NOMBRE: Benjamín Farías Valdés

N.ALUMNO: 17642531

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencias de la Computación

IIC2223 — Teoría de Autómatas y Lenguajes Formales — 2' 2021

Tarea 2

Pregunta 1

a)

La expresión regular propuesta es la siguiente:

$$R = (b+c)^* \cdot (ab \cdot (b \cdot (b+c)^*)? + a \cdot (c \cdot (b+c)^*)?)^*$$

El primer paréntesis $(b+c)^*$ permite que al comienzo de la palabra exista cualquier cantidad de letras b y c, dado que la subpalabra abc **no será encontrada hasta que aparezca la primera letra** a que podría potencialmente generarla (así también es posible tener palabras que no contienen a la letra a).

El segundo paréntesis representa un patrón que puede ser repetido cualquier cantidad de veces (incluso 0), el que será explicado por partes a continuación:

- $R_1 = ab \cdot (b \cdot (b+c)^*)?$: Esta expresión comienza con la subpalabra ab, lo que significa que debemos evitar que siga con una c, puesto que se completaría la subpalabra abc. Por lo tanto, se da el paréntesis opcional $(b \cdot (b+c)^*)?$. Si se utiliza este paréntesis, entonces **se obliga a que aparezca una** b**,** lo que completa la subpalabra abb, y por lo tanto se logró evitar la aparición de la subpalabra abc hasta que se vuelva a encontrar una nueva a que pueda generarla. Debido a esto, se permite tener cualquier cantidad de letras b y c después de completar esta subpalabra abb. Si se elige NO utilizar el paréntesis, entonces se verá un poco más adelante que esto obliga a que la siguiente letra sea una a, o bien, a que la palabra termine, evitando la subpalabra abc en ambos casos.
- $R_2 = a \cdot (c \cdot (b+c)^*)?$: Esta expresión comienza con la letra a, y opcionalmente puede continuar con una c seguida de cualquier cantidad de letras b y c. Si se toma esta opción, se forma la subpalabra ac, lo que **inmediatamente evita que aparezca la subpalabra** abc**,** puesto que faltaría la b entre medio. Si no se toma la opción, se verá más adelante que esto obliga a que la siguiente letra sea una a (formando la subpalabra aa), o bien, a que la palabra termine, evitando la subpalabra abc en ambos casos.

Utilizando estas sub-expresiones R_1 y R_2, se forma el segundo paréntesis, quedando:

$$(R_1 + R_2)^*$$

Esto permite formar subpalabras que parten con la letra a indefinidamente, con la condición de que nunca aparezca abc al comienzo. De hecho, cada iteración sobre este paréntesis puede formar **alguna de las siguientes expresiones regulares:**

$$Exp = \{a, ab, abb \cdot (b+c)^*, ac \cdot (b+c)^*\}$$

Sin importar la cantidad de iteraciones realizadas, es imposible formar la subpalabra abc concatenando las expresiones del conjunto Exp, **ya que todas ellas comienzan con la letra a y no permiten la subpalabra abc.** Aun así, cualquier otra combinación es permitida, debido a que sólo se controla que comience con a y que no llegue hasta abc en cada iteración.

Finalmente, se llega entonces a la expresión inicialmente propuesta:

$$(b+c)^* \cdot (R_1 + R_2)^* = (b+c)^* \cdot (ab \cdot (b \cdot (b+c)^*)? + a \cdot (c \cdot (b+c)^*)?)^* = R$$

Esta expresión **permite tener palabras sólo con letras b y c, o que tengan a pero sin formar la subpalabra abc.**

b)

La expresión regular propuesta es la siguiente:

$$R = (b+c)^* \cdot (a \cdot ((c^*b)^2 \cdot (b+c)^* \cdot a)^* \cdot c^*b \cdot (b+c)^*)?$$

El primer paréntesis $(b+c)^*$ permite que al comienzo de la palabra exista cualquier cantidad de letras b y c, dado que las condiciones del lenguaje **no aplican hasta que aparezca la primera a.**

El segundo paréntesis es opcional, ya que es necesario ignorarlo si se desea formar una palabra que NO contenga letras a. **Para cualquier otro caso,** se explicará la estructura de este paréntesis por partes:

- $R_1 = (c^*b)^2 \cdot (b+c)^* \cdot a$: Esta expresión comienza repitiendo 2 veces un patrón de cantidad variable de letras c seguidas de una b. Esto **asegura que habrán al menos 2 letras b.** Luego, se permite cualquier cantidad de letras b y c, llegando eventualmente a una a. Más adelante se verá que justo antes de esta expresión siempre habrá una letra a, por lo que **la función de esta parte es obligar a que existan al menos 2 letras b entre medio de cada par de letras a.** Además, se satisface trivialmente la otra condición (cada a es eventualmente seguida por una b), puesto que entre las letras a hay al menos 2 letras b.

- $R_2 = c^*b \cdot (b+c)^*$: Esta expresión comienza con una cantidad variable de letras c, eventualmente llegando a una b, para luego terminar con cualquier cantidad de letras b y c. Más adelante se verá que justo antes de esta expresión se encuentra la última letra a de la palabra, por lo que **la función de esta parte es asegurar que se cumpla que eventualmente hay una b después de esa última** a (efectivamente evitando que la palabra termine en una a). Además, se satisface trivialmente la otra condición (al menos 2 letras b entre cada par de letras a), puesto que ya se visitó la última a de la palabra y no existen más pares de letras a.

Utilizando estas sub-expresiones R_1 y R_2, se forma el segundo paréntesis, quedando:

$$(a \cdot R_1^* \cdot R_2)$$

Para este paréntesis, existen 2 posibilidades, dependiendo de la cantidad n de iteraciones realizadas sobre R_1:

- $n = 0$: En este caso la expresión equivale a $(a \cdot R_2)$. Esto satisface la primera condición (a seguida eventualmente por una b), ya que R_2 tiene al menos una b y no contiene letras a. También satisface trivialmente la segunda condición (al menos 2 letras b entre cada par de letras a), ya que no hay ningún par de letras a.

- $n > 0$: En este caso se comienza con una letra a, y cada iteración sobre R_1 formará un patrón que contiene al menos 2 letras b seguidas eventualmente por una a. Es decir, las palabras definidas por la expresión adquieren la siguiente forma:

  $$a \cdot (...b...b...a)^* \cdot R_2$$

  Si además consideramos que R_2 asegura la existencia de al menos una b y evita la aparición de más letras a, las palabras quedan de la siguiente forma:

  $$a \cdot (...b...b...a)^*...b...(b + c)$$

  Al observar esto, es claro que se satisface la segunda condición, ya que sólo la primera letra a es libre, mientras que todas las demás son precedidas por al menos 2 letras b, **cumpliendo con que existen al menos 2 letras b entre cada par de letras a.** Además, como no existen letras a en la última parte de la expresión (definida por R_2), y esta obliga a que haya al menos una b, se tiene que esa letra b es sucesora de todas las letras a de la palabra, **cumpliendo así con que cada letra a está seguida eventualmente por una b.**

Podemos ver que en todos los casos se satisfacen las condiciones del lenguaje.

Finalmente, se llega entonces a la expresión inicialmente propuesta:

$$(b + c)^* \cdot (a \cdot R_1^* \cdot R_2)? = (b + c)^* \cdot (a \cdot ((c^*b)^2 \cdot (b + c)^* \cdot a)^* \cdot c^*b \cdot (b + c)^*)? = R$$

Esta expresión **permite tener palabras sólo con letras b y c, o que tengan a y cumplan ambas condiciones pedidas.**

NOMBRE: Benjamín Farías Valdés

N.ALUMNO: 17642531

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencias de la Computación

IIC2223 — Teoría de Autómatas y Lenguajes Formales — 2' 2021

Tarea 2

Pregunta 2

El lenguaje L corresponde al de todas las palabras formadas por al menos 2 subpalabras en $\{a, b\}^*$ separadas entre sí mediante símbolos $\#$, tales que son todas distintas entre sí. Este lenguaje **NO es regular**, y la demostración se realizará mediante el **contrapositivo del lema de bombeo.**

Sea $w = u_1\#u_2\#...\#u_k$, con $u_j = a^{N+j-1}$ para todo $j = 1...k$, y con $k > N$.

Para todo $N > 0$:

- $u_j \in \{a, b\}^*$, ya que cada subpalabra u_j está compuesta solamente de letras a.
- Como $N > 0$ y N es un número natural, se tiene que $N \geq 1$, y como $k > N \geq 1$, debe cumplirse que $k \geq 2$.
- Para todo par de subpalabras (u_i, u_j), con $i \neq j$, se tiene que $N + i - 1 \neq N + j - 1$, luego $a^{N+i-1} \neq a^{N+j-1}$ y por lo tanto $u_i \neq u_j$ para todo $i \neq j$.

Viendo las condiciones satisfechas arriba, es claro que $w \in L$. Entonces, tomamos:

$$x = \varepsilon$$

$$y = u_1$$

$$z = \#u_2\#...\#u_k$$

Luego, toda división de y tendrá la forma:

$$y = u \cdot v \cdot w = a^b a^c a^d$$

Donde $b + c + d = N$ y $1 \leq c \leq N$.

Tomando $i = 2$, tendremos:

$$x \cdot u \cdot v^i \cdot w \cdot z = a^{N+c}\#a^{N+1}\#...\#a^{N+k-1}$$

Esta palabra **NO pertenece al lenguaje** L, lo que será demostrado a continuación.

Como $1 \leq c \leq N$, tenemos que $N+1 \leq N+c \leq 2N$, y además sabemos que $k > N$ por construcción. Entonces se desprende que $k + N > 2N$, lo que implica que $k + N - 1 \geq 2N$, y finalmente tenemos:

$$N + 1 \leq N + c \leq N + k - 1$$

Esto último significa que el valor de $N + c$ corresponde a un número natural entre $N + 1$ y $N + k - 1$ (incluyéndolos).

Entonces, demostraremos por inducción que para todo número natural m, tal que $N+1 \leq m \leq N+k-1$, existe un u_j en la palabra w tal que se tiene $u_j = a^m$ y $2 \leq j \leq k$.

- **Caso Base:** Para $m = N + 1$, por construcción sabemos que $u_2 = a^{N+1} = a^m$, es decir, se cumple con $j = 2$.
- **Caso Inductivo:** Si para $m < N+k-1$ se cumple que existe un j tal que $2 \leq j < k$ y $u_j = a^{N+j-1} = a^m$, entonces $m = N + j - 1$, y por lo tanto:

  $$u_{j+1} = a^{N+(j+1)-1} = a^{N+j} = a^{m+1}$$

  Lo que demuestra lo indicado arriba.

Gracias a esto último, tenemos que para todo valor posible de $N + C$, existe un $2 \leq j \leq k$ tal que $u_j = a^{N+C}$, lo que satisface:

$$i = 1$$

$$i \neq j$$

$$u_i = u_j$$

Es decir, se encontró un par de subpalabras con distinto índice que eran iguales, y por lo tanto se rompe la condición del lenguaje L y finalmente se tiene que $x \cdot u \cdot v^i \cdot w \cdot z \notin L$ para cualquier valor de b, c y d escogidos en la división de y, siempre que se escoja un $k > N$ en la construcción de w.

De esta forma, queda demostrado que el lenguaje L **NO es regular**.