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

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

PAUTA INTERROGACIÓN 2

Pregunta 1

Pregunta 1.1

Una posible solución para esta pregunta es tomar en cuenta la siguiente propiedad:

$$i + k = j \text{ si y solo si } a^i b^j a^k = a^i b^i b^k a^k$$

Así, se propone la siguiente gramática libre de contexto:

$$\begin{gathered} S \longrightarrow AB \\ A \longrightarrow aAb|\epsilon \\ B \longrightarrow bBa|\epsilon \end{gathered}$$

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

- **(1 punto)** Por definir la propiedad.
- **(1.5 puntos)** Por definir la gramática libre de contexto.
- **(0.5 punto)** Por considerar los casos bordes.

Pregunta 1.2

Se demuestra que la gramática libre de contexto definida es correcta mediante la demostración de $\mathcal{L}(\mathcal{G}) = \text{L}$.

Por demostrar: $\text{L} \subseteq \mathcal{L}(\mathcal{G})$.

Sea $a^i b^j a^k \in \text{L}$. Luego, como $i + k = j$, entonces $a^i b^j a^k = a^i b^i b^k a^k$.
Ahora, a partir de la CFG propuesta, se tiene que:

$$\begin{gathered} A \overset{*}{\Rightarrow} a^i b^i \\ B \overset{*}{\Rightarrow} b^k a^k \end{gathered}$$

De esta manera, existe la siguiente derivación:

$$S \Rightarrow AB \overset{*}{\Rightarrow} a^i b^i B \overset{*}{\Rightarrow} a^i b^i b^k a^k$$

Y como $a^i b^j a^k = a^i b^i b^k a^k$, entonces $a^i b^j a^k \in \mathcal{L}(\mathcal{G})$.

Por demostrar: $\mathcal{L}(\mathcal{G}) \subseteq \text{L}$.

Sea $w \in \mathcal{L}(\mathcal{G})$, por lo que existe una derivación de la forma:

$$S \overset{*}{\Rightarrow} w$$

Se analiza la forma que tiene esta derivación. En primer lugar, desde S, en un paso, solo hay una posible derivación:

$$S \Rightarrow AB$$

Luego, la derivación completa por la izquierda debe tener la siguiente forma:

$$S \Rightarrow AB \underset{lm}{\Rightarrow} aAbB \underset{lm}{\Rightarrow} aaAbbB \underset{lm}{\overset{*}{\Rightarrow}} a^iAb^iB \underset{lm}{\Rightarrow} a^ib^iB \underset{lm}{\Rightarrow} a^ib^ibBa \underset{lm}{\overset{*}{\Rightarrow}} a^ib^ib^kBa^k \underset{lm}{\Rightarrow} a^ib^ib^ka^k$$

Así, como $i + k = j$, entonces $w = a^ib^ib^ka^k = a^ib^ja^k$, por lo que $w \in$ L.

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

- **(1.5 puntos)** Por demostrar que L $\subseteq \mathcal{L}(\mathcal{G})$.
- **(1.5 puntos)** Por demostrar que $\mathcal{L}(\mathcal{G}) \subseteq$ L.

Pregunta 2

Modificando el algoritmo visto en clases para la evaluación de NFA *on-the-fly*, tenemos que para un transductor $\mathcal{T} = (Q, \Sigma, \Omega, \Delta, I, F)$ tal que $\Delta \subseteq Q \times \Sigma \times (\Omega \cup \{\varepsilon\}) \times Q$ tenemos:

```
Function Eval-Transducer(T, w = a_1...a_n)
    S := hash(Q, null)
    foreach q ∈ I do
        S[q] := ε
    for i := 1 to n do
        S_old := S
        S := hash(Q, null)
        foreach p ∈ Q do
            if S_old[p] ≠ null then
                foreach (p, a_i, b, q) ∈ Δ do
                    if S[q] = null then
                        S[q] = S_old[p] · b
    foreach p ∈ F do
        if S[q] ≠ null then
            return S[q]
    return null
```

La idea del algoritmo es llevar una tabla de hash con las palabras escritas por las ejecuciones actuales que se encuentran en cada estado, e ir completándola según nos indican las transiciones del transductor.

Estas tablas comienzan con un valor *null* **distinto de epsilon**, indicando que no hay ninguna ejecución en proceso, y a continuación se inicializan con ε las entradas de los estados iniciales, indicando que hay una ejecución activa, pero que no ha escrito nada aún. Teniendo lo anterior, se recorre la palabra letra por letra, visitando los estados que tengan alguna ejecución activa ($S_{old}[p] \neq null$), se revisan todas las transiciones salientes de este estado (para la letra actual) y en caso de encontrar una transición válida se concatena la letra que indica la transición a la palabra asociada al estado de llegada. Cabe señalar que solamente se concatena esta letra si es la primera transición válida que encontramos ($S[q] \neq null$), pues buscamos que el algoritmo retorne solamente una palabra, por lo que no es necesario llevar registro de todas las posibles.

Finalmente, luego de haber leído la palabra completa, visitamos todos los estados finales, retornando la palabra escrita por el transductor en caso de encontrar alguna y retornando *null* en caso de que ninguna ejecución llegue a un estado final.

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

- **(1 punto)** Por llevar una tabla de hash o similar para las ejecuciones (en lugar de un conjunto).
- **(1 punto)** Por crear la tabla con nulos e inicializar con ε para las variables iniciales.
- **(1.5 puntos)** Por iterar sobre los estados que llevan alguna ejecución (no nulos).
- **(1.5 puntos)** Por seguir las transiciones y actualizar la tabla correctamente.
- **(1 punto)** Por retornar correctamente el output.
  - **(0.5 punto)** Por retornar una palabra escrita por el transductor.
  - **(0.5 punto)** Por retornar *null* (o similar) cuando no hay output.

Pregunta 3

Por demostrar que si una gramática libre de contexto no tiene loops, entonces define un lenguaje regular. Sea $\mathcal{G} = (V, \Sigma, P, S)$ una CFG sin loops y $G_{\mathcal{G}} = (V, E)$ un grafo tal que $|V| = n$ y $(X, Y) \in E$ si, y solo si, $X \to \alpha Y \beta \in P$.

Por demostrar que si $\mathcal{G}$ no tiene loops, entonces $G_{\mathcal{G}}$ tampoco. Por contrapositivo, si $G_{\mathcal{G}}$ tiene loops, entonces existe un camino $X = X_1 \to X_1 \to X_2 \to \cdots \to X_m = X$. Luego, existen las producciones $X_1 \to \alpha_2 X_2 \beta_2$, $X_2 \to \alpha_3 X_3 \beta_3$, …, $X_{m-1} \to \alpha_m X_m \beta_m$. Componiéndolas se obtiene $X \overset{+}{\Rightarrow} \alpha X \beta$.

Como $G_{\mathcal{G}}$ no tiene loops, entonces define un grafo dirigido acíclico (DAG). Así, se puede definir la secuencia $X_1, X_2, \ldots, X_n$ de las variables en V ordenadas topológicamente tal que, con $i < j \leq n$, desde X_i no se puede llegar a X_j (al revés de la definición convencional).

Sea $\mathcal{G}_X = (V, \Sigma, P, X)$. Por demostrar, utilizando inducción fuerte, que $\forall i \in \{1, n\}.\mathcal{L}(\mathcal{G}_{X_i})$ es finito.

- Caso base: Desde X_1 no se puede llegar a ninguna variable, es decir, todas las son de la forma $X \to w$ con $w \in \Sigma^*$. Como P es finito entonces, $\mathcal{L}(\mathcal{G}_{X_1})$ también lo es.
- Caso inductivo: Sea X_i una variable en la secuencia ordenada. Por hipótesis de inducción $\forall k < i.\mathcal{L}(\mathcal{G}_{X_k})$ es finito. Gracias al orden topológico y que $\mathcal{G}$ no tiene loops, entonces las producciones de X_i son de la forma $X_i \to \alpha$ con $\alpha \in (\Sigma \cup \{X_1, \ldots, X_{i-1}\})^*$. Luego, considerando las producciones de X_i de la forma $X_i \to \alpha_1 | \alpha_2 | \ldots | \alpha_l$, si se reemplaza cada aparición de X_k con $k < i$ en cada $\alpha_1, \alpha_2, \ldots, \alpha_l$ por toda posible palabra en $\mathcal{L}(\mathcal{G}_{X_k})$ se logra definir $\mathcal{L}(\mathcal{G}_{X_i})$ a través de producciones de la forma $X_i \to w$ con $w \in \Sigma^*$. Como los $\mathcal{L}(\mathcal{G}_{X_k})$ son finitos y las producciones originales de X_i son finitas, entonces las producciones con reemplazo son finitas. Por lo tanto, $\mathcal{L}(\mathcal{G}_{X_i})$ es finito.

Finalmente, $\mathcal{L}(\mathcal{G}_{X_S}) = \mathcal{L}(\mathcal{G})$ es finito y todo lenguaje finito es regular.

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

- **(1 punto)** Por plantear la demostración de que el grafo de una CFG sin ciclos tampoco los tiene.
- **(1 punto)** Por la demostración de que el grafo de una CFG sin ciclos tampoco los tiene.

- **(1 punto)** Por ordenar las variables para facilitar la inducción.
- **(1 punto)** Por el caso base.
- **(1 punto)** Por el caso inductivo.
- **(1 punto)** Por la conclusión y mencionar que todo lenguaje finito es regular.