# Máquinas de Turing 2a parte

Autómatas y lenguajes formales

6 de noviembre de 2022

## 1. Reducción

Quería irme directo al teorema de Rice, pero necesitamos saber un poco de reducción. Como ya habrán visto en su camino por la facultad en las matemáticas se trata de buscar problemas similares a uno que ya se haya resuelto, ver como reducirlo al ya resuelto y no hacer más, pues si un problema puede por pasos sencillos convertirse en uno que ya resolvieron prácticamente ya está resuelto también.

Antes de resolver, si se sabe de antemano que hay un problema que busquemos por donde busquemos no tiene solución, reducir nos puede indicar problemas similares que tampoco tienen solución.

Del texto anterior recordemos las definiciones:

**Definición 1** *Un conjunto es* recursivamente enumerable *(r.e.) si el $L(M)$ para alguna máquina de Turing $M$, es decir $L(M)$ es el conjunto de cadenas aceptadas por la máquina $M$, entonces el conjunto $L$ es* recursivamente enumerable *si es aceptado por alguna máquina de Turing $M$.*

**Definición 2** *Un conjunto es* recursivo *(r.) si el $L(M_T)$ para alguna máquina de Turing total $M_T$. Una máquina de Turing total es aquella que acepta o rechaza todas las cadenas, sin nunca entrar a un ciclo infinito.*

Y una definición extra que no estaba en las notas pasadas:

**Definición 3** *Un conjunto es* co-recursivamente enumerable *(co-r.e.) si su complemento es recursivamente enumerable.*

Dado que el problema de la detención ($HP$) es un problema que sabemos es indecidible (el lenguaje no es recursivo), y hasta podríamos considerarlo el ejemplo paradigmático (aquí hablo muy informalmente), la idea es reducir el problema de la detención, que sabemos indecidible, a algún otro para ver que al igual es indecidible.

**Definición 4** *Dados los conjuntos $A \subseteq \Sigma^*$ y $B \subseteq \Delta^*$, una (muchos-uno) reducción de $A$ a $B$ es una función computable:*

$$\sigma : \Sigma^* \to \Delta^*$$

*tal que para toda $x \in \Sigma^*$:*

$$x \in A \iff \sigma(x) \in B$$

Si recuerdan en unas notas pasadas que hablamos del homomorfismo, $\sigma$ es algo similar, toda cadena en $A$ va a una cadena a $B$ bajo $\sigma$, aunque en este caso la función no requiere ser sobre o uno a uno, basta con que sea total y efectivamente computable. No considero necesario ahondar mucho en ello a esta altura, pero si no, pregunten.

Cuando se cumple lo de la definición decimos que $A$ es reducible a $B$ a través del mapeo $\sigma$, lo que para escribir en corto se pone como $A \leq_m B$. La $m$ hace referencia a muchos-uno (*many-one*), ya que no es sobre ni uno a uno, identifica a esta relación.

Un teorema que será de ayuda pronto:

**Teorema 1** *Sean $A$ y $B$ dos conjuntos, tenemos:*

1. *Si $A \leq_m B$ ($A$ es reducible a $B$) y $B$ es un conjunto recursivamente enumerable, entonces $A$ también lo es. De manera equivalente si $A \leq_m B$ y $A$ no es un conjunto recursivamente enumerable, entonces $B$ tampoco lo es.*

2. *Si $A \leq_m B$ y $B$ es un conjunto recursivo, entonces $A$ también lo es. De manera equivalente si $A \leq_m B$ y $A$ no es un conjunto recursivo, entonces $B$ tampoco lo es.*

Usando lo visto hasta ahora para un ejemplo, sea el conjunto

$$FIN = \{M | L(M) \text{ es finito}\} \tag{1}$$

Ni este conjunto ni su complemento son recursivamente enumerables. Como se imaginarán la idea es reducir el problema de la detención a ambos conjuntos, aunque ahora nos conviene reducir su complemento, que se escribe $\sim HP$, dado que H.P. no es recursivo pero sí recursivamente enumerable, su complemento no es recursivamente enumerable. Tenemos

$$\sim HP = \{M\#x \mid M \text{ no se detiene con } x\}$$

Que se reducirá a ambos conjuntos

1. $\sim HP \leq_m FIN$,
2. $\sim HP \leq_m \sim FIN$

De tener estas reducciones, por el teorema arriba mencionado ya tendríamos la demostración. Sabemos que $HP$ no es recursivo,y aún peor, $\sim HP$, su complemento, ni siquiera es recursivamente enumerable. Ello implicaría que ni $FIN$ ni su complemento serían recursivamente enumerables.

Vamos a construir un mapeo que vaya de las cadenas aceptadas por $\sim HP$, es decir, de $M\#x$ a cadenas en $FIN$, que las vamos a denotar como $\sigma(M\#x)$, de esta forma construiremos la máquina de Turing a la que le daremos como entrada el conjunto finito, es decir:

$$M' = \sigma(M\#x)$$
$$\text{Si } M \text{ no se detiene con } x \iff L(M') \text{ es finito}$$

Dada $M\#x$ construimos $M'$ tal que al darle como entrada $y$ sucede:

1. borra la entrada $y$
2. escribe en la cinta la cadena $x$ ($x$ ya está guardada en la máquina $M'$, solo es traída de memoria y se pone en la cinta para manipularla)
3. corre la máquina $M$ (también guardada en alguna parte de la memoria de $M'$) con entrada $x$
4. la cadena $y$ es aceptada si $M$ se detiene con $x$.

Entonces si en el paso tres la máquina $M$ no se detiene con $x$, la máquina $M'$ tampoco se detiene, no sigue al paso 4, no acepta $y$, sea lo que sea $y$, el conjunto que acepta es el vacío. Si por el contrario la máquina $M$ se detiene con $x$ la máquina $M'$ sigue al paso cuatro y acepta $y$ (no importa que sea $y$, ya hasta la borramos y aún así la acepta). Como acepta cualquier $y$ pues entonces es un conjunto infinito de cadenas ($y \in \Sigma^*$). Resumiendo:

$$\begin{aligned} M \text{ se detiene con } x &\implies L(M') = \Sigma^* \implies L(M') \text{ es infinito} \\ M \text{ no se detiene con } x &\implies L(M') = \emptyset \implies L(M') \text{ es finito} \end{aligned}$$

Como pueden ver $M'$ depende del problema de la detención, que la máquina $M$ se detenga con la cadena $x$, como ese problema no es recursivamente enumerable, entonces el problema de saber si la máquina que acepta las cadenas de un lenguaje finito tampoco es recursivamente enumerable.

Lo que hicimos fue construir una máquina que reducía el complemento del problema de detención en el problema de aceptar cadenas finitas $\sim HP \leq_m FIN$ (cuando $M$ no se detiene entonces $M'$ acepta cadenas de un lenguaje finito, en particular el conjunto vacío). Por lo tanto ya que el complemento del problema de detención, que no es recursivamente enumerable, se reduce al problema dado, entonces este tampoco es recursivamente enumerable.

Algo similar s hace para $\sim FIN$. Ahora se reduce $\sim$ **HP** a $\sim$ **FIN**, que es equivalente a reducir[1] **HP** a **FIN** a partir de un mapeo $\tau$ tal que:

$$M\#x \in \textbf{HP} \iff \tau(M\#x) \in \textbf{FIN}$$

Con $M\#x$ construimos $M''$ que en la entrada $y$:

1. Escribe $y$ en una segunda cinta
2. Escribe $x$ en la cinta original
3. Corre $M$ con $x$ por $|y|$ pasos
4. Si $M$ no se ha detenido en esos pasos acepta, de otra forma rechaza

Así **HP** se reduce a **FIN**, sacamos el complemento y tenemos lo que buscábamos.

[1] Es equivalente operativamente, pero los resultados deben analizarse con el complemento. Es decir, esta reducción nos dice que $\mathbb{FIN}$ no es recursivo, pero lo que queremos demostrar es que no es recursivamente enumerable, para eso necesitamos ver los complementos.

# Teorema de Rice

En cada conjunto *recursivamente enumerable* podemos definir una propiedad

**Definición 5** *Una propiedad de un conjunto* recursivamente enumerable *es un mapeo del tipo:*

$$P : \{subconjuntos\ r.e.\ de\ \Sigma^*\} \to \{\top, \bot\},$$

Donde $\top$ se refiere a verdadero y $\bot$ a falso, asignaciones lógicas. Un ejemplo de estas propiedades sería:

$$P(A) = \begin{cases} \top & \text{si } A = \emptyset \\ \bot & \text{si } A \neq \emptyset \end{cases}$$

Es decir, saber si un conjunto (no una máquina) es vacío, por ejemplo, si yo digo que $A$ es el conjunto recursivamente enumerable (que lo puedo meter a una máquina de Turing y acepta o rechaza o entra en un ciclo infinito) de las cadenas de longitud $n$, esta propiedad me diría si tal conjunto es vacío o no.

**Teorema 2** *(Teorema de Rice) Toda propiedad no trivial de un conjunto* recursivamente enumerable *es indecidible.*

Para poder inspeccionar estas propiedades lo hacemos a través de una máquina de Turing, las propiedades deben estar escritas de tal forma que sean una cadena aceptada por una máquina.

Pero una parte importante es saber qué es trivial y qué no lo es. En este caso no trivial es que la propiedad no sea universalmente falsa ni universalmente verdadera, que al menos existe un elemento que cumple la propiedad y al menos uno que no la cumple.

Para entender un poco mejor esto veamos la demostración:

**Demostración 1** *Sea $P$ una propiedad no trivial de un conjunto recursivamente enumerable, supongamos, sin perder generalidad, que esta propiedad es falsa para un conjunto vacío, $P(\emptyset) = \bot$ (bien podría ser cierta e igual se puede continuar la demostración, esta decisión es, como se imaginarán, para tener definido el caso base). Como ya dijimos que $P$ es no trivial entonces existe al menos un conjunto $A$ para el cual la propiedad es verdadera, $P(A) = \top$. Sea $K$ la máquina de Turing que acepta el conjunto $A$.*

*Como ya lo hemos hecho antes, vamos a apoyarnos del problema más famoso de indecibilidad, el problema de la detención (halting problem HP), para ver que esta propiedad tampoco es decidible. Vamos a escribir HP de forma reducida como $\{M|P(L(M)) = T\}$, es decir, todas las máquinas $M$ tales que su conjunto recursivamente enumerable asociado tiene dicha propiedad como verdadera.*

*Construimos una máquina $M'$ relacionada a la máquina $M$ de la forma $M' = \sigma(M\#x)$ ($M'$ es una variante de $M$ a la que además se le ha anexado una cadena $x$, en la definición de $M'$ de incluye, de alguna manera, la definición de $M$ y una cadena estática $x$).*

*Hasta aquí ya llevamos tres máquinas definidas, $K$ que al momento no hemos dicho como va a usarse, $M$ que son las máquinas que aceptan como lenguaje el conjunto para el cual la propiedad es verdadera, y $M'$ la máquina definida*

*a partir de una $x$ y la descripción de la máquina $M$, desbaratemos un poco el nudo viendo más de la construcción de $M'$.*

*Cuando a la máquina $M'$ se le da como entrada la cadena y sucede lo siguiente:*

1. *guarda $y$ en algún lugar separado de la cinta*
2. *escribe $x$ en la cinta ($x$ ya estaba guardada en algún lugar de la máquina, por ejemplo una cinta e sólo lectura, en este paso es puesta en la cinta en la que se puede manipular)*
3. *corre $M$ para la entrada $x$ (igual, $M$ es traída desde algún lugar de la memoria de la máquina $M'$, esto no se le da en la entrada)*
4. *si $M$ se detiene en $x$ (recuerden que es una descripción de H.P.) ahora $M'$ corre $K$ con $y$ como entrada y acepta sí $K$ acepta.*

*Aquí el paso crucial es el tercero, $M$ puede detenerse o no con $x$ (si $x$ no cumple la propiedad la máquina no se detiene). Si $M$ no se detiene con $x$ en el paso cuatro $M'$ rechaza la entrada y (no tiene que correr a $K$, sólo rechaza). En cambio, si $M$ se detiene con $x$ se pasa al paso cuatro, se le da $y$ como entrada a $K$, esta $y$ es aceptada por $M'$ si es aceptada por $K$.*

*Las opciones se ven así:*

$$M \text{ se detiene con } x \implies L(M') = A \implies P(L(M')) = P(A) = \top$$
$$M \text{ no se detiene con } x \implies L(M') = \emptyset \implies P(L(M')) = P(\emptyset) = \bot$$

*Como en el ejemplo de la sección anterior hemos reducido el problema de la aceptación a este problema, es decir $HP \leq_m \{M \mid P(L(M)) = \top\}$, por el teorema de la sección anterior entonces $\{M \mid P(L(M)) = \top\}$ tampoco es recursivo, entonces el lenguaje que satisface la propiedad $P$ tampoco es recursivo (pero si es recursivamente enumerable, eso ya lo tenia por default), por ende tampoco es decidible.*

Pero hay una parte dos de este teorema, porque el primero fue un éxito en taquilla... no, porque es una cosa un poco más específica, para propiedades monótonas.

**Definición 6** *Una propiedad $P : \{\text{Conjuntos r.e.}\} \to \{\top, \bot\}$ de los conjuntos recursivamente enumerables es monótona si para todos los conjuntos r.e. $A$ y $B$, si $A \subseteq B$ entonces $P(A) \leq P(B)$, donde el orden se especifica como $\bot \leq \top$. Para ponerlo menos revuelto, todo subconjunto de un conjunto más grande hereda sus propiedades.*

Por ejemplo $FIN$ que vimos en la sección anterior es monótono, todo subconjunto es finito también. Vamos al teorema.

**Teorema 3** *Ninguna propiedad monótona de un conjunto recursivamente enumerable (r.e.) es semidecidible. Es decir, si $P$ es una propiedad no-monótona de conjuntos r.e., entonces el conjunto $T_P = \{M \mid P(L(M)) = \top\}$ no es recursivamente enumerable.*

Ya de ese más o menos pueden hacer la demostración.

## Jerarquía aritmética

Hasta el momento hemos encontrado un problema paradigmático para establecer los limites de lo que es computable y lo que no, el problema de la detención (H.P. por sus siglas en inglés). A partir de él podemos catalogar el resto de problemas que nos encontremos si H.P. o su complemento, $\tilde{\text{H}}$.P. puede reducirse a ese otro problema.

Decimos que un problema $B$ es Turing reducible a $A$, y lo escribimos $A \leq_T B$. De acuerdo a esa relación podemos clasificar los tipos de lenguaje como:

- $\Sigma_1^0 = \{$ lenguajes recursivamente enumerables$\}$
- $\Delta_1^0 = \{$ lenguajes recursivos$\}$
- $\Sigma_{n+1}^0 = \{$lenguajes recursivamente enumerables en algún $L \in \Sigma_n^0\}$
- $\Delta_{n+1}^0 = \{$lenguajes recursivos en algún $L \in \Delta_n^0\}$
- $\Pi_n^0 = \{$complemento de lenguajes en $\Sigma_n^0\}$

Esto se puede verbalizar en forma sintáctica como:

*Un lenguaje es recursivamente enumerable si y sólo si existe una propiedad $R$ decidible de pares de cadenas tal que $L = \{\alpha | \exists \beta$ tal que $R(\alpha, \beta)\}$.*

De esta forma se puede escribir nuestro problema paradigmático como:

$$HP = \{\langle M \# \alpha \rangle | \exists x. M \text{ se detiene con } \alpha \text{ en } x \text{ pasos}\}$$

Y el problema de la pertenencia:

$$MP = \{\langle M \# \alpha \rangle | \exists x. M \text{ acepta a } \alpha \text{ en } x \text{ pasos}\}$$

Sabemos que ambos problemas son recursivamente enumerables, en este caso podemos identificarlo con sólo escribir el enunciado de la forma mencionada:

- Un lenguaje $L$ está en $\Sigma_n^0$ si y sólo si existe una propiedad $R$ $(n+1)$-aria[2] decidible tal que

$$L = \{\alpha | \exists \beta_1. \forall \beta_2 ... \beta_n R(\alpha, \beta_1 ..., \beta_n)\}$$

- Un lenguaje $L$ está en $\Pi_n^0$ si y sólo si existe una propiedad $R$ $(n+1)$-aria decidible tal que

$$L = \{\alpha | \forall \beta_1. \exists \beta_2 ... \beta_n R(\alpha, \beta_1 ..., \beta_n)\}$$

- Finalmente $\Delta_n^0 = \Sigma_n^0 \cap \Pi_n^0$

De esta forma, todos los problemas que son recursivamente enumerables pueden escribirse de la primera forma, y los que no son ni siquiera recursivamente enumerables, de la segunda forma.

[2]Es decir, una propiedad es binaria si sólo es para un miembro en relación a la cadena mencionada: $R(\alpha, \beta)$. Sería terciaria si s relación a otros dos objetos: $R(\alpha, \beta, \gamma)$, etc.

# Referencias

[1] Kozen, Dexter C. “Automata and Computability” Springer (1997)

[2] Sipser, Michael “Introduction to the Theory of Computation” 2a ed., Thomson Course Tecnology (2006)