\}^*$
M' ignora la entrada x.
Obtenemos $Q' \setminus \{q_*\} = \{p_0, p_1, \ldots, p_m\}$, donde p_0 es el estado inicial de M'.
M' realiza una subrutina de accesibilidad que recorre todos los estados de M' salvo q_*, mediante transiciones auxiliares que no interfieren con la simulación posterior.
Al llegar al estado p_m, M' coloca en la cinta $\langle M\rangle 111w$.
Desde p_m, M' comienza la simulación de M con entrada w.
Si M acepta w **entonces**
    M' entra en el estado especial q_* y acepta.
Fin Si
Si M rechaza w **entonces**
    M' rechaza sin pasar por q_*.
Fin Si
Si M no para con w, entonces M' tampoco para y nunca entra en q_*.

Por construcción, todos los estados de M' distintos de q_* son accesibles, ya que la subrutina de accesibilidad pasa por todos ellos.

El único estado cuya accesibilidad no queda garantizada entonces es q_*. Además, por construcción, M' entra en q_* si y solo si la simulación de M sobre w acepta. Deducimos de lo anterior que

$$q_* \text{ es accesible en } M' \iff M \text{ acepta } w.$$

Como todos los demás estados son accesibles por construcción, se tiene

$$\text{todos los estados de } M' \text{ son accesibles} \iff q_* \text{ es accesible en } M' \iff M \text{ acepta } w.$$

Por tanto,

$$\langle M, w\rangle \in \text{UNIVERSAL} \iff \langle M'\rangle \in P.$$

Así,

$$\text{UNIVERSAL} \propto P.$$

Si P fuera decidible, entonces UNIVERSAL también sería decidible, contradicción. Por tanto, P no es decidible.

b) Dada una MT determinar si su número de estados es menor o igual a 5.

Sea $P = \{\langle M \rangle : |Q_M| \leqslant 5\}$. El problema es decidible, y se resuelve con el siguiente algoritmo:

Algoritmo 23 MT M_P

Entrada: Una codificación $\langle M \rangle$ de una MT M
Comprobamos que $\langle M \rangle$ es una codificación correcta de una MT.
Leemos el conjunto de estados Q de M.
Contamos cuántos estados tiene M.
Si $|Q| \leqslant 5$ **entonces**
    M_p acepta.
Si no
    M_p rechaza.
Fin Si

Ejercicio 25. Demuestra que el Problema de la Correspondencia de Post con un alfabeto A que tiene un sólo elemento es decidible.

Sea $a \in A$ tal que $A = \{a\}$. Entonces, $A^* = \{a^n : n \geqslant 0\}$. Cada bloque del PCP será de la forma

$$\frac{a^{p_i}}{a^{q_i}}$$

con p_i la longitud de la palabra superior, y q_i la longitud de la palabra inferior, $i = 1, \ldots, k$, y k el número de bloques. Entonces, para una sucesión no vacía de enteros $i_1, \ldots, i_m$, la palabra de arriba será $a^{p_{i_1}} \cdots a^{p_{i_m}} = a^{p_{i_1}+\cdots+p_{i_m}}$. Análogamente, la palabra de abajo será $a^{q_{i_1}+\cdots+q_{i_m}}$. Entonces, como solo hay un símbolo, dos palabras son iguales si y solo si tienen la misma longitud, es decir, la sucesión es válida (como solución al PCP) si y solo si $p_{i_1} + \cdots + p_{i_m} = q_{i_1} + \cdots + q_{i_m}$. Definimos $d_i = p_i - q_i$. La condición de solución se reescribe como

$$p_{i_1} + \cdots + p_{i_m} = q_{i_1} + \cdots + q_{i_m} \iff (p_{i_1} - q_{i_1}) + \cdots + (p_{i_m} - q_{i_m}) = 0 \iff$$

$$d_{i_1} + \cdots d_{i_m} = 0$$

Para ver que es decidible usamos el siguiente algoritmo:

Algoritmo 24 MT M_{PCP}

Entrada: Una instancia de PCP sobre $A = \{a\}$ con bloques $\frac{a^{p_1}}{a^{q_1}}, \ldots, \frac{a^{p_k}}{a^{q_k}}$.
Para $i = 1, \ldots, k$ **hacer**
  Calculamos $d_i = p_i - q_i$.
Fin Para
Si existe i tal que $d_i = 0$ **entonces**
  Aceptamos.
Fin Si
Si existen i, j tales que $d_i > 0$ y $d_j < 0$ **entonces**
  Aceptamos.
Fin Si
Rechazamos.

Si algún $d_i = 0$, entonces el bloque i es solución. Si hay $d_i > 0$ y $d_j < 0$ para $i, j \in \{1, \ldots, k\}$, escribimos $d_i = r > 0$, y $d_j = -s < 0$. Tomando s copias del bloque i y r copias del bloque j, la diferencia total será $s \cdot r + r \cdot (-s) = 0$, luego hay solución. Si todos los d_i son positivos, cualquier suma no vacía será positiva. Si, por otro lado, todos los d_i son negativos, cualquier suma no vacía será negativa. En ninguno de estos dos casos puede haber solución. Como el algoritmo solo recorre un número finito de bloques y siempre termina, el PCP sobre un alfabeto con un único símbolo es decidible.