MODELOS AVANZADOS DE COMPUTACIÓN

Manual práctico sobre decibilidad y semidecibilidad

1 Formas usuales de demostrar el tipo de un problema

- **Es decidible**. Diseñar un algoritmo que resuelva el problema. Normalmente, esto se demuestra probando que el problema se reduce a realizar un número finito de comprobaciones (aunque, a priori, no lo parezca), cada una de ellas en un número finito de pasos. Entonces, una máquina de Turing que compruebe todas las posibilidades es un algoritmo que resuelve el problema (siempre para, ya que realiza un número finito de pasos).

- **No es decidible**. Para demostrar que un problema no es decidible contamos con el teorema de Rice:

  Teorema 1.1 (Teorema de Rice) *Toda propiedad no trivial de los lenguajes recursivamente enumerables no es decidible.*

  Por propiedad de los lenguajes recursivamente enumerables se debe entender una propiedad que sólo depende del lenguaje en sí y que es independiente de las diferentes máquinas de Turing que acepten dicho lenguaje. Por ejemplo, si consideramos el problema

  "Dada un MT M, determinar si M acepta una palabra que empiece por 0",

  la propiedad "aceptar una palabra que empieza por 0" es una propiedad de los lenguajes recursivamente enumerables, porque es una propiedad que sólo depende de $\mathcal{L}(M)$. Dicho de otra manera, si otra máquina de Turing M' acepta el mismo lenguaje, M cumple esa propiedad si, y sólo si, lo hace M'.

  Por otro lado, si tenemos el problema

  "Dada un MT M, determinar si M acepta una palabra en menos de 5 movimientos",

  la propiedad "acepta una palabra en menos de 5 movimientos" no es una propiedad de los lenguajes recursivamente enumerables. Es una propiedad asociada a las máquinas de Turing ya que, para dos máquinas de Turing que acepten el mismo lenguaje, una puede aceptar una palabra en menos de 5 movimientos y la otra, no.

  Una propiedad de los lenguajes recursivamente enumerables es trivial cuando lo cumplen todos los lenguajes r.e., o bien, no lo cumple ninguno. Por ejemplo,

  – La propiedad "ser aceptado por una máquina de Turing" es trivial, todos los lenguajes r.e. lo cumplen.

- La propiedad "existe una reducción de L_d a L" es trivial, ningún lenguaje r.e. L lo cumple (porque si no, no sería r.e.).
- La propiedad "ser numerable" es trivial, todos los lenguajes son numerables.

Por tanto, si nuestro problema involucra una propiedad no trivial de los lenguajes r.e., simplemente aplicamos el teorema de Rice para demostrar que no es decidible.

Si la propiedad no es de los lenguajes r.e., entonces necesitamos establecer una reducción desde un problema no decidible. Algunos problemas que se pueden utilizar para esto son:

- UNIVERSAL: Dada una MT M y una entrada w, determinar si M acepta w. El lenguaje asociado es el lenguaje universal L_u.
- PARADA: Dada una MT M y una entrada w, determinar si M para con entrada w.
- C-DIAGONAL: Dada una MT M, determinar si M acepta su codificación $\langle M \rangle$.
- En general, cualquier problema visto en clase que no sea decidible.

- **Es semidecidible**. Diseñar una máquina de Turing que acepte las palabras del lenguaje asociado. Esta puede ser determinista o no determinista. Como ayuda para intuir si un problema es semidecidible, normalmente, los problemas semidecidibles son aquellos que requieren comprobar que, en un conjunto infinito A (si es finito, además, es decidible) existe un elemento que cumple cierta propiedad. En este caso, la máquina de Turing no determinista sigue el siguiente esquema:

Máquina de Turing no determinista genérica

Entrada: Propiedad P, conjunto de posibilidades A

1: Seleccionamos de forma no determinista $a \in A$
2: **Si** a cumple la propiedad P **entonces**
3: **acepta** la entrada
4: **Si no**
5: **rechaza** la entrada

Obviamente, esto se puede extender a que existan un número finito n de elementos que cumplan esa propiedad P. Bastaría seleccionar de forma no determinista n elementos distintos de A.

- **No es semidecidible**. Establecer una reducción desde un problema que sabemos que no es semidecidible. Algunos problemas que se pueden utilizar para esto son:

- DIAGONAL: Dada una MT M, determinar si M **no** acepta su codificación $\langle M \rangle$. El lenguaje asociado es el lenguaje de diagonalización L_d.
- C-UNIVERSAL: Dada una MT M y una entrada w, determinar si M **no** acepta w. El lenguaje asociado es el complementario del lenguaje universal $\overline{L_u}$.
- EMPTY: Dada una MT M, determinar si $\mathcal{L}(M) = \emptyset$. El lenguaje asociado es L_e.
- En general, el complementario de cualquier problema visto en clase que sea semidecidible pero no decidible.

Otra forma de demostrar que un problema/lenguaje no es semidecidible, es demostrar que su complementario es semidecidible pero no decidible.

2 Ejemplos

Todos los ejemplos de las siguientes secciones consisten en determinar si los problemas enunciados son decidibles, semidecidibles o no semidecidibles. Se supone que el alfabeto de entrada siempre es $\{0, 1\}$.

Ejemplo 2.1 *Dadas dos máquinas de Turing M_1 y M_2, determinar si $\mathcal{L}(M_1) \subseteq \mathcal{L}(M_2)$.*

Solución. Este problema necesita comprobar si TODA palabra de $\mathcal{L}(M_1)$ (posiblemente un número infinito) está en $\mathcal{L}(M_2)$, por lo que deberíamos intuir que no es semidecible. Para demostrar esto vamos a realizar una reducción desde un problema que no es semidecible, por lo que nuestro problema tampoco lo será. Consideramos los problemas:
P1: Dada una máquina de Turing M, determinar si $\mathcal{L}(M) = \emptyset$.
P2: Dadas dos máquinas de Turing M_1 y M_2, determinar si $\mathcal{L}(M_1) \subseteq \mathcal{L}(M_2)$.
Sabemos, por las transparencias de teoría, que P1 no es semidecidible, ya que es el problema asociado al lenguaje L_e, que no es recursivamente enumerable.

Una reducción de P1 a P2, $P1 \leq P2$, consiste en asociar a cada instancia de P1 (una máquina de Turing) una instancia de P2 (dos máquinas de Turing) mediante un algoritmo F, de manera que si M es una instancia positiva del problema P1, entonces $F(M)$ es una instancia positiva de P2, y, al contrario, si M es una instancia negativa del problema P1, entonces $F(M)$ es una instancia negativa de P2.

Consideramos la máquina de Turing $R = (\{q_0\}, \{0, 1\}, \{0, 1, \#\}, \delta, q_0, \#, \emptyset)$, donde δ no tiene transiciones, por lo que $\mathcal{L}(R) = \emptyset$ (se puede considerar cualquier MT que no acepte palabras). Entonces, dada una instancia M de P1, le asociamos el par $F(M) = (M, R)$. Obviamente, esto es un proceso algorítmico. Basta con escribir la codificación de R después de la codificación de cualquier entrada M. Además,

- Si M es un caso positivo de P1, es decir, $\mathcal{L}(M) = \emptyset$, entonces $F(M) = (M, R)$ es un caso positivo de P2, ya que $\emptyset = \mathcal{L}(M) \subseteq \mathcal{L}(R) = \emptyset$.
- Si M es un caso negativo de P1, es decir, $\mathcal{L}(M) \neq \emptyset$, entonces $F(M) = (M, R)$ es un caso negativo de P2, ya que $\mathcal{L}(M) \not\subseteq \mathcal{L}(R) = \emptyset$.

Por lo tanto, P1 se reduce a P2, y entonces P2 no es semidecidible.

Ejemplo 2.2 *Dado un autómata con pila no determinista y una palabra de entrada, determinar si el autómata acepta la palabra.*

Solución. Este problema es **decidible** ya que podemos diseñar un algoritmo que lo resuelva. Supongamos que P es un autómata con pila y u, una palabra. Como vimos en la asignatura Modelos de Computación (Tema 5 de teoría) existe un algoritmo que calcula una gramática libre de contexto G que genera el mismo lenguaje que acepta P. En el Tema 4 de teoría de Modelos de Computación se explica un algoritmo que elimina producciones nulas de una gramática libre de contexto. Se lo aplicamos a G y obtenemos una gramática G' que genera las mismas palabras que G, salvo la palabra vacía en caso de que G genere ϵ. Si la palabra u es la palabra vacía, basta con comprobar si el símbolo inicial es anulable. Si lo es, entonces se acepta, y si no, no se acepta. Si la palabra u no es la palabra vacía, aplicamos el algoritmo de eliminación de producciones unitarias y obtenemos una gramática G'' equivalente a G' pero sin producciones unitarias. Entonces G'' está en las condiciones para aplicar el algoritmo de cálculo de la forma normal de Chomsky (Tema 4 de teoría de Modelos de Computación), y obtenemos una gramática C equivalente a G'' en la forma normal de Chomsky. A C y a u se le puede aplicar ahora el algoritmo de Cocke-Younger-Kasami (Tema 6 de teoría de Modelos de Computación) que nos asegura si C genera u, o no. Por tanto, decide si P acepta u, o no.

Ejemplo 2.3 *Dada una máquina de Turing, determinar si acepta una palabra de longitud menor o igual que 20.*

Solución. Ya que se trata de un problema de comprobar la existencia de algo, deberíamos intuir que este problema no es decidible pero sí semidecidible.

Lo primero es sencillo de demostrar, ya que es una propiedad no trivial de los lenguajes recursivamente enumerables. Se puede observar que "aceptar una palabra de longitud menor o igual a 20" es una propiedad del lenguaje aceptado por la maquina de Turing, no de la máquina de Turing en sí. Dicho de otra manera, si dos máquinas de Turing acepta el mismo lenguaje, una acepta una palabra de longitud menor o igual a 20 si y sólo si lo hace la otra. Por tanto, por el Teorema de Rice, este problema no es decidible.

Para demostrar que es semidecidible, podemos construir una máquina de Turing no determinista R que acepte la codificación de las máquinas de Turing que cumple esa propiedad (se entiende que si la cadena de entrada no es la codificación de una máquina de Turing, se rechaza).

Máquina de Turing no determinista R

Entrada: La codificación $\langle M \rangle$ de una máquina de Turing M

1: Añadimos la palabra 111 al final de la entrada
2: De forma no determinista seleccionamos una palabra u de longitud menor que o igual que 20 y la copiamos a continuación de 111
3: Ejecutamos la máquina universal sobre la palabra de la cinta, es decir, sobre $\langle M \rangle 111u$
4: **Si** la máquina universal acepta **entonces**
5: $\quad R$ **acepta**
6: **Si no**
7: $\quad R$ **no acepta**

Vemos que basta con que alguna de las ramas de cómputo (al realizar la elección no determinista de la línea 2) se acepte para que la entrada se acepte, que es precisamente lo que nos pide el problema: *"¿Existe una palabra...?"*.

Otra forma de resolverlo es describiendo una máquina determinista. Primero, una propuesta que **NO** es válida:

Máquina de Turing determinista S QUE NO FUNCIONA

Entrada: La codificación $\langle M \rangle$ de una máquina de Turing M

1: Añadimos la palabra 111 al final de la entrada
2: Puesto que las palabras de longitud 20 o menos son un conjunto finito, llamémoslo A, las ordeno de alguna manera
3: **Para** cada palabra $u \in A$, en el orden establecido, **hacer**
4: $\quad$ Copio $\langle M \rangle 111u$ en una segunda cinta
5: $\quad$ Ejecutamos la máquina universal sobre la segunda cinta
6: $\quad$ **Si** la máquina universal acepta **entonces**
7: $\quad\quad S$ **acepta** la entrada
8: S **rechaza** la entrada (ya que no se ha aceptado ninguna entrada en el bucle)

¿Por qué no funciona? Porque es posible que para alguna palabra u de A, al ejecutar la máquina universal (línea 5), esta se rechace al estar indefinidamente en movimiento (la máquina universal cicle con esa entrada). Por tanto, la máquina diseñada S también ciclaría en esa pasada

del bucle (en la línea 5), y rechazaría la entrada (la salida diría que M no acepta palabras de longitud menor que 21). Sin embargo podría no haber ejecutado todas las pasadas del bucle, es decir, faltarían algunas palabras de A por comprobar, y es posible que alguna de ellas sí es aceptada por M.

¿Cómo arreglar esto? Ejecutando "todas las palabras de A a la vez". Como no podemos realizar una selección no determinista, lo que vamos a hacer es ejecutar un número finito de pasos de cálculo con todas las palabras. Si con ninguna se acepta, aumentamos el número de pasos de cálculo y repetimos. Así de forma indefinida hasta que una palabra se acepte (que se aceptará en un número finito de pasos de cálculo). Si, en realidad, la máquina M no acepta ninguna palabra de A, nuestra nueva máquina ciclara indefinidamente (ya que es un bucle infinito)

Máquina de Turing determinista S

Entrada: La codificación $\langle M \rangle$ de una máquina de Turing M

1: Añadimos la palabra 111 al final de la entrada
2: Puesto que las palabras de longitud 20 o menos son un conjunto finito, llamémoslo A, las ordeno de alguna manera
3: **Para** $i = 1, 2, 3, ..., \infty$ **hacer**
4: **Para** cada palabra $u \in A$, en el orden establecido, **hacer**
5: Copio $\langle M \rangle 111u$ en una segunda cinta
6: Ejecutamos la máquina universal i pasos sobre la segunda cinta
7: **Si** la máquina universal acepta **entonces**
8: S **acepta** la entrada

Una forma de ver la diferencia entre ambas máquinas es pensar en el árbol de pasos de cómputo. La primera máquina, que no resuelve correctamente el problema, realiza una búsqueda en profundidad en los pasos de cómputo de todas las palabras de longitud menor que 21, ver Figura 1. Como hay una rama infinita, la búsqueda en profundidad no es capaz de visitar todos los nodos del árbol. Obsérvese que la máquina M cicla con la palabra 2, por lo que nunca llega a verificar la palabra 3, que es la que sí acepta.

Sin embargo, la segunda máquina realiza una búsqueda en anchura, ver Figura 2. Puesto que, si alguna palabra se acepta, el nodo de aceptación debe estar a profundidad finita, esta búsqueda nos asegura que llegaremos a dicho nodo.

Ejemplo 2.4 *Dada un máquina de Turing M, determinar si, para cualquier entrada, M para antes de 10 pasos de cálculo.*

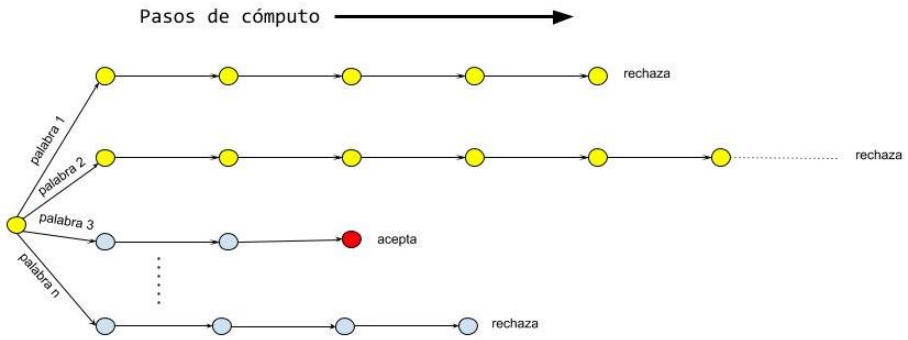

Figure 1: Búsqueda en profundidad

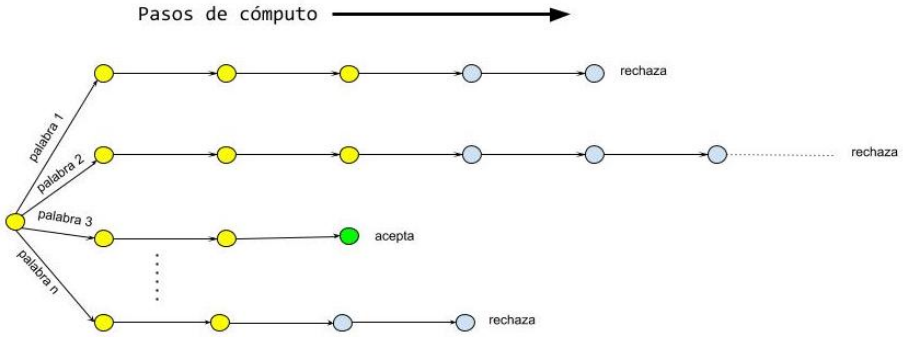

Figure 2: Búsqueda en anchura

Solución. Primero vamos a describir algunos errores muy comunes al resolver este ejercicio.

- ERROR 1. *Este problema no es decidible. Es una propiedad no trivial y entonces, por el teorema de Rice, no es decidible.*

  El problema de este argumento es que la propiedad, que ciertamente no es trivial, no es una propiedad de los lenguajes recursivamente enumerables. Es una propiedad de las máquinas de Turing. Un mismo lenguaje puede ser aceptado por dos máquinas de Turing, una que lo acepte para toda palabra en menos de 10 pasos, y otra en más de 10 para alguna palabra. Así que no se puede aplicar el teorema de Rice.

- ERROR 2. *Es decidible. Este sería el algoritmo: simplemente, para cada entrada, se ejecuta la máquina 10 pasos. Si no se para, entonces la respuesta es no. Si para para todas, la respuesta es sí.*

  En este caso, el problema es que hay un número infinito de palabras a comprobar. Dada una máquina de Turing M, supongamos que comprobamos las palabras en el orden total dado en clase teoría (primero se mira la longitud, y si tienen la misma, lexicográficamente).

Si existe una palabra que hace que la máquina se ejecute más de 10 pasos, como tendrá una longitud finita, llegará un momento en el que se compruebe que efectivamente, no se para con sólo 10 pasos, y la respuesta para M es no. El problema viene si es una máquina que realmente se para siempre antes de 10 pasos. Este procedimiento estará comprobando palabras indefinidamente. Por lo que no es un algoritmo.

La respuesta correcta es que es decidible. En 10 pasos, sólo se pueden leer 10 casillas de la cinta como máximo. Así que, para cualquier palabra (da igual la longitud), solo me interesan los 10 primeros símbolos. Dadas dos palabras u_1 y u_2 con los 10 primeros símbolos iguales, M se para antes de 10 pasos con u_1 de entrada si, y solo si, M se para antes de 10 pasos con u_2 de entrada. Por tanto, sólo necesitamos comprobarlo para las palabras de longitud 10 o menos. Este conjunto es finito (tiene 2^{10} elementos), así que basta ejecutar 10 pasos para cada una de estas palabras. Si con todas para, entonces la respuesta es sí. Si alguna no ha parado, la respuesta es no.

Ejemplo 2.5 *Dada un máquina de Turing M, determinar si para para cualquier entrada.*

Solución. El problema no es semidecidible. Construimos una reducción desde el complementario del *Problema Universal*, cuyo lenguaje asociado es $\overline{L_u}$, el complementario del lenguaje universal. Puesto que se ha demostrado en teoría que L_u es semidecidible pero no decidible, $\overline{L_u}$ no es semidecidible. Consideramos los problemas:
C-UNIVERSAL: Dada una MT M y una palabra w, determinar si M **no** acepta w.
ALG: Dada una MT M, determinar si para para cualquier entrada.
Vamos a construir un proceso algorítmico F para pasar de una instancia de C-UNIVERSAL a una instancia de ALG. Sea (M, w) una instancia de C-UNIVERSAL, construimos las siguiente máquina de Turing $F(M, w)$ (que es una instancia de ALG)

Máquina de Turing $F(M, w)$

Entrada: Una palabra $v \in \{0,1\}^*$

1: Calculamos la longitud de v, y se almacena $|v|$ en una segunda cinta
2: Borramos v de la primera cinta y copiamos en ella $\langle M \rangle 111w$
3: Simulamos $|v|$ pasos de la máquina de Turing M con la entrada w
4: **Si** M acepta w en $|v|$ pasos **entonces**
5: entra en un bucle infinito y se **rechaza** la entrada, porque cicla
6: **Si no**
7: se **acepta** la entrada

Es claro que F es un algoritmo. Además,

- Si (M, w) es un caso positivo de C-UNIVERSAL, es decir, M **no** acepta w, entonces $F(M, w)$ acepta cualquier entrada v (ya que M no acepta w para cualquier número de pasos). Entonces $\mathcal{L}(F(M, w)) = \{0, 1\}^*$ y, como consecuencia, $F(M, w)$ para para cualquier palabra de entrada. Por lo que $F(M, w)$ es un caso positivo de ALG.

- Si (M, w) es un caso negativo de C-UNIVERSAL, entonces M acepta w. Supongamos que se acepta en n pasos. Entonces consideramos una palabra v de longitud n y, para dicha entrada, $F(M, w)$ cicla. Por tanto, existe una entrada en la que no para y $F(M, w)$ es un caso negativo de ALG.

Entonces, C-UNIVERSAL se reduce a ALG. Como C-UNIVERSAL no es semidecidible, tampoco lo es ALG.

Ejemplo 2.6 *Dada un máquina de Turing M, determinar si no acepta ningún palíndromo.*

Solución. Este problema no es semidecidible. Para demostrarlo, en vez de utilizar una reducción, estudiaremos el problema complementario. Consideramos el problema
C-PAL: Dada un máquina de Turing M, determinar si M acepta un palíndromo.
Aceptar un palíndromo es una propiedad no trivial de los lenguajes recursivamente enumerables (por ejemplo, $\{11\}$ sí la cumple pero $\{10\}$, no). Por el teorema de Rice, C-PAL no es decidible. Para demostrar que es semidecidible, utilizamos la siguiente máquina de Turing no determinista

Máquina de Turing no determinista

Entrada: Una máquina de Turing $\langle M \rangle$ (su codificación)

1: Seleccionamos de forma no determinista una palabra $w \in \{0, 1\}^*$
2: Comprobamos si w es un palíndromo
3: **Si** w es un palíndromo **entonces**
4: Simulamos M con entrada w
5: **Si** M acepta w **entonces**
6: se acepta la entrada
7: **Si no**
8: no se **acepta** la entrada
9: **Si no**
10: no se **acepta** la entrada

Una máquina de Turing determinista que acepte el mismo lenguaje podría ser la siguiente.

Máquina de Turing determinista

Entrada: Una máquina de Turing $\langle M \rangle$ (su codificación)

Para $i = 1, 2, 3, \ldots, \infty$ **hacer**
  Consideramos el conjunto A_i de la palabras de longitud menor que i (finito)
  Para cada palabra $a \in A_i$ **hacer**
    Si a es un palíndromo **entonces**
      Simulamos i pasos de M con entrada a
      Si M acepta a **entonces**
        se acepta la entrada

Ejemplo 2.7 *Dada un máquina de Turing M, determinar si termina escribiendo un 1 cuando comienza con una cinta completamente en blanco.*

Solución. En este caso, no podemos utilizar el teorema de Rice. Esta propiedad no es una propiedad de los lenguajes recursivamente enumerables. Un mismo lenguaje puede tener MT's que lo aceptan que escriban un 1 cuando empiezan con la cinta vacía, y otras que no. Construimos entonces una reducción desde el problema universal. Consideramos los siguientes problemas:

UNIVERSAL: Dada una máquina de Turing M y una palabra w, determinar si M acepta w.

ESC-1: Dada una MT M, determinar si escribe un 1 cuando la entrada es ϵ.

Vamos a construir un proceso algorítmico F para pasar de una instancia de UNIVERSAL a una instancia de ESC-1. Sea (M, w) una instancia de UNIVERSAL, construimos las siguiente máquina de Turing $F(M, w)$ (que es una instancia de ESC-1)

Máquina de Turing $F(M, w)$

Entrada: Una palabra $v \in \{0, 1\}^*$

Si v no es ϵ **entonces**
  rechaza la entrada
Si no
  Realizamos una copia M' de M donde el símbolo 1 se ha sustituido por un símbolo 1' (en el alfabeto de entrada, de la cinta, en las trasiciones, etc)
  Llamamos w' a la palabra w sustituyendo los 1 por 1'
  Escribimos $\langle M' \rangle 111 w'$
  Simulamos M' con entrada w'
  Si M' acepta w' (observad que M' acepta w' si, y solo si, M acepta w) **entonces**
    Escribe un 1

Es claro que F es un algoritmo. Además,

- Si M acepta w, entonces $F(M, w)$ escribe un 1 cuando comienza con la cinta vacía, porque M' acepta w'.
- Si M no acepta w, entonces M' no acepta w' y nunca escribe 1 (ya que $F(M, w)$ sólo puede escribir 1').

Entonces UNIVERSAL se reduce a ESC-1, por lo que ESC-1 no es decidible. Sin embargo, sí es semidecidible.

Máquina de Turing

Entrada: Una máquina de Turing M

1: **Para** $i = 1, 2, \ldots \infty$ **hacer**
2: Simular M con entrada ϵ un número i de pasos
3: **Si** M escribe un 1 en esos i pasos **entonces**
4: acepta la entrada

Ejemplo 2.8 *Dada un máquina de Turing M, determinar si acepta un número infinito de palabras.*

Solución. El problema no es semidecidible. Realizamos una reducción desde el problema diagonal. Consideramos los problemas:
DIAGONAL: Dada una máquina de Turing M, determinar si M **no** acepta $\langle M \rangle$.
INFINITY: Dada una MT M, determinar si acepta un número infinito de palabras.
Vamos a construir un proceso algorítmico F para pasar de una instancia de DIAGONAL a una instancia de INFINITY. Sea M una instancia de DIAGONAL, construimos las siguiente máquina de Turing $F(M)$:

Máquina de Turing $F(M)$

Entrada: Una palabra $v \in \{0,1\}^*$

1: **Si** v no es de la forma 0^n con $n > 0$ **entonces**
2: rechaza la entrada
3: **Si** $v = 0^n$ para cierto $n > 0$ **entonces**
4: Simulamos M con entrada $\langle M \rangle$ un número n de pasos
5: **Si** M acepta $\langle M \rangle$ **entonces**
6: se rechaza la entrada
7: **Si no**
8: se acepta la entrada

Es claro que F es un algoritmo. Además,

- si M es un caso positivo de DIAGONAL, entonces M **no** acepta $\langle M \rangle$. Calculamos $\mathcal{L}(F(M))$ el lenguaje aceptado por $F(M)$. Dada una palabra $v \in \{0,1\}^*$:
  - si $v \neq 0^n$ para algún $n > 0$, entonces la rechaza.
  - si $v = 0^n$ para algún $n > 0$, ejecuta M (con entrada $\langle M \rangle$) n pasos, y no la acepta (ya que M no acepta $\langle M \rangle$), por lo que $F(M)$ sí acepta v.

  Es decir, $\mathcal{L}(F(M)) = 0^+$, que tiene un número infinito de palabras. Luego $F(M)$ es un caso positivo de INFINITY.

- Si M es un caso negativo de DIAGONAL, entonces M acepta $\langle M \rangle$. Supongamos que lo acepta en t pasos. Calculamos $\mathcal{L}(F(M))$ el lenguaje aceptado por $F(M)$. Dada una palabra $v \in \{0,1\}^*$:
  - si $v \neq 0^n$ para algún $n > 0$, entonces la rechaza.
  - si $v = 0^n$ para algún $n < t$, ejecuta M (con entrada $\langle M \rangle$) n pasos, y no la acepta (ya que M acepta $\langle M \rangle$ en t pasos), por lo que $F(M)$ sí acepta v.
  - si $v = 0^n$ para algún $n \geq t$, ejecuta M (con entrada $\langle M \rangle$) n pasos, y sí la acepta (ya que M acepta $\langle M \rangle$ en t pasos), por lo que $F(M)$ rechaza v.

  Es decir, $\mathcal{L}(F(M)) = \{0, 0^2, \ldots, 0^{t-1}$, que tiene un número finito de palabras. Luego $F(M)$ es un caso negativo de INFINITY.

Entonces DIAGONAL se reduce a INFINITY, por lo que INFINITY no es semidecidible.