Universidad de Chile
Departamento de Ciencias de la Computación
CC3102 - Teoría de la Computación

Solución Control 2

Profesor: Jorge Pérez [jperez@dcc.uchile.cl]
Auxiliares: Nicolás Lehmann [nlehmann@dcc.uchile.cl]
Rodrigo Alonso [ralonso@dcc.uchile.cl]

P1. Considere el lenguaje $L = L(r)$ generado por la expresión regular $r = (100 + 010 + 001)^*$.

(a) Determine el mínimo número de estados que puede tener un AFD que acepte a L.

Solución:

Notemos que el lenguaje pedido corresponde a las palabras de largo $3n$ (con $n \in \mathbb{N}$) formadas por la concatenación de palabras de largo 3 que tienen exactamente un uno y dos ceros. Por lo tanto, mientras quede entrada por leer, basta leer 3 símbolos, verificar que sean exactamente dos ceros y un uno, y repetir.

Luego, nuestras clases de equivalencia según la relación $\mathcal{R}$ del teorema de Myhill-Nerode (es decir, $w_1 \mathcal{R} w_2 := \forall x : w_1 x \in L \Leftrightarrow w_2 x \in L$) representarán a los distintas palabras en Σ^* según los símbolos que faltan para obtener una palabra en L y una clase especial para aquellas palabras que simplemente no pueden ser un prefijo de una palabra de L.

Tendremos entonces las clases F, 0, 1, 00, 01, S.

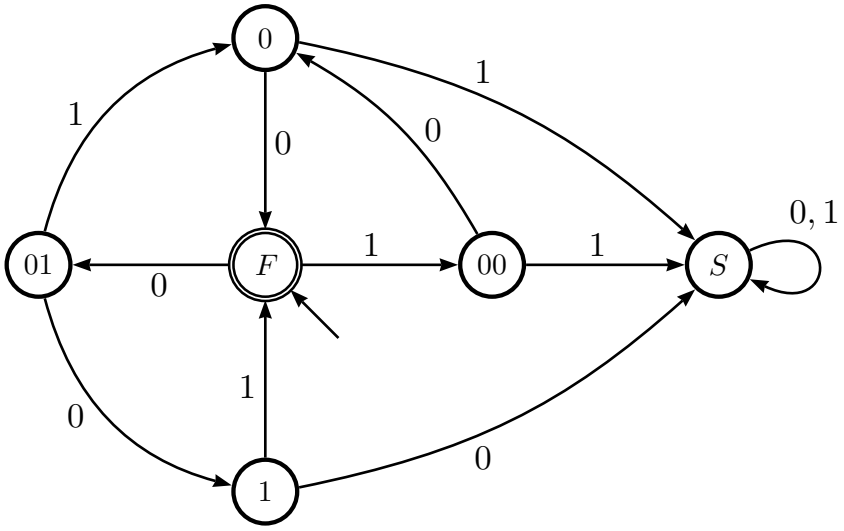
- F corresponde a las palabras en L.
- 0 corresponde a las palabras w tales que $w0 \in L$.
- 1 corresponde a las palabras w tales que $w1 \in L$.
- 00 corresponde a las palabras w tales que $w00 \in L$.
- 01 corresponde a las palabras w tales que $w01 \in L \wedge w10 \in L$.
- S corresponde a las palabras que no pueden ser prefijo de una palabra en L pues alguno de sus grupos de 3 símbolos quedará siempre con tres ceros, o más de un uno.

Es fácil ver que estas clases de equivalencia son diferentes entre sí: en el caso de S, basta tomar x tal que los strings de la clase de equivalencia con la que estamos comparando quede en L; para otros pares de clases, basta elegir un x cualquiera tal que una de las clases quede en L, este x necesariamente será un prefijo que dejará a los strings de la otra clase fuera de L.

(b) Construya el AFD con el mínimo número de estados.

Solución:

A continuación una definición gráfica del AFD. Los nombres de los estados corresponden a los nombres utilizados para las clases de equivalencia en la parte (a).



P2. Construya un autómata apilador que acepte por estado final el siguiente lenguaje.

$$L = \{a^i b^j \mid i \neq j \text{ y } 2i \neq j\}$$

Solución:

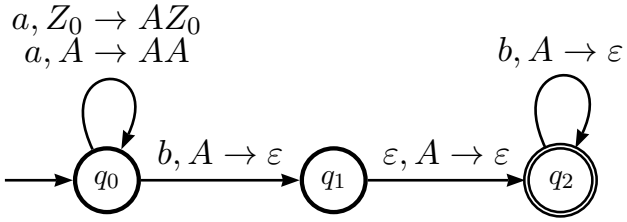
El lenguaje L está expresado como una conjunción de LLCs. De forma general la clase de los lenguajes libres de contexto no es cerrada bajo conjunción, pero en este caso particular veremos que es posible reformular el lenguaje para expresarlo como una disyunción de LLCs.

Partimos notando que la restricción $i \neq j$ y $2i \neq j$ es equivalente a $j < i$, $i < j < 2i$ o $2i < j$, y por lo tanto podemos expresar el lenguaje de la siguiente forma:

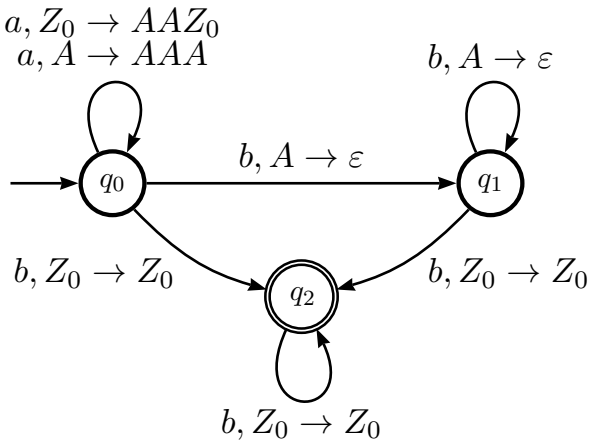
$$\text{L} = \{a^i b^j \mid j < i \vee i < j < 2i \vee 2i < j\}$$

El lenguaje expresado de esta forma corresponde a una conjunción de tres lenguajes. Bastará construir un AP para cada uno de estos con el fin de encontrar un AP para el lenguaje completo.

- $L_1 = \{a^i b^j \mid j < i\}$
  Este es un ejemplo clásico de un lenguaje libre de contexto. A continuación se detalla el AP que acepta el lenguaje por estado final. Sólo basta notar que $j < i$ es equivalente a $j + 1 \leq i$, para representar esto sacamos dos A's del stack por la primera b leída.

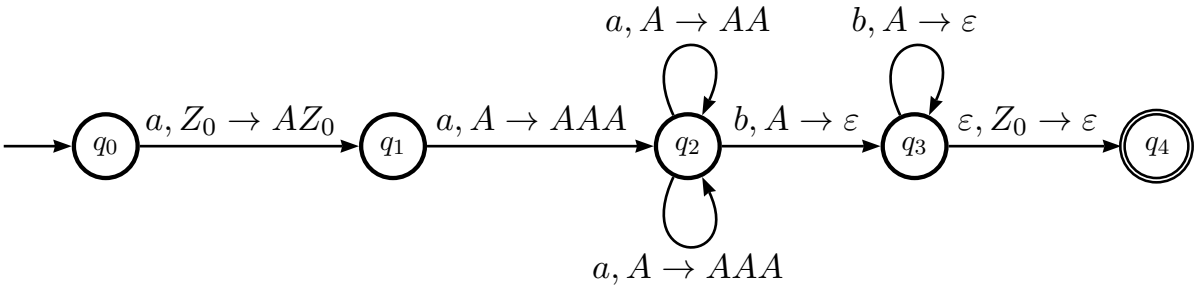


- $L_2 = \{a^i b^j \mid 2i < j\}$
  En este caso sólo basta notar que por cada a que leemos en la cadena hay que agregar dos A's a la pila. Luego debemos sacar una A del stack por cada b leída, notando que habrá al menos una b más que símbolos en el stack si nos encontramos con Z_0. A continuación se detalla el autómata.



- $L_3 = \{a^i b^j \mid i < j < 2i\}$
  Este es el caso más difícil pues debemos comprobar que j cumpla las dos cotas. Para realizar esto es necesario notar que j debe ser igual a alguno de los valores entre i y $2i$. Como no sabemos a *priori* a cuál de estos valores es igual, usaremos no determinismo para "adivinarlo". Para esto, cada vez que veamos una a en la cadena, no determinísticamente pondremos uno o dos símbolos en el stack. Posteriormente es necesario comprobar que la cantidad de b's sea exactamente igual a la cantidad de símbolos en el stack. Como cada una de las ramas de ejecución contendrá en el stack algún número de símbolos entre i y $2i$, si alguna de las ramas acepta significará que la cantidad de $b's$ es igual a un número entre i y $2i$. Finalmente sólo hay que considerar un último detalle, si siempre ponemos uno o dos símbolos en el stack entre los casos generados estaríamos considerando también

los valores i y $2i$, y por lo tanto comprobando la cota $i \leq j \leq 2i$. Para comprobar que las cotas son estrictas es necesario que el string contenga al menos 2 $a's$ y que pongamos 3 símbolos en el stack por las primeras 2 $a's$.



Finalmente el AP para $L = L_1 \cup L_2 \cup L_3$ corresponde a la unión clásica de autómatas, es decir, generar un nuevo estado inicial y crear transiciones en vacío hacia cada uno de los autómatas en la disyunción.

P3. Para cada uno de los siguientes lenguajes sobre el alfabeto $\{0, 1, \#\}$, decida si el lenguaje es o no un lenguaje libre de contexto. Si el lenguaje es libre de contexto construya una gramática libre de contexto que lo genere. Si el lenguaje no es libre de contexto, demuestre esta propiedad.

(a) $L = \{x\#y \mid x, y \in \{0,1\}^* \text{ y } x \neq y\}$.

Solución:

En este caso el lenguaje es libre de contexto y explicaremos a continuación por qué. Suponga que tiene un programa en su lenguaje favorito que comprueba si dos strings son distintos. La forma natural de realizar esto es ir comparando a la par los símbolos de cada string hasta encontrar una posición en que los símbolos difieran. En el caso de que se termine uno de los strings (el más corto es prefijo del más largo) reportar también que son distintos. Podemos ocupar esta idea para caracterizar que dos strings x y y son distintos si se cumple al menos una de las siguientes propiedades:

- Los largos de los strings son distintos, $|x| \neq |y|$.
- Existe una posición en la que los símbolos de los strings difieren.

Podemos entender el lenguaje del enunciado como una unión de los casos anteriores. El primero es sencillo y claramente define un lenguaje libre de contexto. El segundo es más complejo y una intuición para entender por qué es libre de contexto consiste en entender que podemos usar no determinismo para "adivinar" la posición en que los strings difieren.

Construiremos gramáticas para los lenguajes definidos en cada caso.

- $L_1 = \{x\#y \mid |x| \neq |y|\}$
  Este caso es sencillo. Simplemente partimos generando la misma cantidad de símbolos a ambos lados de # y en algún momento rompemos esta restricción para generar más símbolos en alguno de los lados.

  $$\begin{array}{lcl} S & \to & BSB \mid P\# \mid \#P \\ P & \to & BP \mid B \\ B & \to & 0 \mid 1 \end{array}$$

- $L_2 = \{x\#y \mid$ existe una posición en la que los símbolos de x e y difieren$\}$
  Podemos caracterizar este caso diciendo que existen particiones $x = u_x b_x w_x$ e $y = u_y b_y w_y$ con $u_x, u_y, w_x, w_y \in \Sigma^*$ y $b_x, b_y \in \Sigma$, tal que $|u_x| = |u_y|$ y $b_x \neq b_y$. Es decir las palabras en el lenguaje tendrán la siguiente forma.

  $$\underbrace{u_x b_x w_x \# u_y}_{P} \underbrace{b_y w_y}_{Q}$$

  Para cumplir la restricción $|u_x| = |u_y|$ debemos generar ambos strings de forma simultánea. Para esto ocuparemos la variable P como se muestra en la figura anterior. El resto del string será generado por una variable Q. Notar además que no es necesario imponer como restricción que $u_x \neq u_y$. Ahora bien, como b_x y b_y deben ser distintos, tendremos dos casos para P y Q de forma que el símbolo que escriban sea distinto. Juntando todo esto obtenemos la siguiente gramática.

  $$\begin{array}{lcl} S & \to & P_0Q_1 \mid P_1Q_0 \\ P_0 & \to & BP_0B \mid 0Z\# \\ P_1 & \to & BP_1B \mid 1Z\# \\ Q_0 & \to & 0Z \\ Q_1 & \to & 1Z \\ Z & \to & BZ \mid \varepsilon \\ B & \to & 0 \mid 1 \end{array}$$

  Finalmente la gramática para este lenguaje corresponde a la unión de ambas gramáticas. Notar que la segunda gramática no puede imponer la restricción de que el largo de ambos strings sea igual y por lo tanto la intersección de los lenguajes generados por cada gramática es no vacío, implicando que la unión genera una gramática ambigua.

(b) $L = \{x\#y\#x \mid x, y \in \{0,1\}^*$ y $x \neq y\}$.

Solución:

Este lenguaje no es libre de contexto (intuitivamente, porque reconoceríamos ww). En efecto, demostraremos que no es libre de contexto usando el teorema de bombeo.

Sea P el largo de bombeo de L (suponiendo que es libre de contexto). Escogemos la palabra $w = 0^{P!+P}\#0^P\#0^{P!+P}$. Mostraremos que independiente de la partición $w = abcde$ elegida, existirá una palabra obtenida al bombear que no estará en el lenguaje. Las condiciones que toda partición debe cumplir son $|bcd| \leq P$ y $|bd| \geq 1$.

¿Qué particiones son posibles? Como $|bcd| \leq P$, es imposible que bcd contenga los dos símbolos #, por lo tanto necesariamente está a la izquierda del # derecho, a la derecha del # izquierdo, o entremedio de ambos. Primero descartaremos un caso trivial: si $|bd|$ contiene un #, entonces, al bombear, la palabra obtenida no estará en L (pues toda palabra en L debe tener exactamente dos #). Eso nos deja 3 casos:

- $|bd|$ contiene ceros del lado izquierdo de los # (es decir, ceros del x izquierdo) y posiblemente ceros entre los # (en y).
- $|bd|$ contiene ceros del lado derecho de los # (es decir, ceros del x derecho) y posiblemente ceros entre los # (en y).
- $|bd|$ sólo contiene ceros entre los símbolos #, es decir, está completamente contenido en y.

En el primer y segundo caso, basta bombear con un $i \neq 1$ cualquiera para que el lado izquierdo del # izquierdo y el lado derecho del # difieran. Luego, el tercer caso es el único por analizar.

En el tercer caso, considere $|bd| = k$. Luego, la palabra bombeada será de la forma:

$$w_i = 0^{P!+P}\#0^{P+k(i-1)}\#0^{P!+P}$$

donde basta elegir $i - 1 = \frac{P!}{k}$ para que $x = y$ y por lo tanto obtener así una palabra que no pertenece al lenguaje. Notemos que el valor de $\frac{P!}{k}$ es al menos 1 (y por lo tanto el valor de i elegido es al menos 2).

De esta forma, reemplazando $i = 1 + \frac{P!}{k}$ se obtiene:

$$w_i = 0^{P!+P}\#0^{P!+P}\#0^{P!+P}$$

que no pertenece al lenguaje pues $x = y$. Por lo tanto no existe partición que cumpla bombeo para la palabra elegida, y por lo tanto el lenguaje L no cumple el teorema de bombeo.

En conclusión, L no es libre de contexto.