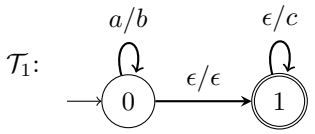
Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

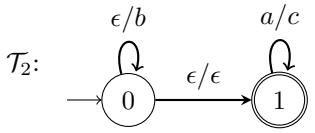
IIC2223 — Teoría de Autómatas y Lenguajes Formales — 2' 2022

PAUTA INTERROGACIÓN 2

Pregunta 1

Una posible solución para esta pregunta es construyendo los siguientes transductores $\mathcal{T}_1$ y $\mathcal{T}_2$, con $\Sigma = \{a\}$ su alfabeto de input y $\Omega = \{b, c\}$ su alfabeto de output:





Es fácil ver que: $[|\mathcal{T}_1|] = \{(a^n, b^n c^m) \mid n, m \geq 0\}$, y que $[|\mathcal{T}_2|] = \{(a^n, b^m c^n) \mid n, m \geq 0\}$. Luego, tenemos que $[|\mathcal{T}_1|] \cap [|\mathcal{T}_2|] = \{(a^n, b^n c^n) \mid n \geq 0\}$.

Por contradicción, supongamos que $[|\mathcal{T}_1|] \cap [|\mathcal{T}_2|]$ es una relación racional. Se tiene que $\pi_2([|\mathcal{T}_1|] \cap [|\mathcal{T}_2|]) = \{b^n c^n \mid n \geq 0\}$, y notamos que **no** es un lenguaje regular. Por teorema visto en clases, sabemos que las proyecciones de las relaciones racionales son lenguajes regulares. Luego, llegamos a una contradicción, y queda demostrado que $[|\mathcal{T}_1|] \cap [|\mathcal{T}_2|]$ no es una relación racional.

Dado lo anterior, la distribución de puntaje es la siguiente:

- **(1.5 puntos)** Por construir $\mathcal{T}_1$
- **(1.5 puntos)** Por construir $\mathcal{T}_2$
- **(1 puntos)** Por expresar la relación definida por $[|\mathcal{T}_1|] \cap [|\mathcal{T}_2|]$
- **(1 puntos)** Por identificar que $\pi_2([|\mathcal{T}_1|] \cap [|\mathcal{T}_2|])$ no es un lenguaje regular
- **(1 puntos)** Por concluir que $[|\mathcal{T}_1|] \cap [|\mathcal{T}_2|]$ no es una relación racional

Nota: Es correcto proponer otros transductores $\mathcal{T}_1$ y $\mathcal{T}_2$, mientras cumplan que $\pi_1([|\mathcal{T}_1|] \cap [|\mathcal{T}_2|])$ no sea un lenguaje regular, o que $\pi_2([|\mathcal{T}_1|] \cap [|\mathcal{T}_2|])$ no sea un lenguaje regular.

Pregunta 2

Sean los lenguajes L_1 y L_2 definidos de la siguiente forma:

$$L_1 = \{a^i b^j c^k d^l \mid 2i = l \wedge 3j = k\}$$
$$L_2 = \{a^i b^j c^k d^l \mid 2i = k \wedge 3j = l\}$$

En este caso, L_1 es un lenguaje libre de contexto mientras que L_2 no lo es. A continuación se demuestran ambas afirmaciones.

1) L_1 es LLC:

Para demostrar que L_1 es libre de contexto, procedemos a diseñar una gramática para el lenguaje y explicar su correctitud. La gramática generada es de la forma:

$$\mathcal{G} : X \to aXdd \mid Y$$
$$Y \to bYccc \mid \epsilon$$

La gramática se construye de forma recursiva desde los extremos hasta el centro. Nos aseguramos con las derivaciones de X de agregar las letras a y d en los extremos de la palabra correspondientes, procurando seguir la proporción pedida. Luego, al cambiar de variable a Y, agregamos en el centro las letras b y c que también siguen su propia proporción pedida. Finalmente, solo es posible seguir agregando b y c, o terminar la palabra.

2) L_2 no es LLC:

Para demostrar que L_2 no es LLC, usaremos el contrapositivo del lema de bombeo para lenguajes libres de contexto. Sea un $N > 0$, definimos la siguiente palabra z perteneciente al lenguaje:

$$z = a^N b^N c^{2N} d^{3N}$$

Y sea la siguiente descomposición cualquiera para z:

$$z = uvwxy$$

Con $vx \neq \epsilon$ y $|vwx| \leq N$. Según la estructura que tome dicha descomposición, podemos tener la siguientes tres posibilidades:

1. $vwx \in a^*b^*$. En este caso, sea $i = 0$ de forma que la palabra queda como:

$$uv^0wx^0y = a^{N_1} b^{N_2} c^{2N} d^{3N} \tag{1}$$

   Lo que fuerza a que $N_1 < N$ ó $N_2 < N$, por lo que la palabra nueva no pertenece a L_2.

2. $vwx \in b^*c^*$. Es este caso, sea $i = 0$:

$$uv^0wx^0y = a^{N} b^{N_2} c^{N_3} d^{3N} \tag{2}$$

   Lo que fuerza a que $N_2 < N$ ó $N_3 < 2N$, por lo que la palabra nueva no pertenece a L_2.

3. $vwx \in c^*d^*$. Es este caso, sea $i = 0$:

$$uv^0wx^0y = a^{N} b^{N} c^{N_3} d^{N_4} \tag{3}$$

   Lo que fuerza a que $N_3 < 2N$ ó $N_4 < 3N$, por lo que la palabra nueva no pertenece a L_2.

Al ser analizadas todas las posibilidades de descomposición para la palabra z, se demuestra que L_2 no es un lenguaje libre de contexto.

Dado lo anterior, la distribución de puntaje es la siguiente:

- **(0.5 puntos)** Por indicar que L_1 es LLC y que L_2 no lo es.
- **(2 puntos)** Por generar la gramática correspondiente para L_1.
- **(0.75 puntos)** Por explicar la correctitud de la gramática generada.
- **(1 punto)** Por seleccionar una palabra z adecuada para el lema de bombeo de L_2.
- **(0.25 puntos)** Por seleccionar la descomposición $z = uvwxy$ adecuada, junto a sus restricciones.
- **(1.5 puntos)** Por analizar correctamente todas las posibilidades de la descomposición anterior de z.

Pregunta 3

Pregunta 3

Sea L un lenguaje libre de contexto y el lenguaje $h(L)$ definido como:

$$h(L) = \{h(w) \mid w \in L\}.$$

Sea $\mathcal{G} = \{V, \Sigma, P, S\}$ en forma normal de Chomsky (CNF) tal que $L(\mathcal{G}) = L \setminus \{\epsilon\}$, por teorema visto en clases. Se define $h(\mathcal{G}) = (V \cup \{S^{'}\}, \Omega, P^{'}, S^{'})$ tal que:

$$\begin{aligned} P^{'} = \quad & \{X \to YZ \mid X \to YZ \in P\} \\ \cup \quad & \{X \to h(a) \mid X \to a \in P\} \\ \cup \quad & \{S^{'} \to S\} \\ \cup \quad & \{S^{'} \to \epsilon \mid \epsilon \in L\} \end{aligned}$$

Por demostrar: $h(L) = L(h(\mathcal{G}))$

($\subseteq$) Si $u \in h(L)$, sea $w \in L$ tal que $h(w) = u$. Como $w \in L(\mathcal{G})$ sea T un árbol de derivación de $\mathcal{G}$ sobre w.

Sea $h(T)$ el árbol T donde reemplazando cada hoja $a \in \Sigma$ por $h(a)$. Entonces, por construcción es fácil ver que $h(T)$ es árbol de derivación de $h(\mathcal{G})$ con $h(w)$.

Si $u = \epsilon \Rightarrow S \Rightarrow \epsilon$ y $\epsilon \in L(h(\mathcal{G}))$.

($\supseteq$) Sea $u \in L(h(\mathcal{G}))$ y sea T árbol de derivación de $h(\mathcal{G})$ sobre u. Sea $T^{'}$ el árbol tal que para cada nodo de la forma $X(b_1 \dots b_k)$ lo reemplazamos por $X(a)$ con $a \in \Sigma$ tal que $h(a) = b_1 \dots b_k$. Sea w la palabra en las hojas de $T^{'}$:

- $h(w) = u$
- $T^{'}$ es árbol de derivación de $\mathcal{G}$ sobre w

Por lo tanto, $u \in h(L)$.

Dado lo anterior, la distribución de puntaje es la siguiente:

- **(0.5 puntos)** Por manejar correctamente la función h (respeta la concatenación)
- **(1.5 puntos)** Por conservar las derivaciones del tipo $X \to YZ$
- **(1.5 puntos)** Por conservar las derivaciones del tipo $X \to a$
- **(0.5 puntos)** Por incluir el ϵ en la gramática
- **(2 puntos)** Por demostrar correctamente cada lado (1 punto c/u)

Pregunta 4

Una posible solución para esta pregunta es modificar el algoritmo *eval-NFAonthefly* visto en clases. Específicamente, en lugar de llevar dos conjuntos S y S_{old} para llevar los estados en los que van las ejecuciones al leer la i-ésima letra de w, se llevará un **contador** de los sufijos que hay en cada estado al leer la i-ésima letra. En el nuevo algoritmo, S y S_{old} corresponderán a arreglos de tamaño $|Q| = m$, donde cada entrada del arreglo le corresponde a un estado $j \in Q$, y esta llevará la cuenta de sufijos de la palabra que al ejecutar $\mathcal{A}$ llegan al estado j.

Sin pérdida de generalidad, suponga que el autómata $\mathcal{A}$ tiene como conjunto de estados $Q = \{0, 1, \ldots, m-1\}$. Si los m estados de $\mathcal{A}$ no son números, estos pueden ser ordenados arbitrariamente y ser asignados a números. También suponemos, sin pérdida de generalidad, que $\mathcal{A}$ es un autómata sin ϵ-transiciones. Luego, escribimos el siguiente algoritmo:

#SUFFIX-DFA($\mathcal{A} = (Q, \Sigma, \Delta, I, F)$, $w = a_1 a_2 \ldots a_n$)

```
S, S_old ← Arreglos de largo m, y entradas con valor 0
for i = 1 to n do
    S_old ← S
    S ← [0]                                  // arreglo de 0s
    S_old[0] ← S_old[0] + 1
    for j = 0 to m − 1 do
        S[δ(j, a_i)] ← S[δ(j, a_i)] + S_old[j]
    end for
end for
C ← 0
for j ∈ F do
    C ← C + S[j]
end for
return C
```

Para la demostración de la correctitud del algoritmo, usaremos inducción sobre el tamaño de w, para mostrar que en toda iteración el arreglo S lleva correctamente la cantidad de sufijos de $a_1 ... a_{i-1}$ que llegan a cada estado.

CB: Antes de empezar a leer letras de w, se da que el estado inicial de $\mathcal{A}$ no tiene sufijos y no se ha pasado por ningún otro estado. Esto se ve representado correctamente al inicializar las entradas de S como 0.

HI: Suponemos que luego de procesar $a_1 \cdots a_i$ se cumple que, para todo estado j, $S[j]$ es igual al número de sufijos de $\mathcal{A}$ sobre $a_1 \cdots a_i$ que llegan al estado j.

TI: Por HI, S lleva correctamente la cantidad de ejecuciones por estado en la i-ésima iteración. En la iteración $i + 1$, se asigna a S_{old} los valores de S, y S se reinicializa con valores 0. Luego, por cada transición (j_1, a_{i+1}, j_2) del autómata, a $S[j_2]$ se le suma la cantidad de sufijos de j_1. Al realizar esto sobre todas las transiciones del autómata, se cumple que $S[j]$ es igual al número de sufijos de $a_1 \cdots a_{i+1}$ al ejecutar $\mathcal{A}$ que llegan a $j \in Q$.

Ya demostrado que el S lleva correctamente la cantidad de sufijos que llegan a cada estado para cualquier largo de w. La cantidad de sufijos de w que son aceptados por $\mathcal{A}$ será la suma de la cantidad de sufijos de las ejecuciones que llegaron a estados finales del autómata después de leer las n letras de w. Esto se guarda en la variable C al final del algoritmo, y se retorna dicho valor. Luego, el algoritmo propuesto retorna correctamente la cantidad de sufijos de w que son aceptados por $\mathcal{A}$.

Dado lo anterior, la distribución de puntaje es la siguiente:

- **(1 puntos)** Por llevar un contador para cada estado.
- **(1.5 puntos)** Por actualizar correctamente S y S_{old}
- **(1.5 puntos)** Por actualizar correctamente $S[\delta(j, a_i)]$
- **(1 puntos)** Por contar los sufijos.
- **(1 puntos)** Por demostrar correctitud.

Nota 1: No era correcto si:

- el algoritmo tiene un tiempo de ejecución mayor al pedido, por tanto se asignaron 0 puntos a esta solución.
- si solo se escribe el algoritmo NFA-onthefly‘ sin cambios, por tanto se asignaron 0 puntos a esta solución.

Nota 2: Si el autómata se “invierte” para leer la palabra de derecha a izquierda, había que tener cuidado en darse cuenta que al invertir el autómata queda no-determinista. Para esto debían demostrar que el autómata invertido acepta el mismo lenguaje y después aprovechar el autómata invertido correctamente. En muchos casos esto no ocurrió y por tanto se asignaron 0 puntos a esta solución. Se recomienda consultar al profesor, si su solución esta en este caso.