Lema del bombeo para lenguajes independientes de contexto y autómatas de pila

Autómatas y lenguajes formales

16 de abril de 2023

Lema del bombeo para lenguajes independientes de contexto

Como vimos en nuestro capítulo anterior existe un lema del bombeo para lenguajes regulares, que nos ayuda a demostrar que un lenguaje no es regular. Para lenguajes independientes de contexto tenemos su versión que de manera similar sirve para demostrar que un lenguaje no es independiente de contexto.

Si recuerdan de su tarea todo lenguaje regular puede ser expresado como una gramática independiente de contexto (no necesariamente al revés), entonces el lema es una ampliación del anterior.

Lema 1 *(**Lema del bombeo para CFL**) Para todo lenguaje independiente de contexto* $\mathbf{A}$ *existe una* $k \geq 0$ *tal que para toda* $z \in \mathbf{A}$ *de longitud al menos* k *puede ser dividida en cinco subcadenas* $z = uvwxy$ *tales que* $vx \notin \epsilon$, $|vwx| \leq k$, *y para toda* $i \geq 0$, $uv^iwx^iy \in \mathbf{A}$.

Son claras las diferencias con el lema del bombeo para lenguajes regulares, pero además, para que vean de qué se trata el lema es útil verlo en la forma normal de Chomsky (de quien espero ya hayan escuchado su *split* con Bad Religion). Podemos ver el árbol de generación de un lenguaje independiente de contexto.

Tenemos el lenguaje independiente de contexto, en forma normal de Chomsky, $\mathbf{A}$, siguiente:

$$\begin{aligned} S &\to AC|AB \\ A &\to a \\ B &\to b \\ C &\to SB, \end{aligned} \tag{1}$$

que puede generar la cadena *aaaabbbb* a partir del árbol de generación mostrado en la figura 1.

Para mostrar como es que el lema del bombeo para CFL es cierto tomamos el paseo más largo en el árbol, que corresponde al de la primera b, leyendo la palabra de izquierda a derecha. La longitud de este paseo es de ocho aristas, y sólo hasta la última hoja se obtiene un símbolo terminal (en el camino intermedio hay puras variables).

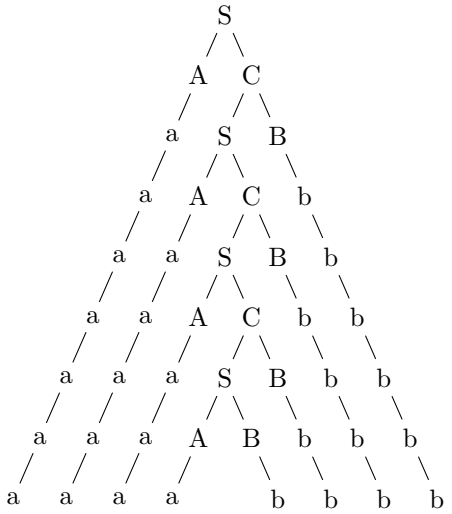


Figura 1: Árbol de generación para la cadena *aaaabbbb* con la gramática dada en la ecuación 1.

Aquí me podrían decir que todos los paseos que llegan hasta las hojas miden ocho aristas, pero esto es un poco tramposo por como se dibujó el árbol, si revisan bien la primera a, de izquierda a derecha, se obtuvo tras sólo dos arista, así como la última b (la que está más a la derecha, porque extrema derecha suena muy feo) se obtuvo a las tres aristas. Las demás aristas son sólo para tener a la misma altura la palabra ya final.

Pero entonces nos paramos en la primera b de izquierda a derecha, si recorremos el árbol en dirección inversa nos detenemos en alguna variable que aparezca más de una vez, viendo de abajo hacia arriba la primera que salta a la vista es la S.

Además dividimos la cadena producida en subcadenas, de la forma que indica el lema, como se puede ver en la figura 2a. Tenemos $u = aa$, $v = a$, $w = ab$, $x = b$ y $y = bb$, que cumplen las condiciones del lema, ahora vamos a ver como a partir de este árbol de producción podemos generar una palabra uv^iwx^iy y sigue siendo una cadena en el lenguaje independiente del contexto.

Para repetir las cadenas v y x las veces que sea necesario tomamos el sub árbol derivado de la aparición superior de S (la segunda marcada de abajo hacia arriba), lo copiamos y lo pegamos en la primera aparición de esa misma variable (leída de abajo hacia arriba, la otra marcada), quedando algo como se muestra en la figura 2b.

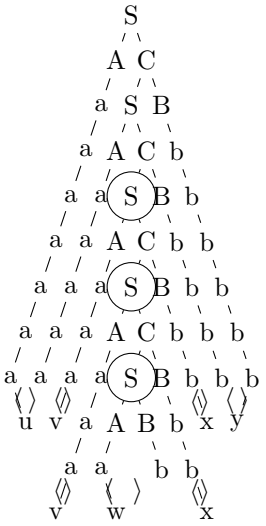
Si se repite este paso i veces tendremos que v y x se repiten i veces, y como está en el mismo árbol de generación, sin romper ninguna regla, sigue siendo parte del lenguaje independiente de contexto.

La prueba del lema va más o menos así, es la explicación a groso modo, ahora veamos una aplicación:

Demuestra que el lenguaje $\{0^n\#0^{2n}\#0^{3n} | n \geq 0\}$ no es independiente de contexto.

Para hacerlo seguimos los pasos del juego como en el lema de bombeo para

(a) Árbol de generación para la gramática dada en la ecuación 1, marcando las apariciones de la regla S y la división de la cadena en subcadenas.



(b) Árbol donde se aplica el lema del bombeo, repitiendo las cadenas v y x.

Figura 2: Figuras de los pasos para el lema del bombeo en gramáticas libres de contexto.

lenguajes regulares:

1. Te dan una $k \geq 0$.
2. Tú eliges una $z \in \mathbf{A}$ tal que $|z| \geq k$, en este caso puedes elegir $z = 0^k \# 0^{2k} \# 0^{3k}$, ya que $z \in \mathbf{A}$ y $|z| = k + 2k + 3k + 2$, se suma un dos por los símbolos # en la palabra.
3. Ahora el contrincante (el demonio, o quien quiera que sea) divide la cadena que diste en subcadenas de forma que $z = uvwxy$ con $vx \neq \epsilon$ y $|vwx| \leq k$. No tiene que decírtelas, pero aún sin decirlas puedes ver que le vas a ganar en el siguiente paso.
4. Eliges $i \geq 0$ para ver si con las cadenas dadas en el paso anterior puedes hacer la palabra $uv^i wx^i y$ y que siga siendo parte del lenguaje libre de contexto.

Y ahí es donde vale, pues si el demonio elige una v o x que contengan el símbolo (al menos uno) # ya le ganaste, pues al repetir esa cadena tendrás más de dos # en la cadena final, y ya no es parte del lenguaje.

Ahora imagina que el contrincante toma v y x que no contengan esos símbolos, entonces serían fracciones de las potencias, y al repetirlas ya no quedaría la regla de potencia. Por ejemplo, toma $u = 0^{k-m}$, $v = 0^m$, $w = \#$, $x = 0^s$ y $y = 0^{2k-s} \# 0^{3k}$, que cumple lo que se pide: $vx = 0^m 0^s \neq \epsilon$ y $|vwx| = 0^m \# 0^s \leq k$ si $s + m + 1 \leq k$, que en parte es condición para que tenga sentido su elección de cadenas.

En este caso si construimos la nueva cadena $uv^iwx^iy = 0^{k-m}0^{im}\#0^{is}0^{2k-s}\#0^{3k}$, para cualquier $i \geq 2$ esta cadena ya no es parte del lenguaje, no es un lenguaje independiente de contexto.

Quizá la explicación fue muy escueta, espero sea clara la idea, de cualquier forma no dejen de consultarlo en su libro de preferencia.

Autómatas de pila

Un autómata de pila no determinista es una ampliación al autómata finito determinista, se le agrega una cinta donde puede guardar, una memoria de pila del tipo *Last In First Out* (LIFO, el último entra, el primero sale). Su definición formal:

Definición 1 ***Autómata de pila no determinista****. Es una septeta*

$$M = (Q, \Sigma, \Gamma, \delta, s, \perp, F),$$

donde

- Q *es le conjunto de estados*
- Σ *es el alfabeto de salida*
- Γ *es el alfabeto e la memoria*
- $\delta \subseteq (Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma) \times (Q \times \Gamma^*)$*, la relación de transición.*

Las funciones de transición tendrán como entrada (p, a, A), donde p es un estado, a un símbolo del alfabeto de entrada y A un símbolo del alfabeto de la cinta de memoria, dando como resultado la transición a un estado q y escribiendo en la memoria una cadena $B_1B_2...B_k$ (B_k al inicio de la memoria y B_1 al final), un esquema de este autómata se puede ver en la figura 3, no necesariamente es un diagrama útil para trabajar en ellos.

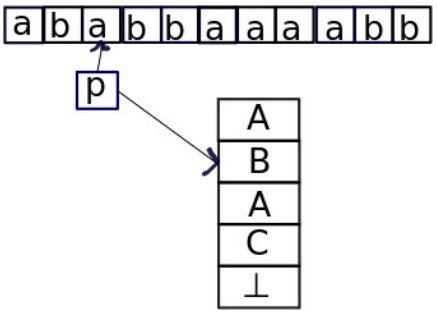


Figura 3: Diagrama de un autómata de pila, p es el estado en que se encuentra, la cinta superior guarda la cadena de entrada y la cinta de la derecha funciona como memoria.

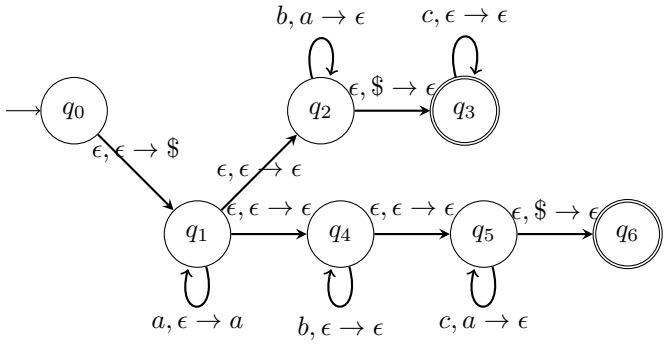
¿Cómo trabajar con los autómatas de pila no deterministas? Se pueden describir a partir de todas sus funciones de transición, pero como ya nos acostumbramos, queremos hacer un dibujito con bolitas y flechitas[1].

Veamos un ejemplo:

El autómata de pila que acepta el lenguaje

$$\{a^i b^j c^k | i, j, k \geq 0, i = j \vee i = k\}$$

Lo que hará será guardar las a's en la memoria y luego checar si las b's o c's están en la cadena en el mismo número ¿cómo sabe el autómata si checa las b's o c's? Ahí entra el no determinismo, supondremos que de avanzada el autómata puede prever.



Empezamos en el estado q_0, aún no recorre la cinta de entrada y la de memoria está en blanco. Como el autómata es no determinista puede irse por dos caminos, uno en el que checa las b's y otro donde checa las c's (en esta construcción no puede checar los dos a la vez). Entonces sin tener nada en la entrada, en la cinta, se pasa al estado q_1 y marca la cinta de salida con un \$ como para decir "empezamos a contar".

Ya en el estado q_1 empezamos a recibir las a's, todas las que lleguen son marcadas en la memoria y se sigue en el mismo estado, de ahí no nos movemos hasta que llegue una b o una c, paso que hace de forma no determinista (sin que lea nada en la cadena de entrada se pasa a los estados correspondientes, q_2 hace el conteo de b's, y q_5 el de c's).

Vamos a la rama para contar b's, sin que se lea nada en la cadena/cinta de entrada, sin leer nada en la memoria y sin escribir nada en ella el autómata "adivina" que debe contar b's y se pasa al estado q_2. Ya ahí, por cada b que se lea en la cinta de entrada se lee una a, se mueve en ambas cintas a leer el siguiente caracter y no escribe nada. Si no lee nada en la cinta de entrada y lee un símbolo \$ en la cinta de memoria se mueve al estado q_3 que ya es de aceptación. Si llegan cualquier número de c's, ya no importa, ya se contaron las b's, se aceptan y se queda como aceptado.

Si lo que va a contar son c's el autómata, sin leer nada de entrada, leer nada en memoria y sin escribir nada pasa al estado q_4. En ese estado lo que sigue, si

[1] ¿Eso ayuda a nuestra comprensión de la idea? Esa es una buena pregunta que varios especialistas en enseñanza de las matemáticas se hacen, si las gráficas o diagramas son realmente esenciales para la comprensión de las ideas, o sólo es una construcción para los que ya están acostumbrados a los métodos escolarizados. Hay personas con un conocimiento matemático funcional, pero no formal,que pueden entender conceptos elaborados, pero no de la misma forma

la cadena está en el lenguaje, es leer las b's, pero como estas no se van a contar, se leen sin leer nada en memoria y sin escribir nada en memoria, hasta que llega una c.

Antes de leer la c el autómata "adivina" que ya vienen, y sin leer ni escribir nada se pasa al estado q_5 y aquí cuenta las c's, por cada c de entrada lee una a en memoria y no escribe nada, las compara.

Si ya no lee nada de entrada y en la memoria hay un símbolo \$ se pasa al estado de aceptación q_6 y se queda ahí (ya no hay nada por leer, ya se leyeron las b's).

No dejen de revisar más ejemplos, por el momento ahí lo dejo, y paso a la equivalencia entre autómatas de pila y gramáticas en forma normal de Greibach.

De forma normal de Greibach a autómatas de pila

Antes de leer esta sección recuerden ver las notas referentes a las formas normales de Chomsky y Greibach, partiré de que saben como llegar a esas formas, sobre todo la de Greibach que es la que necesitamos aquí.

Como segura ya vieron en notas o clases pasadas ya saben que hay limitaciones para los lenguajes regulares. Hay lenguajes que no pueden ser aceptados por un autómata finito determinista o no. En la jerarquía de Chomsky el siguiente nivel de complejidad son las gramáticas independientes de contexto. Algunas de ellas pueden traducirse a un automata finito pero muchas otras van más allá de ese mecanismo. El siguiente paso en la representación gráfica de estos lenguajes es el autómata de pila.

Como ya vieron párrafos arriba el autómata de pila incluye una pila que funciona como una memoria de acceso bastante limitado. Las gramáticas independientes de contexto que pueden convertirse a forma normal de Greibach pasan de manera directa a un autómata de pila (he ahí una de las funciones de tener una gramática independiente de contexto en tal forma normal).

Para armar el autómata de pila se siguen las instrucciones a continuación:

1. Poner la gramática en forma normal de Greibach (como ya saben, para llegar a ella lo más fácil es poner la gramática en forma normal de Chomsky antes y partir de ahí).

2. Mandamos el símbolo inicial a la pila por la función de transición

$$\delta(q_0, \epsilon, \perp) \to (q_1, S\ \perp)$$

   donde q_0 es el estado inicial.

3. Para una producción de la forma $V_1 \to aV_i...V_n$, la función de transición será

$$\delta(q_1, a, V_1) \to (q_1, V_i...V_n)$$

4. Para una producción de la forma $V_1 \to b$ la función de transición será

$$\delta(q_1, b, V_1) \to (q_1, \epsilon)$$

5. Para aceptar la cadena dos funciones de transición se agregan, una para aceptar con pila vacía y otra al aceptar al llegar al estado de aceptación.

Noten que solo hay tres estados, el inicial q_0, un q_1 al que van todas las producciones antes de aceptar y un estado final de aceptación q_f.

Para aplicar lo mencionado veamos un ejemplo.

Construye el autómata de pila que acepta el lenguaje generado por la gramática

$$S \to aB$$
$$B \to bA \mid b$$
$$A \to aB$$

Paso 1: Damos la transición inicial para ir tomando en cuenta las producciones:

$$\delta(q_0, \epsilon, \perp) \to (q_1, S \perp)$$

Paso 2: Convertimos las producciones del tipo $V_a \to aV_i...V_n$ y del tipo $V_a \to a$ de acuerdo a las instrucciones:

$$\delta(q_1, a, S) \to (q_1, B), \text{ de } S$$
$$\delta(q_1, b, B) \to (q_1, A), \text{ de } B$$
$$\delta(q_1, b, B) \to (q_1, \epsilon), \text{ de } B$$
$$\delta(q_1, a, A) \to (q_1, B), \text{ de } A$$

Paso 3: Damos los estados finales

$$\delta(q_1, \epsilon, \perp) \to (q_f, \perp)$$
$$\delta(q_1, \epsilon, \perp) \to (q_1, \epsilon)$$

Y listo, tenemos nuestro autómata de pila, no es necesario hacer el diagrama, pero ya como se les haga más fácil. Basta con tener las funciones de transición.

Extra

Ahora no tengo discos de punk relacionados, les tengo un programa de radio sobre el antecedente latinoamericano y socialista del internet.

En Chile durante el gobierno de la Unidad Popular, encabezado por Salvador Allende, se trabajó en un plan para interconectar a las empresas estatales por medio de una red. Así se podrían atender emergencias en producción, transporte, comunicación, además de que todo sería transparente para la ciudadanía.

Si quieren saber más de esta historia, enterarse de como fue que se inició y porque no llegó a buen término no dejen de escuchar el programa dedicado, en radio ambulante: `https://radioambulante.org/audio/la-sala-que-era-un-cerebro`.

Si en esa liga no pueden descargar el podcast, bájenlo de acá: `https://play.podtrac.com/npr-510315/edge1.pod.npr.org/anon.npr-mp3/npr/ambul/2019/09/20190917_ambul_la_sala_que_era_un_cerebro.mp3`

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)

[3] Kandar, Shyamalendu “Introduction to Automata Theory, Formal Languages and Computation” Dorling Kindersley India (2013).