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

Solución Control 1

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 sobre el alfabeto $\{1, 2\}$ definido de la siguiente forma

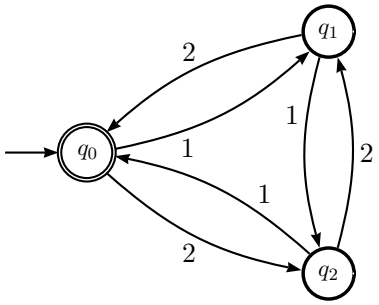
$$L = \{a_1 a_2 \cdots a_k \mid a_1, a_2, \ldots a_k \in \{1, 2\} \text{ y } a_1 + a_2 + \cdots + a_k \text{ es un múltiplo de } 3\}.$$

Por ejemplo el string 12212 no está en L ya que $1 + 2 + 2 + 1 + 2 = 8$ que no es un múltiplo de 3. Por otro lado el string 21222111 si está en L ya que $2 + 1 + 2 + 2 + 2 + 1 + 1 + 1 = 12$ que sí es un múltiplo de 3. Demuestre que L cumple el Lema de Bombeo. Si lo necesita puede suponer que ε está en L.

Solución:

Presentaremos dos formas para hacer este problema.

(a) Como sabemos que todo lenguaje regular cumple con el lema de bombeo, basta demostrar que el lenguaje es regular. En efecto, es fácil pensar en un autómata que acepta el lenguaje pues sólo necesitamos recordar la suma de los dígitos módulo 3. El autómata tiene tres estados pues el resultado del módulo puede ser 0, 1 o 2. A continuación se muestra un diagrama de estados del autómata.



Notar que hacemos uso de la indicación considerando que $\varepsilon \in L$. No sería difícil modificar este autómata para que no acepte ε.

(b) Otra forma de enfrentar el problema es mostrar directamente que cumple con las condiciones del lema de bombeo. Demostraremos que el lenguaje cumple las condiciones para $N \geq 3$.

Sea $w = w_1 w_2 \ldots w_n$ una palabra en el lenguaje de largo mayor o igual que 3 y consideremos los tres primeros símbolos. Podemos distinguir dos casos

- **Los tres primeros símbolos son iguales.**
  En este caso basta tomar como partición $x = \varepsilon, y = w_1w_2w_3, z = w_4w_5 \dots w_n$. No es difícil darse cuenta que xy^iz pertenece al lenguaje para cualquier i pues al bombear estaríamos agregando a la suma una cantidad que es múltiplo de tres y por lo tanto no estaríamos cambiando el valor del módulo, además si $|w| = 3$ al bombear con $i = 0$ nos quedaría la palabra vacía y por la indicación podemos asumir que está en el lenguaje.
- **Hay al menos un 2 y un 1 en los tres primeros símbolos**
  En este caso basta con que en la partición el substring y contenga un 1 y un 2 para que el mismo argumento aplique.

Notar nuevamente que en este caso tampoco es necesario hacer el supuesto de que ε pertenece al lenguaje pues en caso contrario solo habría que escoger $N = 4$ para que el mismo argumento funcione.

P2. Un *Super Autómata Finito* (SAF) sobre Σ es similar a un autómata finito usual con la diferencia de que cada transición, en vez de estar etiquetada con un símbolo de Σ, está etiquetada con una expresión regular sobre Σ. Por ejemplo en un SAF puede existir una transición de la forma $\delta(p, a^*ba^*) = q$. Esta última transición indica que si el SAF está en p, puede pasar a q procesando *cualquier string* que sea generado por la expresión regular a^*ba^*. Por ejemplo el autómata podría pasar de p a q procesando el string *aaaba*. Note que por ser finito, independiente de que existan infinitas expresiones regulares posibles, el SAF sólo puede tener una cantidad finita de transiciones desde cada estado. Formalmente, para un SAF $A = (Q, \Sigma, \delta, q_0, F)$ la relación de *computa en un paso* se define por:

$$(q, w) \vdash_A (p, v) \quad \text{si existe } x \text{ tal que } w = xv,\ \delta(q, r) = p \text{ y } x \in L(r).$$

Desde esta relación se puede definir la relación de *computa en 0 o más pasos*, $\vdash_A^*$, tal como lo hicimos en clases. Finalmente, decimos que un string w es aceptado por el SAF A si

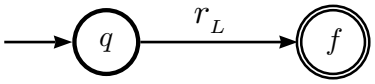
$$(q_0, w) \vdash_A^* (p, \varepsilon) \quad \text{para algún } p \in F.$$

El lenguaje aceptado por un SAF es el conjunto de todos los strings aceptados por el SAF.

(a) Demuestre que todo lenguaje regular puede ser aceptado por un SAF con exactamente dos estados.

Solución:

Sea L un lenguaje regular. Como L es regular, existe una expresión regular $r_{_L}$ que lo genera. Luego, un SAF de dos estados que acepta L es $(\{q, f\}, \Sigma, \delta, q, \{f\})$, con $\delta(q, w) = q_f$ si $w \in \mathcal{L}(r_{_L}) = L$.



(b) Demuestre que si un lenguaje L es aceptado por un SAF, entonces L es regular.

Solución:

Hay muchas formas de demostrar que todo lenguaje reconocido por un SAF es regular.

- Una forma, es notar que al reemplazar las expresiones regulares $r_1, r_2, \cdots$ en las transiciones del SAF por símbolos $s_1, s_2, \cdots$ diferentes obtenemos un AFND A. Dada la equivalencia entre expresiones regulares y autómatas, existe una expr. regular r_A que genera el lenguaje aceptado por A. Esta expresión regular estará formada por $s_1, s_2, \cdots$, y concatenaciones, disyunciones y estrellas de Kleene de éstos. Por último, si reemplazamos cada s_i por la expresión que reemplazó en el SAF, obtenemos una expresión regular que, se puede mostrar que genera el lenguaje del SAF. Como no asumimos nada sobre el SAF inicial, entonces hemos demostrado que todo SAF acepta un lenguaje regular.
- Otra forma, es notar que para la expresión regular r_i de cada transición entre p_i y q_i de un SAF, existe un autómata A_i que acepta exactamente el lenguaje generado por r_i. Es fácil ver que al reemplazar una transición del SAF entre p_i y q_i por una transición-ε de p al estado inicial de A_i, y transiciones- ε desde los estados finales de A_i (si existen) a q_i, obtenemos un SAF equivalente. Si repetimos esto para cada transición etiquetada por una expresión regular, obtendremos un AFND equivalente al SAF inicial, por lo que el lenguaje aceptado por el SAF es regular. Luego, todo SAF acepta un lenguaje regular.
- Una tercera forma es notar que el SAF es un estado intermedio del procedimiento mediante el cual se puede construir una expresión regular que genera el lenguaje aceptado por un AF. Si se continúa con este procedimiento, se obtendrá un SAF de dos estados, donde la expresión regular que etiqueta la transición del estado inicial al final corresponde a la expresión regular que genera el lenguaje aceptado por el SAF. Luego, el lenguaje aceptado por el SAF es regular, y por lo tanto, todo SAF acepta un lenguaje regular.

P3. Demuestre que cada uno de los siguientes lenguajes no son regulares.

(a) $L = \{a^i b^j c^k \mid \text{ si } i \neq 0 \text{ entonces } j = k\}$

Solución:

Tratar de demostrar que este lenguaje no es regular usando directamente el lema de bombeo no nos llevará a ningún lado pues el lenguaje lo cumple. Intentaremos utilizando otras propiedades sobre los lenguajes regulares.

Analizando un poco el lenguaje nos podemos dar cuenta que lo que nos "molesta" para poder usar el lema de bombeo son las a's al inicio de los strings. Y bueno, si nos molestan, tratemos de sacarlas. Lo primero que hay que notar es que no podemos sacarlas completamente por que la parte que hace no regular al lenguaje se da cuando el número de a's es distinto de 0. Entonces como no podemos sacarlas tratemos de fijarlas a algún número que sea distinto de 0. Para hacer esto veremos que pasa si lo intersectamos con el lenguaje $M = ab^*c^*$. Dado que este lenguaje es regular, y considerando que los lenguajes regulares son cerrados bajo intersección, al demostrar que $L \cap M$ no es regular también demostraríamos que L no lo es.

Es claro que la intersección es $L \cap M = \{ab^k c^k | k \geq 0\}$. Demostraremos ahora que este lenguaje no cumple el lema de bombeo y por lo tanto no es regular. En efecto, sea N un natural cualquier (el largo de la constante de bombeo) y consideremos la palabra $w = ab^N c^N$. Si en una partición $w = xyz$, y sólo contiene la a para cualquier i el bombeo sacaría a la palabra del lenguaje pues estaríamos cambiando el número de a's. Ahora, una partición nunca podrá contener b's y c's a la vez pues $|xy|$ tiene que ser menor o igual que N. En este caso bombear la partición para cualquier $i \neq 1$ hará que el número de b's y c's sea distinto y por lo tanto sacará a la palabra del lenguaje.

Concluyendo, como $L \cap M$ no cumple bombeo es entonces regular y como M es regular no puede ser que L sea también regular.

Podría haberse hecho una análisis similar pero tomando L^R en vez de la intersección con ab^*c^*.

(b) $L = \{a^i b^j \mid i \neq 5j\}$

Solución:

Trataremos de demostrar que este lenguaje no es regular mostrando directamente que no cumple el lema de bombeo. Sea entonces $N > 0$ una constante cualquiera y tomemos la palabra $w = a^{N!}b^{N!}$ que claramente está en el lenguaje. Toda partición será de la forma $x = a^p, y = a^q, z = a^r b^{N!}$ con $p+q+r = N!$. Si bombeamos estas particiones obtendremos $w_i = a^p a^{iq} a^r b^{N!} = a^{N!+(i-1)q} b^{N!}$. Hay que escoger i tal que $N!+(i-1)q = 5N!$ para sacar a la palabra del lenguaje. Esto se cumple para $i = \frac{4N!}{q} + 1$ que es siempre un entero pues $0 < q \leq N$. Con esto hemos demostrado que el lenguaje no cumple el lema de bombeo y por lo tanto no es regular.

P4. Javiera y sus amigos creen en el amor a "primera cita" y han decidido hacer un pacto. Pueden tener citas con muchas parejas distintas, pero luego de la primera cita con la "pareja perfecta", harán un acuerdo de unión civil con ella y no volverán a tener otra cita en toda su vida. Por el contrario, si luego de una cita deciden que esa persona no era su pareja perfecta, podrán volver a tener una cita con ella muchas veces, claro, esto para no aburrirse mientras encuentran a su pareja perfecta. Suponga que las posibles personas con las que Javiera y sus amigos pueden tener citas son Amanda (A), Bernardo (B), Constanza (C), Daniel (D), Enzo (E) y Fernanda (F). Entonces, por ejemplo, una posible secuencia de citas antes de hacer el acuerdo de unión

civil sería $BBBAACABFD$. Note que la última cita fue con Daniel luego de lo cuál no hubo más citas, y por lo tanto, la unión civil fue con Daniel. Por el contrario la secuencia $EAEABBBDFA$ no es una secuencia de citas válida pues la última cita fue con Amanda y ya había una cita con Amanda en el pasado. Construya un autómata sobre el alfabeto $\{A, B, C, D, E, F\}$ que acepte todos los strings que representan secuencias de citas válidas para Javiera y sus amigos antes de contraer el acuerdo de unión civil.

Solución:

En primer lugar notemos que para verificar que una cita sea la cita perfecta, la única condición es que esa cita sea la primera con esa persona, y la última cita.

Luego, nuestros estados deben indicar con quiénes ha salido Javiera o el amigo que está buscando la cita perfecta y posterior acuerdo de unión civil[1].

Por ello, los estados de nuestro AFD serán de la forma $c_A c_B c_C c_D c_E c_F X$, donde $c_i \in \{0, 1\}$ indica si Javiera ha salido con la persona representada por $i \in \{A, B, C, D, E, F\}$, y X indica si la última persona con la que salió había sido una cita previa de Javiera o no.

Luego, definimos el AFD como $(Q, \{A, B, C, D, E, F\}, \delta, 0000000, F)$ con:

- $Q = \{\{0, 1\}^7\}$
- $F = \{c_A c_B c_C c_D c_E c_F 1 \mid c_i \in \{0, 1\}\}$
- $\delta(c_A c_B c_C c_D c_E c_F X, \sigma) = \begin{cases} c'_A c'_B c'_C c'_D c'_E c'_F 1 & \text{si } c_\sigma = 0 \\ c'_A c'_B c'_C c'_D c'_E c'_F 0 & \text{si } c_\sigma = 1 \end{cases}$ con $c'_i = \begin{cases} 1 & \text{si } i = \sigma \\ c_i & \text{si no} \end{cases}$

Opcionalmente podemos eliminar el estado 0000001, que es inalcanzable, dando lugar a 127 estados. Notemos que un AFD así no es dibujable de forma razonable en un control, por lo que se esperaba una definición como la anterior.

[1] En adelante, y sin pérdida de generalidad, supondremos que las citas son de Javiera.-