Lenguajes Recursivamente Enumerables

Teoría de la Computación

Grado en Ingeniería Informática

Contenidos

Contenidos

1 Lenguajes aceptados por máquinas de Turing

Anteriormente hemos introducido las MT,s y hemos estudiado algunas de sus propiedades.

Más concretamente, la existencia de una **MT Universal** sugiere que cualquier MT puede verse como un programa de ordenador y, por tanto, las MT,s en general podrían modelizar los mecanismos de la computación.

1 Lenguajes aceptados por máquinas de Turing

Anteriormente hemos introducido las MT,s y hemos estudiado algunas de sus propiedades.

Más concretamente, la existencia de una **MT Universal** sugiere que cualquier MT puede verse como un programa de ordenador y, por tanto, las MT,s en general podrían modelizar los mecanismos de la computación.

Más adelante, estudiaremos en profundidad esta conexión, hasta lograr definir los límites de un modelo computacional, mediante el estudio de la **resolubilidad**.

1 Lenguajes aceptados por máquinas de Turing

Anteriormente hemos introducido las MT,s y hemos estudiado algunas de sus propiedades.

Más concretamente, la existencia de una **MT Universal** sugiere que cualquier MT puede verse como un programa de ordenador y, por tanto, las MT,s en general podrían modelizar los mecanismos de la computación.

Más adelante, estudiaremos en profundidad esta conexión, hasta lograr definir los límites de un modelo computacional, mediante el estudio de la **resolubilidad**.

Pero, por el momento, en este capítulo, estudiaremos las clases de lenguajes aceptados por las MT,s e introduciremos el **problema de la parada**, un problema general que afecta a cualquier MT y que será la clave para tratar la resolubilidad.

Y finalmente, veremos el **teorema de la jerarquía**, que relaciona entre sí todos los tipos de lenguajes estudiados.

1 Lenguajes aceptados por máquinas de Turing

Recordemos que una cadena w es aceptada por una MT cuando una computación, que empieza con w en la cinta y con la MT en el estado inicial, termina en un estado final.

1 Lenguajes aceptados por máquinas de Turing

Recordemos que una cadena w es aceptada por una MT cuando una computación, que empieza con w en la cinta y con la MT en el estado inicial, termina en un estado final.

Por otro lado, una cadena puede ser rechazada de dos formas:

- Porque al procesarla la máquina para en un estado no final.
- O porque la máquina nunca para.

1 Lenguajes aceptados por máquinas de Turing

Recordemos que una cadena w es aceptada por una MT cuando una computación, que empieza con w en la cinta y con la MT en el estado inicial, termina en un estado final.

Por otro lado, una cadena puede ser rechazada de dos formas:

- Porque al procesarla la máquina para en un estado no final.
- O porque la máquina nunca para.

Los dos métodos de rechazo no son equivalentes y daban lugar a los siguientes tipos de lenguajes:

- Los **lenguajes recursivamente enumerables**, que son aquéllos aceptados de alguna forma por una MT.
- Y, dentro de ellos, los **lenguajes recursivos**, que son aquéllos aceptados por al menos una MT que para con cualquier cadena de entrada, independientemente de si la cadena pertenece o no al lenguaje de la máquina.

1 Lenguajes aceptados por máquinas de Turing

Recordemos que una cadena w es aceptada por una MT cuando una computación, que empieza con w en la cinta y con la MT en el estado inicial, termina en un estado final.

Por otro lado, una cadena puede ser rechazada de dos formas:

- Porque al procesarla la máquina para en un estado no final.
- O porque la máquina nunca para.

Los dos métodos de rechazo no son equivalentes y daban lugar a los siguientes tipos de lenguajes:

- Los **lenguajes recursivamente enumerables**, que son aquéllos aceptados de alguna forma por una MT.
- Y, dentro de ellos, los **lenguajes recursivos**, que son aquéllos aceptados por al menos una MT que para con cualquier cadena de entrada, independientemente de si la cadena pertenece o no al lenguaje de la máquina.

Por tanto, todo lenguaje recursivo es también recursivamente enumerable, y veremos que esta inclusión es propia. Es decir, efectivamente existen lenguajes recursivamente enumerables que no son recursivos.

Contenidos

2 Lengs. regulares e indep. del contexto como recursivos

Sea $M = (Q, \Sigma, s, F, \delta)$ un AFD. Podemos construir $M' = (Q', \Sigma', \Gamma, s', B, F', \delta')$, una MT tal que $L(M) = L(M')$, como sigue:

$$Q' = Q \cup \{q'\} \qquad \Sigma' = \Sigma \qquad \Gamma = \Sigma \cup \{B\} \qquad s' = s \qquad F' = \{q'\}$$

$$\delta'(q, \sigma) = (\delta(q, \sigma), \sigma, R),\ \forall q \in Q,\ \forall \sigma \in \Sigma'$$

$$\delta'(q, B) = (q', B, S),\ \forall q \in F$$

Esta MT analiza las cadenas de entrada de izquierda a derecha reflejando los correspondientes cálculos del AFD y, cuando encuentra un blanco al final de la cadena, pasa al estado de aceptación sólo si el autómata acepta también la cadena.

2 Lengs. regulares e indep. del contexto como recursivos

Sea $M = (Q, \Sigma, s, F, \delta)$ un AFD. Podemos construir $M' = (Q', \Sigma', \Gamma, s', B, F', \delta')$, una MT tal que $L(M) = L(M')$, como sigue:

$$Q' = Q \cup \{q'\} \qquad \Sigma' = \Sigma \qquad \Gamma = \Sigma \cup \{B\} \qquad s' = s \qquad F' = \{q'\}$$

$$\delta'(q, \sigma) = (\delta(q, \sigma), \sigma, R),\ \forall q \in Q,\ \forall \sigma \in \Sigma'$$

$$\delta'(q, B) = (q', B, S),\ \forall q \in F$$

Esta MT analiza las cadenas de entrada de izquierda a derecha reflejando los correspondientes cálculos del AFD y, cuando encuentra un blanco al final de la cadena, pasa al estado de aceptación sólo si el autómata acepta también la cadena.

Así pues, dado que todo lenguaje regular puede ser aceptado por un AFD, se verifica el siguiente resultado.

Teorema

Si L es un lenguaje regular, entonces L es también un lenguaje recursivo. □

2 Lengs. regulares e indep. del contexto como recursivos

Sabemos que $L = \{a^n b^n \mid n \geq 0\}$ no es regular, pero sabemos que se puede construir una MT que acepta L y que además para con cualquier cadena de entrada. Por tanto, hay lenguajes recursivos que no son regulares, como por ejemplo algunos LIC,s tales como L.

2 Lengs. regulares e indep. del contexto como recursivos

Sabemos que $L = \{a^n b^n \mid n \geq 0\}$ no es regular, pero sabemos que se puede construir una MT que acepta L y que además para con cualquier cadena de entrada. Por tanto, hay lenguajes recursivos que no son regulares, como por ejemplo algunos LIC,s tales como L.

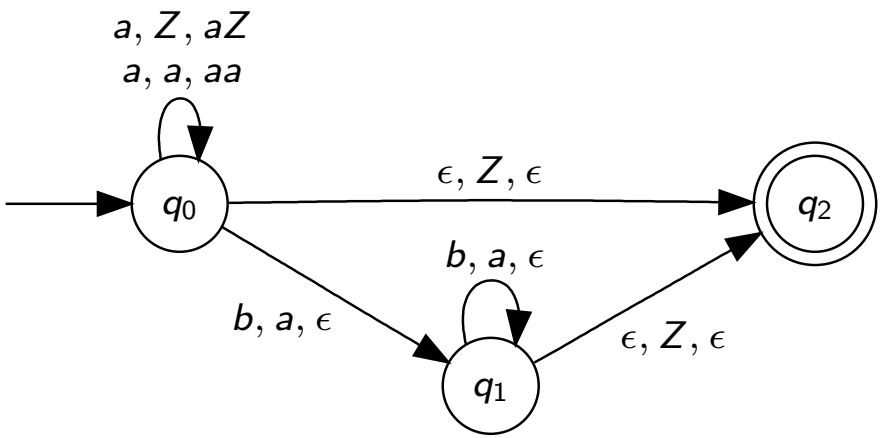
Es más, igual que podemos construir una MT equivalente a un AFD, podemos también construir una MT equivalente a un APN. Aquí lo ilustraremos con un ejemplo para este lenguaje L, aunque por supuesto este proceso se puede generalizar para cualquier APN.

2 Lengs. regulares e indep. del contexto como recursivos

Sabemos que $L = \{a^n b^n \mid n \geq 0\}$ no es regular, pero sabemos que se puede construir una MT que acepta L y que además para con cualquier cadena de entrada. Por tanto, hay lenguajes recursivos que no son regulares, como por ejemplo algunos LIC,s tales como L.

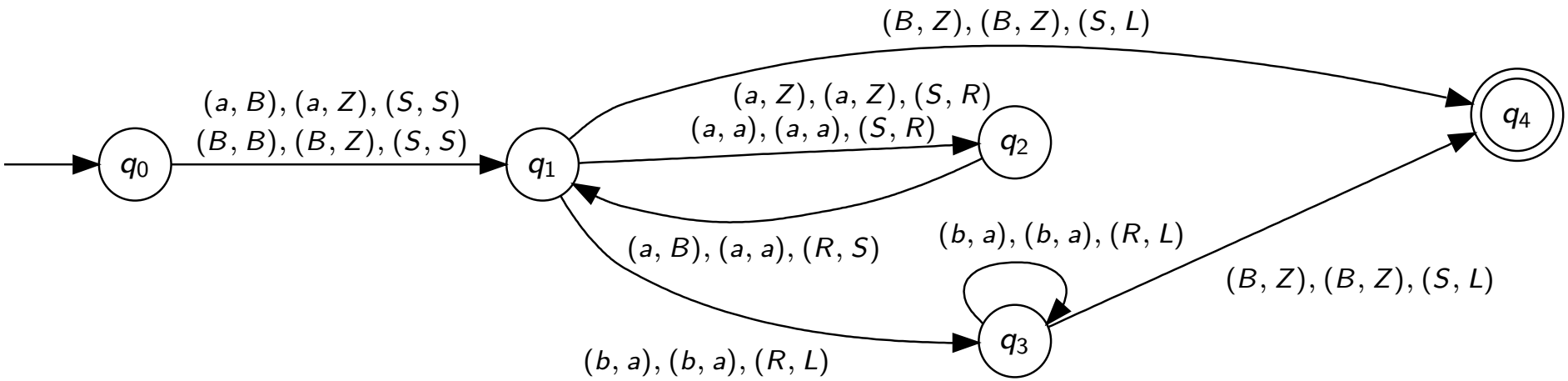
Es más, igual que podemos construir una MT equivalente a un AFD, podemos también construir una MT equivalente a un APN. Aquí lo ilustraremos con un ejemplo para este lenguaje L, aunque por supuesto este proceso se puede generalizar para cualquier APN.

Para ello, consideremos entonces el siguente APN, el cual acepta precisamente nuestro lenguaje $L = \{a^n b^n \mid n \geq 0\}$:



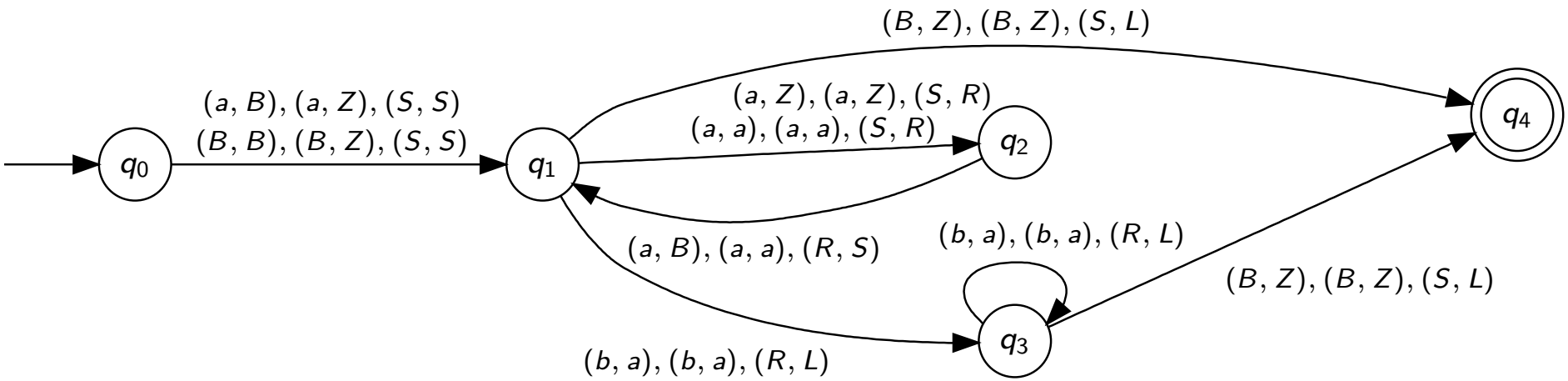
2 Lengs. regulares e indep. del contexto como recursivos

Para simplificar la conversión, utilizamos una MT de dos cintas: la primera almacenará la cadena de entrada y la segunda simulará las operaciones de la pila del APN.



2 Lengs. regulares e indep. del contexto como recursivos

Para simplificar la conversión, utilizamos una MT de dos cintas: la primera almacenará la cadena de entrada y la segunda simulará las operaciones de la pila del APN.



Así pues, dado que todo lenguaje independiente del contexto es aceptado por un APN, se verifica también el siguiente resultado.

Teorema

Si L es un lenguaje independiente del contexto, entonces L es también un lenguaje recursivo. □

Contenidos

3 Propiedades de los lengs. recursivos y rec. enumerables

Teorema

Si L_1 y L_2 son lenguajes recursivos, entonces $L_1 \cap L_2$ también lo es.

Teorema

Si L_1 y L_2 son lenguajes recursivos, entonces $L_1 \cap L_2$ también lo es.

Demostración:

Consideremos M_1 y M_2, dos MT,s que paran con cualquier cadena de entrada, y tales que $L(M_1) = L_1$ y $L(M_2) = L_2$.

Al procesar una cadena w en M_1, si $w \in L_1$, M_1 parará en un estado de aceptación, y si $w \notin L_1$, M_1 parará en un estado de no aceptación. En la máquina M_2, ocurrirá lo mismo.

Así pues, se puede construir una MT M de dos cintas, de forma que:

- La cadena de entrada w estará en la primera cinta y se hará una copia de ella también sobre la segunda cinta.
- M simulará a M_1 con w sobre la primera cinta, hasta que M_1 pare.
- Si M_1 para en un estado no final, M hará lo mismo.
- Si M_1 para en un estado final, M simulará a M_2 con w sobre la segunda cinta.
- Si M_2 para en un estado no final, M hará lo mismo.
- Y si M_2 para en un estado final, M también.

Por tanto, $w \in L(M)$ si y sólo si $w \in L(M_1)$ y $w \in L(M_2)$.
Es decir, $L(M) = L(M_1) \cap L(M_2)$ y por tanto $L_1 \cap L_2$ es un lenguaje recursivo. □

3 Propiedades de los lengs. recursivos y rec. enumerables

Llegados a este punto, es necesario recordar que $L = \{a^i b^i c^i \mid i \geq 0\}$ es la intersección de dos LIC,s aunque él mismo no es un LIC (tal y como demostramos en su momento mediante el lema del bombeo para LIC,s).

Pero por medio de los dos teoremas anteriores se obtiene que L sí es un lenguaje recursivo. Por tanto, existen efectivamente más lenguajes recursivos que los regulares y los LIC,s.

3 Propiedades de los lengs. recursivos y rec. enumerables

Llegados a este punto, es necesario recordar que $L = \{a^i b^i c^i \mid i \geq 0\}$ es la intersección de dos LIC,s aunque él mismo no es un LIC (tal y como demostramos en su momento mediante el lema del bombeo para LIC,s).

Pero por medio de los dos teoremas anteriores se obtiene que L sí es un lenguaje recursivo. Por tanto, existen efectivamente más lenguajes recursivos que los regulares y los LIC,s.

Ejercicio

Demuestre que si L_1 y L_2 son lenguajes recursivos, entonces $L_1 \cup L_2$ también lo es.

3 Propiedades de los lengs. recursivos y rec. enumerables

Llegados a este punto, es necesario recordar que $L = \{a^i b^i c^i \mid i \geq 0\}$ es la intersección de dos LIC,s aunque él mismo no es un LIC (tal y como demostramos en su momento mediante el lema del bombeo para LIC,s).

Pero por medio de los dos teoremas anteriores se obtiene que L sí es un lenguaje recursivo. Por tanto, existen efectivamente más lenguajes recursivos que los regulares y los LIC,s.

Ejercicio

Demuestre que si L_1 y L_2 son lenguajes recursivos, entonces $L_1 \cup L_2$ también lo es.

Solución:

Dadas M_1 y M_2, dos MT,s que paran con cualquier cadena de entrada, y tales que $L(M_1) = L_1$ y $L(M_2) = L_2$, construimos una nueva MT M de dos cintas, de forma que:

- La cadena de entrada w se copia en ambas cintas.
- M simulará a M_1 con w sobre la primera cinta, hasta que M_1 pare.
- Si M_1 para en un estado final, M hará lo mismo.
- Si M_1 para en un estado no final, M simulará a M_2 con w sobre la segunda cinta.
- Si M_2 para en un estado final, M hará lo mismo.
- Y si M_2 para en un estado no final, M también.

Por tanto, $w \in L(M)$ si y sólo si $w \in L(M_1)$ o bien $w \in L(M_2)$.
Es decir, $L(M) = L(M_1) \cup L(M_2)$ y por tanto $L_1 \cup L_2$ es un lenguaje recursivo.

3 Propiedades de los lengs. recursivos y rec. enumerables

Ejercicio

Demuestre que si L_1 y L_2 son lenguajes rec. enumerables, entonces $L_1 \cap L_2$ también lo es.

Ejercicio

Demuestre que si L_1 y L_2 son lenguajes rec. enumerables, entonces $L_1 \cap L_2$ también lo es.

Solución:

Dadas M_1 y M_2, dos MT,s tales que $L(M_1) = L_1$ y $L(M_2) = L_2$, construimos una nueva MT M de dos cintas, la cual simulará a M_1 sobre la primera cinta y a M_2 sobre la segunda cinta.

La cadena de entrada w se copiará en ambas cintas. Si M_1 o M_2 no paran sobre w, M tampoco lo hará. Por tanto, $w \in L(M)$ sólo cuando M_1 y M_2 paren con w en un estado de aceptación.

3 Propiedades de los lengs. recursivos y rec. enumerables

Ejercicio

Demuestre que si L_1 y L_2 son lenguajes rec. enumerables, entonces $L_1 \cap L_2$ también lo es.

Solución:

Dadas M_1 y M_2, dos MT,s tales que $L(M_1) = L_1$ y $L(M_2) = L_2$, construimos una nueva MT M de dos cintas, la cual simulará a M_1 sobre la primera cinta y a M_2 sobre la segunda cinta.

La cadena de entrada w se copiará en ambas cintas. Si M_1 o M_2 no paran sobre w, M tampoco lo hará. Por tanto, $w \in L(M)$ sólo cuando M_1 y M_2 paren con w en un estado de aceptación.

Ejercicio

Demuestre que si L_1 y L_2 son lenguajes rec. enumerables, entonces $L_1 \cup L_2$ también lo es.

3 Propiedades de los lengs. recursivos y rec. enumerables

Ejercicio

Demuestre que si L_1 y L_2 son lenguajes rec. enumerables, entonces $L_1 \cap L_2$ también lo es.

Solución:

Dadas M_1 y M_2, dos MT,s tales que $L(M_1) = L_1$ y $L(M_2) = L_2$, construimos una nueva MT M de dos cintas, la cual simulará a M_1 sobre la primera cinta y a M_2 sobre la segunda cinta.

La cadena de entrada w se copiará en ambas cintas. Si M_1 o M_2 no paran sobre w, M tampoco lo hará. Por tanto, $w \in L(M)$ sólo cuando M_1 y M_2 paren con w en un estado de aceptación.

Ejercicio

Demuestre que si L_1 y L_2 son lenguajes rec. enumerables, entonces $L_1 \cup L_2$ también lo es.

Solución:

Dadas M_1 y M_2, dos MT,s tales que $L(M_1) = L_1$ y $L(M_2) = L_2$, construimos una nueva MT M de dos cintas, para simular a M_1 y a M_2, y copiamos la cadena de entrada w en ambas cintas.

Sin embargo, M_1 podría no parar con w (con lo cual M tampoco pararía) y M_2 podría sí parar (con lo cual M debería parar con w y aceptarla). Por tanto, en este caso, la simulación de M_1 y M_2 por parte de M debe ser **simultánea**. Esto se consigue incorporando en M el producto cartesiano de los estados y de las transiciones de M_1 y M_2, además de unas transiciones adicionales que permitan seguir simulando a una MT cuando la otra pare en un estado no final, y otras que permitan aceptar w cuando cualquiera de ellas pare en un estado final.

3 Propiedades de los lengs. recursivos y rec. enumerables

Teorema

Si L es un lenguaje recursivo, $\Sigma^* - L$ es también un lenguaje recursivo.

Teorema

Si L es un lenguaje recursivo, $\Sigma^* - L$ es también un lenguaje recursivo.

Demostración:

Si L es recursivo, entonces existe una MT $M = (Q, \Sigma, \Gamma, s, B, F, \delta)$ que para con cualquier cadena de entrada, y tal que $L(M) = L$.

Por tanto, la MT que acepta $\Sigma^* - L$ y que también para con cualquier cadena de entrada es simplemente $M' = (Q, \Sigma, \Gamma, s, B, Q - F, \delta)$.

M' rechaza todas las cadenas de L (es decir, aquéllas con las que M para en un estado final) y acepta todas las que no están en L (es decir, aquéllas con las que M para en un estado no final).

Teorema

Si L es un lenguaje recursivo, $\Sigma^* - L$ es también un lenguaje recursivo.

Demostración:

Si L es recursivo, entonces existe una MT $M = (Q, \Sigma, \Gamma, s, B, F, \delta)$ que para con cualquier cadena de entrada, y tal que $L(M) = L$.

Por tanto, la MT que acepta $\Sigma^* - L$ y que también para con cualquier cadena de entrada es simplemente $M' = (Q, \Sigma, \Gamma, s, B, Q - F, \delta)$.

M' rechaza todas las cadenas de L (es decir, aquéllas con las que M para en un estado final) y acepta todas las que no están en L (es decir, aquéllas con las que M para en un estado no final).

Sin embargo, ésta es una propiedad que, en general, no se cumple para los lenguajes recursivamente enumerables.

Para verlo, supongamos un alfabeto Σ. Dado que Σ es finito, Σ^* es numerable y podemos enumerarlo como sigue: $\Sigma^* = \{w_1, w_2, w_3, \dots\}$.

Y al hablar de la MT Universal, vimos también que todas las posibles MT,s sobre un alfabeto Σ podían ser enumeradas por medio de las cadenas de ceros y unos que representan a esas mismas MT,s codificadas, y que finalmente se pueden asociar a números binarios y, por tanto, a un subconjunto de $\mathbb{N}$. Es decir, podemos también listar todas las máquinas de Turing sobre un alfabeto Σ como sigue: $M_1, M_2, M_3, \dots$

3 Propiedades de los lengs. recursivos y rec. enumerables

Consideremos entonces el siguiente lenguaje:

$$L = \{w_i \mid w_i \text{ es aceptado por } M_i\}$$

Este lenguaje es recursivamente enumerable, pero su complementario no.

3 Propiedades de los lengs. recursivos y rec. enumerables

Consideremos entonces el siguiente lenguaje:

$$L = \{w_i \mid w_i \text{ es aceptado por } M_i\}$$

Este lenguaje es recursivamente enumerable, pero su complementario no.

Para demostrarlo, primero veremos que L es recursivamente enumerable.

Para ello, obtendremos una MT M que acepte L, mediante la composición de varias máquinas:

- Dada una cadena de entrada w, M generará $w_1, w_2, w_3, \ldots$, hasta encontrar un i para el cual $w_i = w$.

  Entonces generará la codificación de la i-ésima MT M_i, codificará w_i, y pasará M_i y w_i a M_U, la MT Universal, la cual simulará a M_i sobre w_i.
- Si M_i para y acepta w_i, entonces M_U para en un estado de aceptación, y por tanto M para y acepta w.
- Por otro lado, M_i puede parar y no aceptar w_i, o bien no parar. En ambos casos, M no para y no acepta w.
- Así pues, $w_i \in L(M)$ si y sólo si $w_i \in L(M_i)$.

3 Propiedades de los lengs. recursivos y rec. enumerables

En segundo lugar, veremos que $\Sigma^* - L$ no es recursivamente enumerable.

Supongamos que sí lo es. Entonces debería ser aceptado por una MT que llamaremos M_j. Consideremos entonces qué ocurre con la cadena w_j:

- Si $w_j \in L(M_j)$, entonces se cumple también que $w_j \in L$, con lo cual w_j estaría tanto en L como en $\Sigma^* - L$, y esto es una contradicción.
- Y si $w_j \notin L(M_j)$, entonces se cumple también que $w_j \notin L$, y por tanto w_j no estaría ni en L ni en $\Sigma^* - L$, lo cual también es una contradicción porque w_j es una cadena de Σ^*.

En ambos casos se llega a una contradicción. Por tanto, $\Sigma^* - L$ no es aceptado por ninguna MT, y por tanto no es recursivamente enumerable.

En segundo lugar, veremos que $\Sigma^* - L$ no es recursivamente enumerable.

Supongamos que sí lo es. Entonces debería ser aceptado por una MT que llamaremos M_j. Consideremos entonces qué ocurre con la cadena w_j:

- Si $w_j \in L(M_j)$, entonces se cumple también que $w_j \in L$, con lo cual w_j estaría tanto en L como en $\Sigma^* - L$, y esto es una contradicción.
- Y si $w_j \notin L(M_j)$, entonces se cumple también que $w_j \notin L$, y por tanto w_j no estaría ni en L ni en $\Sigma^* - L$, lo cual también es una contradicción porque w_j es una cadena de Σ^*.

En ambos casos se llega a una contradicción. Por tanto, $\Sigma^* - L$ no es aceptado por ninguna MT, y por tanto no es recursivamente enumerable.

Esto da lugar al siguiente resultado.

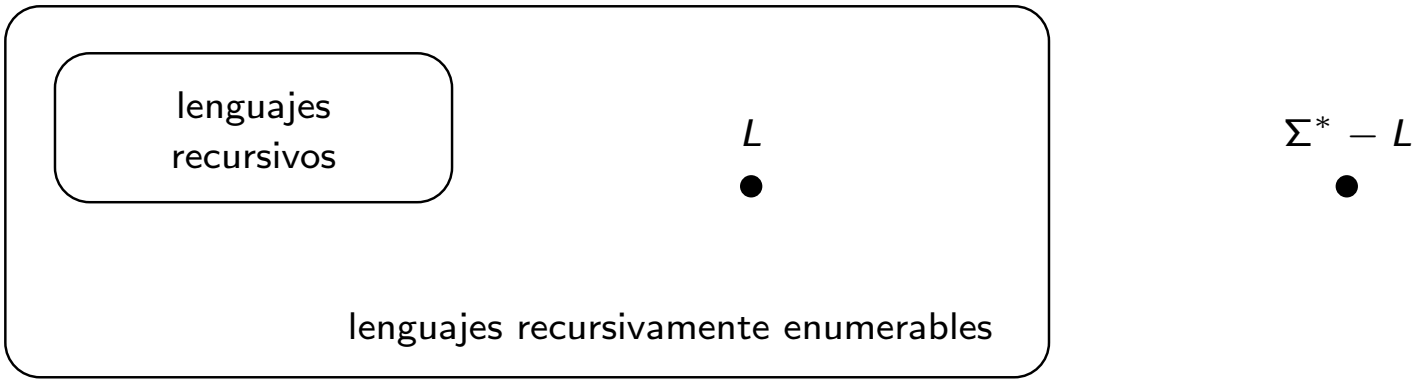
Teorema

Existe un lenguaje recursivamente enumerable L para el cual $\Sigma^* - L$ no es recursivamente enumerable. □

3 Propiedades de los lengs. recursivos y rec. enumerables

Este resultado es muy importante por los siguientes motivos:

- El lenguaje $L = \{w_i \mid w_i \text{ es aceptado por } M_i\}$ no puede ser recursivo. Si lo fuera, su complementario también lo sería. Por tanto, tenemos un lenguaje que es recursivamente enumerable, pero que no es recursivo, lo que implica que la inclusión de los lenguajes recursivos dentro de los recursivamente enumerables es una inclusión propia.
- El lenguaje $\Sigma^* - L$ no es recursivamente enumerable. Por tanto, fuera de los lenguajes recursivamente enumerables también hay cosas. ¿Qué cosas? Las cosas que una MT no puede aceptar, las cuales se corresponden, tal y como veremos más adelante, con los problemas irresolubles.



3 Propiedades de los lengs. recursivos y rec. enumerables

Como ya hemos comentado, la MT se puede considerar como un modelo abstracto de computación. Una MT que realiza una tarea (aceptar un lenguaje o ejecutar un determinado comportamiento patrón sobre una o varias cintas) modela un proceso. Los procesos que siempre terminan se denominan **algoritmos**, y por tanto una MT que pare sobre cualquier cadena de entrada es un modelo de algoritmo.

3 Propiedades de los lengs. recursivos y rec. enumerables

Como ya hemos comentado, la MT se puede considerar como un modelo abstracto de computación. Una MT que realiza una tarea (aceptar un lenguaje o ejecutar un determinado comportamiento patrón sobre una o varias cintas) modela un proceso. Los procesos que siempre terminan se denominan **algoritmos**, y por tanto una MT que pare sobre cualquier cadena de entrada es un modelo de algoritmo.

Esto significa que, para los lenguajes recursivos, existe un algoritmo para ver si una cadena w está en un lenguaje recursivo L. Este algoritmo viene dado por la MT que para sobre cualquier cadena y acepta L.

3 Propiedades de los lengs. recursivos y rec. enumerables

Como ya hemos comentado, la MT se puede considerar como un modelo abstracto de computación. Una MT que realiza una tarea (aceptar un lenguaje o ejecutar un determinado comportamiento patrón sobre una o varias cintas) modela un proceso. Los procesos que siempre terminan se denominan **algoritmos**, y por tanto una MT que pare sobre cualquier cadena de entrada es un modelo de algoritmo.

Esto significa que, para los lenguajes recursivos, existe un algoritmo para ver si una cadena w está en un lenguaje recursivo L. Este algoritmo viene dado por la MT que para sobre cualquier cadena y acepta L.

Sin embargo, si L es recursivamente enumerable pero no recursivo, no existe ninguna MT que pare sobre cualquier cadena y acepte L. De esto se deduce que no hay ningún modelo de algoritmo que determine si $w \in L$, para una cadena arbitraria w.

3 Propiedades de los lengs. recursivos y rec. enumerables

Como ya hemos comentado, la MT se puede considerar como un modelo abstracto de computación. Una MT que realiza una tarea (aceptar un lenguaje o ejecutar un determinado comportamiento patrón sobre una o varias cintas) modela un proceso. Los procesos que siempre terminan se denominan **algoritmos**, y por tanto una MT que pare sobre cualquier cadena de entrada es un modelo de algoritmo.

Esto significa que, para los lenguajes recursivos, existe un algoritmo para ver si una cadena w está en un lenguaje recursivo L. Este algoritmo viene dado por la MT que para sobre cualquier cadena y acepta L.

Sin embargo, si L es recursivamente enumerable pero no recursivo, no existe ninguna MT que pare sobre cualquier cadena y acepte L. De esto se deduce que no hay ningún modelo de algoritmo que determine si $w \in L$, para una cadena arbitraria w.

Es decir, el problema de ver si $w \in L$ para los lenguajes recursivamente enumerables que no son recursivos es un **problema irresoluble**. Y de hecho, éste es realmente el primer ejemplo de problema irresoluble que aparece, aunque volveremos sobre esta cuestión más adelante y veremos más ejemplos de este tipo de problemas.

3 Propiedades de los lengs. recursivos y rec. enumerables

Para finalizar la sección, presentamos otros dos resultados también muy importantes.

Teorema

Si L es un lenguaje recursivamente enumerable para el cual $\Sigma^* - L$ también es recursivamente enumerable, entonces L es recursivo.

3 Propiedades de los lengs. recursivos y rec. enumerables

Para finalizar la sección, presentamos otros dos resultados también muy importantes.

Teorema

Si L es un lenguaje recursivamente enumerable para el cual $\Sigma^* - L$ también es recursivamente enumerable, entonces L es recursivo.

Demostración:

Dadas M_1 y M_2, dos MT,s tales que $L(M_1) = L$ y $L(M_2) = \Sigma^* - L$, construimos una nueva MT M de dos cintas, que simule simultáneamente a M_1 sobre la primera y a M_2 sobre la segunda, y copiamos la cadena de entrada w en ambas cintas.

En este caso, cualquier cadena de entrada w estará en L o bien en $\Sigma^* - L$.
Por tanto, alguna de las dos MT,s M_1 o M_2 parará con w como entrada.
Si M_1 para, M para y acepta, y si M_2 para, M para y rechaza.
Así pues, $L(M) = L$ y M para siempre, con lo cual L es recursivo. □

3 Propiedades de los lengs. recursivos y rec. enumerables

Para finalizar la sección, presentamos otros dos resultados también muy importantes.

Teorema

Si L es un lenguaje recursivamente enumerable para el cual $\Sigma^* - L$ también es recursivamente enumerable, entonces L es recursivo.

Demostración:

Dadas M_1 y M_2, dos MT,s tales que $L(M_1) = L$ y $L(M_2) = \Sigma^* - L$, construimos una nueva MT M de dos cintas, que simule simultáneamente a M_1 sobre la primera y a M_2 sobre la segunda, y copiamos la cadena de entrada w en ambas cintas.

En este caso, cualquier cadena de entrada w estará en L o bien en $\Sigma^* - L$.
Por tanto, alguna de las dos MT,s M_1 o M_2 parará con w como entrada.
Si M_1 para, M para y acepta, y si M_2 para, M para y rechaza.
Así pues, $L(M) = L$ y M para siempre, con lo cual L es recursivo. □

Teorema

Un lenguaje L es recursivamente enumerable si y sólo si L es enumerado por alguna máquina de Turing. □

Contenidos

4 Gramáticas no restringidas y lenguajes rec. enumerables

Hemos visto que se pueden especificar lenguajes mediante la descripción de procesos para:

- Reconocer sus cadenas (para lo cual se han estudiado varios tipos de autómatas).
- O bien generar sus cadenas (labor realizada por las gramáticas).

En el caso particular de las gramáticas regulares ($A \to \alpha$, donde $A \in N$ y $\alpha \in \Sigma^*(N \cup \epsilon)$) y de las gramáticas independientes del contexto ($A \to \alpha$, donde $A \in N$ y $\alpha \in (N \cup \Sigma)^*$), habíamos restringido bastante la forma de las producciones.

4 Gramáticas no restringidas y lenguajes rec. enumerables

Hemos visto que se pueden especificar lenguajes mediante la descripción de procesos para:

- Reconocer sus cadenas (para lo cual se han estudiado varios tipos de autómatas).
- O bien generar sus cadenas (labor realizada por las gramáticas).

En el caso particular de las gramáticas regulares ($A \to \alpha$, donde $A \in N$ y $\alpha \in \Sigma^*(N \cup \epsilon)$) y de las gramáticas independientes del contexto ($A \to \alpha$, donde $A \in N$ y $\alpha \in (N \cup \Sigma)^*$), habíamos restringido bastante la forma de las producciones.

Veremos ahora hasta qué punto se pueden relajar estas restricciones, y cómo se relaciona esto con los lenguajes aceptados por las MT,s.

Definición

Una **gramática no restringida** o **estructurada por frases** es aquélla con reglas de producción de la forma $\alpha \to \beta$, donde $\alpha \in (N \cup \Sigma)^+$ y $\beta \in (N \cup \Sigma)^*$.

Es decir, la parte derecha de las producciones es lo más general posible, al igual que ocurre en las gramáticas independientes del contexto.

Y, en este caso, hacemos lo mismo con la parte izquierda, siempre y cuando contenga al menos un símbolo no terminal.

4 Gramáticas no restringidas y lenguajes rec. enumerables

Hemos visto que se pueden especificar lenguajes mediante la descripción de procesos para:

- Reconocer sus cadenas (para lo cual se han estudiado varios tipos de autómatas).
- O bien generar sus cadenas (labor realizada por las gramáticas).

En el caso particular de las gramáticas regulares ($A \to \alpha$, donde $A \in N$ y $\alpha \in \Sigma^*(N \cup \epsilon)$) y de las gramáticas independientes del contexto ($A \to \alpha$, donde $A \in N$ y $\alpha \in (N \cup \Sigma)^*$), habíamos restringido bastante la forma de las producciones.

Veremos ahora hasta qué punto se pueden relajar estas restricciones, y cómo se relaciona esto con los lenguajes aceptados por las MT,s.

Definición

Una **gramática no restringida** o **estructurada por frases** es aquélla con reglas de producción de la forma $\alpha \to \beta$, donde $\alpha \in (N \cup \Sigma)^+$ y $\beta \in (N \cup \Sigma)^*$.

Es decir, la parte derecha de las producciones es lo más general posible, al igual que ocurre en las gramáticas independientes del contexto.

Y, en este caso, hacemos lo mismo con la parte izquierda, siempre y cuando contenga al menos un símbolo no terminal.

Obsérvese que cualquier gramática regular o independiente del contexto es también una gramática no restringida. Además, debido a que este tipo de gramáticas es menos restrictivo, tendremos una potencia generativa mayor que con cualquier gramática de los tipos anteriores.

4 Gramáticas no restringidas y lenguajes rec. enumerables

Por ejemplo, el lenguaje $\{a^n b^n c^n \mid n \geq 1\}$ puede generarse con la gramática no restringida:

$$
\begin{array}{ll}
S \to aSBC \mid aBC & bB \to bb \\
CB \to BC & bC \to bc \\
aB \to ab & cC \to cc
\end{array}
$$

4 Gramáticas no restringidas y lenguajes rec. enumerables

Por ejemplo, el lenguaje $\{a^n b^n c^n \mid n \geq 1\}$ puede generarse con la gramática no restringida:

$$\begin{array}{ll} S \to aSBC \mid aBC & bB \to bb \\ CB \to BC & bC \to bc \\ aB \to ab & cC \to cc \end{array}$$

Al reescribir S, siempre habrá el mismo número de $a$es, Bes y Ces, y además de forma que las $a$es preceden a las Bes, y éstas a las Ces:

$$\ldots\ a\ a\ a\ a\ a\ B\ C\ B\ C\ B\ C\ B\ C\ B\ C\ \ldots$$

La producción $CB \to BC$ permite ir reuniendo las Bes y las Ces para obtener:

$$a^n B^n C^n$$

Finalmente, $aB \to ab$ y $bB \to bb$ transforman toda B en b, y $bC \to bc$ y $cC \to cc$ transforman toda C en c, para obtener:

$$a^n b^n c^n$$

4 Gramáticas no restringidas y lenguajes rec. enumerables

Por ejemplo, el lenguaje $\{a^n b^n c^n \mid n \geq 1\}$ puede generarse con la gramática no restringida:

$$\begin{array}{ll} S \to aSBC \mid aBC & bB \to bb \\ CB \to BC & bC \to bc \\ aB \to ab & cC \to cc \end{array}$$

Al reescribir S, siempre habrá el mismo número de $a$es, Bes y Ces, y además de forma que las $a$es preceden a las Bes, y éstas a las Ces:

$$\ldots\ a\ a\ a\ a\ a\ B\ C\ B\ C\ B\ C\ B\ C\ B\ C\ \ldots$$

La producción $CB \to BC$ permite ir reuniendo las Bes y las Ces para obtener:

$$a^n B^n C^n$$

Finalmente, $aB \to ab$ y $bB \to bb$ transforman toda B en b, y $bC \to bc$ y $cC \to cc$ transforman toda C en c, para obtener:

$$a^n b^n c^n$$

En resumen, queda demostrado que esta gramática puede efectivamente derivar un lenguaje no independiente del contexto.

4 Gramáticas no restringidas y lenguajes rec. enumerables

Otro ejemplo es el del lenguaje $\{a^{2^n} \mid n > 0\}$, el cual puede generarse con:

1) $S \to ACaB$
2) $CB \to DB$
3) $CB \to E$
4) $AD \to AC$
5) $AE \to \epsilon$
6) $Ca \to aaC$
7) $aD \to Da$
8) $aE \to Ea$

Otro ejemplo es el del lenguaje $\{a^{2^n} \mid n > 0\}$, el cual puede generarse con:

1) $S \to ACaB$	3) $CB \to E$	5) $AE \to \epsilon$	7) $aD \to Da$
2) $CB \to DB$	4) $AD \to AC$	6) $Ca \to aaC$	8) $aE \to Ea$

Como ejemplo de derivación, consideremos $a^{2^2} = a^4 = aaaa$.

$$\underline{S} \overset{1}{\Longrightarrow} A\underline{Ca}B \overset{6}{\Longrightarrow} Aaa\underline{CB} \overset{2}{\Longrightarrow} Aaa\underline{DB} \overset{7}{\Longrightarrow} Aa\underline{Da}B \overset{7}{\Longrightarrow} \underline{AD}aaB$$

$$\overset{4}{\Longrightarrow} A\underline{Ca}aB \overset{6}{\Longrightarrow} Aaa\underline{Ca}B \overset{6}{\Longrightarrow} Aaaaa\underline{CB} \overset{3}{\Longrightarrow} Aaaa\underline{aE} \overset{8}{\Longrightarrow} Aaa\underline{aE}a$$

$$\overset{8}{\Longrightarrow} Aa\underline{aE}aa \overset{8}{\Longrightarrow} A\underline{aE}aaa \overset{8}{\Longrightarrow} \underline{AE}aaaa \overset{5}{\Longrightarrow} aaaa$$

4 Gramáticas no restringidas y lenguajes rec. enumerables

Otro ejemplo es el del lenguaje $\{a^{2^n} \mid n > 0\}$, el cual puede generarse con:

1) $S \to ACaB$	3) $CB \to E$	5) $AE \to \epsilon$	7) $aD \to Da$
2) $CB \to DB$	4) $AD \to AC$	6) $Ca \to aaC$	8) $aE \to Ea$

Como ejemplo de derivación, consideremos $a^{2^2} = a^4 = aaaa$.

$$\underline{S} \overset{1}{\Longrightarrow} A\underline{Ca}B \overset{6}{\Longrightarrow} Aaa\underline{CB} \overset{2}{\Longrightarrow} Aaa\underline{DB} \overset{7}{\Longrightarrow} Aa\underline{Da}B \overset{7}{\Longrightarrow} \underline{AD}aaB$$

$$\overset{4}{\Longrightarrow} A\underline{Ca}aB \overset{6}{\Longrightarrow} Aaa\underline{Ca}B \overset{6}{\Longrightarrow} Aaaaa\underline{CB} \overset{3}{\Longrightarrow} Aaaa\underline{aE} \overset{8}{\Longrightarrow} Aaa\underline{aE}a$$

$$\overset{8}{\Longrightarrow} Aa\underline{aE}aa \overset{8}{\Longrightarrow} A\underline{aE}aaa \overset{8}{\Longrightarrow} \underline{AE}aaaa \overset{5}{\Longrightarrow} aaaa$$

A y B actúan como marcadores de los extremos de la cadena de $a$es que está siendo generada. C se desplaza hacia la derecha dupicando el número de $a$es hasta llegar junto a B y entonces se transforma en una D. D se desplaza hacia la izquierda hasta encontrar a A y se convierte en C. Pero CB podría ser reemplazada también por E para terminar la generación de $a$es. Entonces E se desplaza hacia la izquierda hasta encontrar a A y en ese momento se elimina AE.

Una vez más, tenemos una gramática que deriva un lenguaje no independiente del contexto.

4 Gramáticas no restringidas y lenguajes rec. enumerables

Para cualquier gramática G no restringida, se podrían obtener todas las cadenas w tales que $S \Rightarrow w$ en un solo paso. El número de cadenas de este tipo será finito, ya que el número de reglas de la gramática es finito. Podemos entonces listar las cadenas que se derivan en 2, 3, 4 o más pasos, y cada una de estas colecciones será también finita. Por tanto, todo el lenguaje $L(G)$ podría ser listado así, y por tanto podría haber una MT que enumerase $L(G)$.

Este razonamiento constituye de hecho la demostración intuitiva del siguiente resultado.

Teorema

Si G es una gramática no restringida, entonces $L(G)$ es recursivamente enumerable. □

4 Gramáticas no restringidas y lenguajes rec. enumerables

Para cualquier gramática G no restringida, se podrían obtener todas las cadenas w tales que $S \Rightarrow w$ en un solo paso. El número de cadenas de este tipo será finito, ya que el número de reglas de la gramática es finito. Podemos entonces listar las cadenas que se derivan en 2, 3, 4 o más pasos, y cada una de estas colecciones será también finita. Por tanto, todo el lenguaje $L(G)$ podría ser listado así, y por tanto podría haber una MT que enumerase $L(G)$.

Este razonamiento constituye de hecho la demostración intuitiva del siguiente resultado.

Teorema

Si G es una gramática no restringida, entonces $L(G)$ es recursivamente enumerable. □

Y también se cumple que todo lenguaje recursivamente enumerable es generado por una gramática no restringida. Por lo tanto, también se verifica este otro resultado.

Teorema

Un lenguaje L es recursivamente enumerable si y sólo si $L = L(G)$ para alguna gramática G no restringida. □

4 Gramáticas no restringidas y lenguajes rec. enumerables

Para cualquier gramática G no restringida, se podrían obtener todas las cadenas w tales que $S \Rightarrow w$ en un solo paso. El número de cadenas de este tipo será finito, ya que el número de reglas de la gramática es finito. Podemos entonces listar las cadenas que se derivan en 2, 3, 4 o más pasos, y cada una de estas colecciones será también finita. Por tanto, todo el lenguaje $L(G)$ podría ser listado así, y por tanto podría haber una MT que enumerase $L(G)$.

Este razonamiento constituye de hecho la demostración intuitiva del siguiente resultado.

Teorema

Si G es una gramática no restringida, entonces $L(G)$ es recursivamente enumerable. □

Y también se cumple que todo lenguaje recursivamente enumerable es generado por una gramática no restringida. Por lo tanto, también se verifica este otro resultado.

Teorema

Un lenguaje L es recursivamente enumerable si y sólo si $L = L(G)$ para alguna gramática G no restringida. □

En resumen, los lenguajes que se obtienen a partir de las gramáticas no restringidas son exactamente los mismos que son aceptados por las MT,s.

Contenidos

5 Lenguajes sensibles al contexto y la jerarquía de Chomsky

Entre las gramáticas no restringidas y las independientes del contexto se pueden definir varios tipos de gramáticas con diferentes niveles de restricción. No todas producen lenguajes interesantes, pero unas que sí lo hacen son las **gramáticas sensibles al contexto**. Estas gramáticas producen una clase de lenguajes que está situada estrictamente entre los lenguajes independientes del contexto y los lenguajes recursivos, y cuyo formalismo reconocedor viene dado por los autómatas lineales acotados o ALA,s.

Definición

Una **gramática sensible al contexto** es aquélla en la que todas las reglas son de la forma $\alpha \to \beta$, donde $\alpha, \beta \in (N \cup \Sigma)^+$ y $|\alpha| \leq |\beta|$.

5 Lenguajes sensibles al contexto y la jerarquía de Chomsky

Entre las gramáticas no restringidas y las independientes del contexto se pueden definir varios tipos de gramáticas con diferentes niveles de restricción. No todas producen lenguajes interesantes, pero unas que sí lo hacen son las **gramáticas sensibles al contexto**. Estas gramáticas producen una clase de lenguajes que está situada estrictamente entre los lenguajes independientes del contexto y los lenguajes recursivos, y cuyo formalismo reconocedor viene dado por los autómatas lineales acotados o ALA,s.

Definición

Una **gramática sensible al contexto** es aquélla en la que todas las reglas son de la forma $\alpha \to \beta$, donde $\alpha, \beta \in (N \cup \Sigma)^+$ y $|\alpha| \leq |\beta|$.

En 1959, Noam Chomsky clasificó las gramáticas (y, por tanto, también los lenguajes a los que dan lugar) en cuatro familias:

- Tipo 0: No restringidas (que dan lugar a los lenguajes recursivamente enumerables o *r.e.*).
- Tipo 1: Sensibles al contexto (que dan lugar a los lenguajes sensibles al contexto o *s.c.*).
- Tipo 2: Independientes del contexto (que dan lugar a los lenguajes independientes del contexto o *i.c.*).
- Tipo 3: Regulares (que dan lugar a los lenguajes regulares o *reg.*).

Todo lenguaje de tipo i es también de tipo $i-1$, siendo cada una de estas inclusiones una inclusión propia. Esto es lo que se conoce como la **jerarquía de Chomsky**.

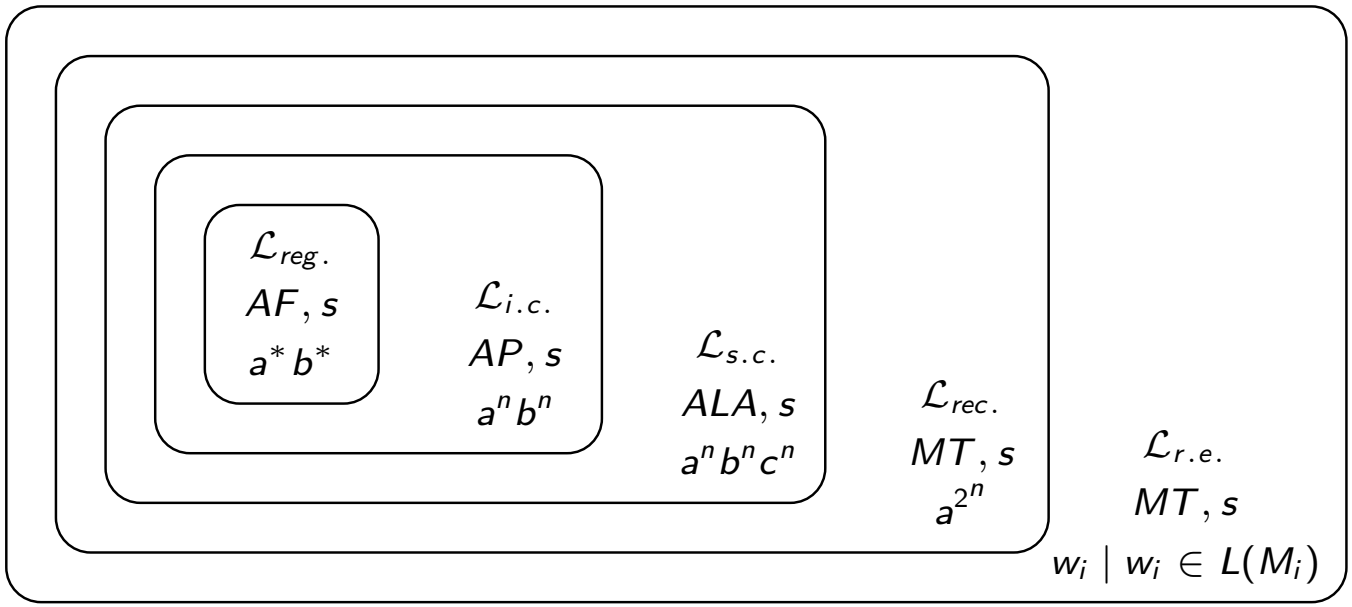
5 Lenguajes sensibles al contexto y la jerarquía de Chomsky

Además, nosotros hemos estudiado los lenguajes recursivos (o *rec.*), que están entre los sensibles al contexto y los recursivamente enumerables, y todavía se podrían definir más dentro de esta misma jerarquía. Pero todas las relaciones que hemos visto hasta aquí se pueden resumir formalmente en el siguiente resultado y gráficamente en la siguiente figura.

Teorema de la Jerarquía

Siendo $\mathcal{L}_x$ la representación del conjunto de lenguajes de tipo x, se verifica que:

$$\mathcal{L}_{reg.} \subset \mathcal{L}_{i.c.} \subset \mathcal{L}_{s.c.} \subset \mathcal{L}_{rec.} \subset \mathcal{L}_{r.e.}$$



Fin del capítulo

Fin del capítulo

"Lenguajes Recursivamente Enumerables"