Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC2223 – Teoría de Autómatas y Lenguajes Formales
Segundo semestre de 2024
Profesor: Cristian Riveros
Ayudante: Amaranta Salas

Ayudantia 15

Repaso Exámen

Problema 1

Sea $\Sigma = \{a, b\}$. Para lenguajes L_1, $L_2 \subseteq \Sigma^*$ se define:

$$L_1|L_2 = \{uv \in L_1 \mid v \in L_2\}$$

Demuestre que si L_1 y L_2 son lenguajes regulares, entonces $L_1|L_2$ también es regular.

Solución

Como L_1 y L_2 son lenguajes regulares, sabemos que existirán los DFA $\mathcal{A}_1 = (Q_1, \Sigma, \delta_1, q_{01}, F_1)$ y $\mathcal{A}_2 = (Q_2, \Sigma, \delta_2, q_{02}, F_2)$ tales que $\mathcal{L}(\mathcal{A}_1) = L_1$ y $\mathcal{L}(\mathcal{A}_2) = L_2$. Luego, para demostrar que $L_1|L_2$ es regular, bastará con encontrar un autómata que lo defina.
Sea $\mathcal{A} = (Q_1 \cup Q_1 \times Q_2, \Sigma, \Delta, \{q_{01}\}, F_1 \times F_2)$, con Δ dado por:

$$\begin{aligned}\Delta \ = \ & \{(p_1, a, q_1) \mid \delta(p_1, a) = q_1\} \cup \\ & \{(p_1, \varepsilon, (p_1, q_{02})) \mid p_1 \in Q_1\} \cup \\ & \{((p_1, p_2), a, (q_1, q_2)) \mid \delta(p_1, a) = q_1 \ \wedge \ \delta(p_2, a) = q_2\}\end{aligned}$$

Ahora debemos demostrar que $\mathcal{L}(\mathcal{A}) = L_1|L_2$.

1. $L_1|L_2 \subseteq \mathcal{L}(\mathcal{A})$:

   Sea $w = uv \in L_1|L_2$. Sabemos entonces que $w \in L_1 \ \wedge \ v \in L_2$, con $w = a_1...a_n$ y $v = a_i...a_n$. Luego, sabemos que existirán ejecuciones:

   $$\rho_1 : \ q_{01} \xrightarrow{a_1} p_1 \xrightarrow{a_2} ... \xrightarrow{a_n} p_n \ \wedge \ p_n \in F_1$$

   $$\rho_2 : \ q_{02} \xrightarrow{a_i} q_i \xrightarrow{a_{i+1}} ... \xrightarrow{a_n} q_n \ \wedge \ q_n \in F_2$$

   A partir de ρ_1 y ρ_2 podemos construir:

   $$\rho : \ q_{01} \xrightarrow{a_1} p_1 \xrightarrow{a_2} ... \xrightarrow{a_{i-1}} p_{i-1} \xrightarrow{\varepsilon} (p_{i-1}, q_{02}) \xrightarrow{a_i} (p_i, q_i) \xrightarrow{a_{i+1}} ... \xrightarrow{a_n} (p_n, q_n) \ \wedge \ p_n \in F_1 \ \wedge \ q_n \in F_w$$

   Es claro que ρ es una ejecución de aceptación de $\mathcal{A}$ sobre w, es decir, $w \in \mathcal{L}(\mathcal{A})$ y por lo tanto $L_1|L_2 \subseteq \mathcal{L}(\mathcal{A})$.

2. $\mathcal{L}(\mathcal{A}) \subseteq L_1|L_2$:

   Sea $w \in \mathcal{L}(\mathcal{A})$. Por la construcción de $\mathcal{A}$, sabemos que existe una ejecución de $\mathcal{A}$ sobre w que tendrá la siguiente forma:

   $$\rho : \ q_{01} \xrightarrow{a_1} p_1 \xrightarrow{a_2} ... \xrightarrow{a_{i-1}} p_{i-1} \xrightarrow{\varepsilon} (p_{i-1}, q_{02}) \xrightarrow{a_i} (p_i, q_i) \xrightarrow{a_{i+1}} ... \xrightarrow{a_n} (p_n, q_n) \ \wedge \ p_n \in F_1 \ \wedge \ q_n \in F_2$$

Luego, por definición de Δ, sabemos que para toda transición de la forma $p_{k-1} \xrightarrow{a_k} p_k$ se cumplirá $\delta_1(p_{k-1}, a_k) = p_k$ y para toda transición de la forma $(p_{k-1}, q_{k-1}) \xrightarrow{a_k} (p_k, q_k)$ se cumplirá $\delta_1(p_{k-1}, a_k) = p_k$ y $\delta_2(q_{k-1}, a_k) = q_k$. Utilizando esto, podemos generar las siguientes ejecuciones de $\mathcal{A}_1$ y $\mathcal{A}_2$

$$\rho_1 : q_{01} \xrightarrow{a_1} p_1 \xrightarrow{a_2} ... \xrightarrow{a_n} p_n \ \wedge \ p_n \in F_1$$
$$\rho_2 : q_{02} \xrightarrow{a_i} q_i \xrightarrow{a_{i+1}} ... \xrightarrow{a_n} q_n \ \wedge \ q_n \in F_2$$

Sabemos entonces que $w = a_1...a_n \in L_1$ y que $v = a_i...a_n \in L_2$, lo que a su vez significa que $\exists u.\ uv \in L_1 \wedge v \in L_2$ y, por tanto $w \in L_1|L_2$.

Por lo tanto $\mathcal{L}(\mathcal{A}) = L_1|L_2$ y $L_1|L_2$ es un lenguaje regular.

Problema 2

1. Demuestre que el siguiente lenguaje es libre de contexto:

   $$R = \{a^i b^j c^k \mid i < j \vee j < k\}$$

   Para esto, demuestre una gramática o un PDA que defina el lenguaje y explique su correctitud.

2. Demuestre que el siguiente lenguaje NO es libre de contexto:

   $$S = \{a^i b^j c^k \mid i < j \wedge j < k\}$$

Solución

1. Sea $G = (\{S, X, C, X', B, Y, A, Y'\}, \{a, b, c\}, P, S)$ una Gramática Libre de Contexto, donde P contiene las siguientes producciones:

   $$\begin{array}{rcl} S & \longrightarrow & X \mid Y \\ \\ X & \longrightarrow & X'C \\ C & \longrightarrow & cC \mid \epsilon \\ X' & \longrightarrow & aX'b \mid bB \\ B & \longrightarrow & bB \mid \epsilon \\ \\ Y & \longrightarrow & AY' \\ A & \longrightarrow & aA \mid \epsilon \\ Y' & \longrightarrow & bY'c \mid cC \end{array}$$

   Se escoge cual de las 2 condiciones cumplir. Al elegir una, se agrega una cantidad arbitraria de la letra que no influye en la condición. Después, se agrega una cantidad igual de letras tal que $i = j$ o $j = k$ dependiendo del caso. Finalmente, se agrega al menos una letra tal que $j > i$ o $k > j$ según el caso.

2. Por Lema de Bombeo para CFG. Sea para todo $N > 0$ la palabra $z = a^N b^{N+1} c^{N+2}$. Sean $i = N$, $j = N + 1$ y $k = N + 2$ se tiene que $i < j < k$, esto es, z está en S y $|z| \geq N$.

   Sea $z = uvwxy$ una descomposición cualquiera tal que $vx \neq \epsilon$ y $|vwx| \leq N$. Sea $z' = uv^i wx^i y$ para algún $i \geq 0$. Luego, se tienen los siguientes casos (no necesariamente excluyentes entre sí):

   (a) Si v o x son combinaciones de 2 letras. Entonces, con $i = 2$, z' ya no está en $\mathcal{L}(a^*b^*c^*)$ ni en S.
   (b) Si $x \in \mathcal{L}(a^*)$. Entonces, con $i = 2$, se tienen mayor o igual letras a que b y $z' \notin S$.
   (c) Si $x \in \mathcal{L}(b^*)$. Entonces, con $i = 2$, se tienen mayor o igual letras b que c y $z' \notin S$.
   (d) Si $v \in \mathcal{L}(b^*)$. Entonces, con $i = 0$, se tienen mayor o igual letras a que b y $z' \notin S$.
   (e) Si $v \in \mathcal{L}(c^*)$. Entonces, con $i = 0$, se tienen mayor o igual letras b que c y $z' \notin S$.

   Por lo tanto, S no es un Lenguaje Libre de Contexto.

Problema 3

Una gramática $\mathcal{G}$ esta en forma normal de Greibach (GNF) si todas sus reglas son de la forma:

$$X \to aY_1 \dots Y_k$$

para algún $k \geq 0$.
Sea $\mathcal{G} = (V, \Sigma, P, S)$ una gramática en forma normal de Greibach sin variables inútiles. Demuestre que $\mathcal{G}$ es una gramática $LL(1)$ si, y solo si, para todo par de reglas $X \to a\gamma$ y $X \to a'\gamma'$ en P se tiene que, si $a = a'$, entonces $\gamma = \gamma'$.

Solución

Sea G una gramática en forma normal de Greibach sin variables inútiles, con reglas del tipo $X \to a\gamma$ y con $\gamma \in V^*$. Recordamos la definición de $LL(1)$, que es la siguiente: G es $LL(1)$ si para todo par de derivaciones

$$S \Rightarrow^* uX\beta \Rightarrow u\gamma_1\beta \Rightarrow^* uv_1$$
$$S \Rightarrow^* uX\beta \Rightarrow u\gamma_2\beta \Rightarrow^* uv_2$$

Si $v_1|_1 = v_2|_1$, entonces $\gamma_1 = \gamma_2$.

($\Rightarrow$) Supongamos que G es $LL(1)$. Sea $X \to a\gamma, X \to a'\gamma' \in P$, tal que $a = a'$. Por demostrar: $\gamma = \gamma'$. Como G no tiene variables inútiles, entonces

$$S \Rightarrow^* uX\beta \text{ para algún } u \text{ y } \beta$$
$$S \Rightarrow^* uX\beta \Rightarrow ua\gamma\beta$$
$$S \Rightarrow^* uX\beta \Rightarrow ua'\gamma'\beta$$

Como G no tiene variables inútiles:

$$a\gamma\beta \Rightarrow^* v_1$$
$$a'\gamma'\beta \Rightarrow^* v_2$$

Si $a = a'$, entonces $v_1|_1 = v_2|_1$. Por lo tanto:

$$S \Rightarrow^* uX\beta \Rightarrow ua\gamma\beta \Rightarrow uv_1$$
$$S \Rightarrow^* uX\beta \Rightarrow ua'\gamma'\beta \Rightarrow uv_2$$
$$v_1|_1 = v_2|_1$$

Como la gramática es $LL(1)$, entonces $a\gamma = a'\gamma'$ por lo que $\gamma = \gamma'$.

($\Leftarrow$) Suponemos que para dos reglas $x \to a\gamma$ y $X \to a'\gamma'$, si $a = a'$ entonces $\gamma = \gamma'$. Suponga que

$$S \Rightarrow^* uX\beta \Rightarrow u\gamma_1\beta \Rightarrow^* uv_1$$
$$S \Rightarrow^* uX\beta \Rightarrow u\gamma_2\beta \Rightarrow^* uv_2$$
$$v_1|_1 = v_2|_1$$

Demostraremos que $\gamma_1 = \gamma_2$. Como G está en GNF, $\gamma_1 = a\gamma$ y $\gamma_2 = a\gamma'$. Por derivación por la izquierda, sabemos que $a = v_1|_1 = v_2|_1 = a'$. Luego, tenemos que:

$$X \to a\gamma, X \to a'\gamma' \text{ y } a = a' \Rightarrow \gamma = \gamma'$$

Por lo tanto $\gamma_1 = \gamma_2$ y G es LL(1).

Problema 4

Para un lenguaje $L \subseteq \Sigma^*$ y $a \in \Sigma$, se define $\texttt{follow}_k(a) = \{v|_k \mid u \cdot a \cdot v \in L\}$.
Escriba un algoritmo que reciba como entrada un autómata finito no-determinista $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$, una letra $a \in \Sigma$, una palabra $w \in \Sigma^*$ y $k > 0$, y responda `TRUE` si, y solo si, $w \in \texttt{follow}_k(a)$. Su algoritmo debe tomar tiempo $\mathcal{O}(|\mathcal{A}| \cdot |w|)$. Por último, explique la correctitud de su algoritmo.

Solución

Una posible solución para esta pregunta consiste en la siguiente modificación del algoritmo de evaluación de autómatas no deterministas `eval-NFAonthefly`. Notar que aprovechamos el hecho de que su tiempo está en $\mathcal{O}(|\mathcal{A}| \cdot |w|)$.

`eval-follow`$_k$ $(\mathcal{A} = (Q, \Sigma, \Delta, I, F),\ a,\ w,\ k)$

---

if $|w| > k$ **then**
  return FALSE
end if
$\mathcal{A} =$ `remover_inútiles`$(\mathcal{A})$ ▷ Búsqueda sobre los estados $\mathcal{O}(|\mathcal{A}|)$
$S = \emptyset$
for $(p, t, q) \in \Delta$ **do**
  if $t = a$ **then**
    $S = S \cup \{q\}$
  end if
end for
$S_{final} =$ `eval-NFAonthefly`$(\mathcal{A} = (Q, \Sigma, \Delta, S, F), w)$
if $S_{final} \cap F \neq \emptyset$ **then**
  return TRUE
else if $S_{final} \cap F = \emptyset \wedge |w| = k$ **then**
  return `alcanzables`$(S_{final}) \cap F \neq \emptyset$ ▷ Búsqueda desde S_{final} $\mathcal{O}(|\mathcal{A}|)$
end if
return FALSE

Donde `remover_inútiles` toma un autómata y nos retorna uno nuevo que tiene solo los estados que son alcanzables. Por otro lado `alcanzables`, dado un conjunto de estados, nos entrega todos los estados alcanzables desde estos usando Δ. Para ambos casos se pueden usar algoritmos de búsqueda lineales sobre la cantidad de estados.