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

IIC2223 — Teoría de Autómatas y Lenguajes Formales — 2' 2020

PAUTA EXAMEN

Pregunta 1

Una posible solución es la siguiente. Sea $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$ el autómata finito no-determinista. A partir de $\mathcal{A}$, construimos $\mathcal{A}' = (Q', \Sigma \cup Q, \Delta', I', F')$ tal que:

- $Q' = \{q_0\} \cup Q \cup Q \times Q$ donde q_0 es un estado nuevo (esto es $q_0 \notin Q$)
- $\Delta' = \{(q_0, p, p) \mid p \in I\} \;\cup\; \{(p, q, (p, q)) \mid q \in F\} \;\cup\; \{((p, q), a, (p', q)) \mid (p, a, p') \in \Delta\}$
- $I' = \{q_0\}$
- $F' = \{(q, q) \mid q \in F\}$

Por último, demostramos que $\mathcal{S}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$, una dirección a la vez.

1. ($\mathcal{S}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$) Sea $w \in \mathcal{S}(\mathcal{A})$. Por la definición de $\mathcal{S}(\mathcal{A})$, sabemos que w tiene la forma $p \cdot q \cdot a_1 ... a_n$, con $p \in I$, $q \in F$, $a_i \in \Sigma$ y existe una ejecución de aceptación $\rho : p = p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} ... \xrightarrow{a_n} p_n = q$ de $\mathcal{A}$ sobre $a_1 a_2 ... a_n$. Por la construcción de $\mathcal{A}'$ podemos verificar que la siguiente es una ejecución de $\mathcal{A}'$ sobre w:
$$q_0 \xrightarrow{p} p \xrightarrow{q} (p, q) \xrightarrow{a_1} (p_1, q) ... \xrightarrow{a_n} (p_n, q)$$
Luego, dado que $p_n = q$, w llega a un estado final, entonces w es aceptado por $\mathcal{A}'$ y $w \in \mathcal{L}(\mathcal{A}')$).

2. ($\mathcal{L}(\mathcal{A}') \subseteq \mathcal{S}(\mathcal{A})$) Sea $w' \in \mathcal{L}(\mathcal{A}')$. Por construcción de $\mathcal{A}'$, podemos ver que w' tiene que tener la forma $w' = p \cdot q \cdot a_1 ... a_n$ y existe una ejecución de aceptación
$$\rho' : q_0 \xrightarrow{p} p \xrightarrow{q} (p, q) \xrightarrow{a_1} (p_1, q) ... \xrightarrow{a_n} (q, q)$$
de $\mathcal{A}'$ sobre w'. Por la definición de $\mathcal{A}'$, podemos obtener la ejecución $\rho : p = p_o \xrightarrow{a_1} p_1 \xrightarrow{a_2} ... \xrightarrow{a_n} p_n = q$ a partir de las transiciones del estilo $((p, q), a, (p', q))$ de Δ'. Dado que $p \in I$, $q \in F$ y $(p_i, a_i, p_{i+1}) \in \Delta$, entonces ρ es una ejecución de aceptación sobre $a1 ... a_n$. Esto es $w' \in \mathcal{S}(\mathcal{A})$.

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

- **(0.5 puntos)** Por definir correctmente el alfabeto del autómata $\mathcal{A}'$.
- **(0.5 puntos)** Por definir correctmente el conjunto de estados finales del autómata $\mathcal{A}'$.
- **(1 punto)** Si el autómata lea el estado p y lo almacena.
- **(1 punto)** Si el autómata lea el estado q y lo almacena.
- **(1 punto)** Si el autómata simule las transiciones del autómata $\mathcal{A}$.
- **(1 punto)** Por demostrar correctmente que $\mathcal{S}(\mathcal{A}) \subseteq \mathcal{L}(\mathcal{A}')$.
- **(1 puntos)** Por demostrar correctmente que $\mathcal{L}(\mathcal{A}') \subseteq \mathcal{S}(\mathcal{A})$.

Pregunta 2

Pregunta 2.1

Una posible solución consiste en aplicar el contrapositivo del lema de bombeo para lenguajes regulares de la siguiente manera. Para $N > 0$ cualquiera, podemos escoger la siguiente palabra

$$a^N \# a^{N+1} \# a^{N+2} \# \cdots \# a^{2N}$$

la cuál subdividimos como $x = \varepsilon$, $y = a^N$, $z = \# a^{N+1} \# a^{N+2} \# \cdots \# a^{2N}$, se cumple claramente que $|xyz| \leq N$. Ahora, nuestra competencia, el demonio, elige una descomposición cualquiera de $y = \underbrace{a^j}_{u}\underbrace{a^k}_{v}\underbrace{a^l}_{w}$,en donde se tiene que $k \neq 0$ y además $j + k + l = N$.

Ahora escogemos $i = 2$ para bombear y se tiene una palabra

$$xuv^2wz \ = \ a^j a^{2k} a^l \# a^{N+1} \# a^{N+2} \# \cdots \# a^{2N}$$

con $N + 1 \leq j + 2k + l \leq 2N$, pues $k \neq 0$. Luego existen 2 subpalabras $u_1 = a^j a^{2k} a^l$ y u_r que coinciden, de forma tal que la palabra xuv^2wz no pertenece al lenguaje L, pues no son todos los fragmentos de forma a^m son distintos de a pares. De acá se concluye que el lenguaje no es regular.

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

- **(1,5 puntos)** Por elegir correctamente una palabra xyz de largo mayor a N.
- **(0,5 puntos)** Por elegir correctamente una subdivisión de y.
- **(0,5 puntos)** Por elegir correctamente la cantidad de veces i a bombear.
- **(0,5 puntos)** Por argumentar que la palabra obtenida no pertenece al lenguaje.

Pregunta 2.2

La idea general es modificar la demostración de la versión clásica del lema. Si tenemos una palabra ω de largo mayor a un N suficientemente grande, podemos repetir un subarbol dentro del árbol de derivación. Dado el lenguaje libre de contexto L del enunciado, podemos suponer que existe una grámatica en forma normal de Greibach que lo genera, es decir todas sus producciones son de forma:

$$A \to aA_1 \cdots A_n$$

Donde a es una terminal y lo que sigue a la derecha es una secuencia de variables no terminales.

Ahora la demostración sigue la misma estrategia vista en clases. Elegimos $N = 2^{|V|+1}$ y, como cada producción tiene la forma de arriba, todo árbol de derivación de esta gramática tiene una rama con al menos $\log_2(N)$ nodos internos. Ahora si tomamos una palabra ω de largo mayor a N, por lo anterior su árbol de derivación tiene una rama de altura mayor a $|V|$. Luego por el principio del palomar, en esa rama existe una variable al menos que se repite, llamemosla X, por ende podemos bombear, luego la palabra uv^iwx^iy pertenece al lenguaje L. Como las reglas son de la forma $A \to aA_1 \cdots A_n$ entonces el subarbol desde X a X tendrá al menos una letra a la derecha, lo que implica que $v \neq \epsilon$.

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

.

- **(1 punto)** Por usar la forma normal de Greibach.
- **(0,5 puntos)** Por usar la forma normal para argumentar que $v \neq \epsilon$.
- **(0,5 puntos)** Por elegir correctamente el valor de N.
- **(0,5 puntos)** Por usar el principio del palomar.
- **(0,5 puntos)** Por explicar como bombear.

Pregunta 3

Demostraremos que para todo lenguaje regular L, existe una gramática libre de contexto $\mathcal{G}$ tal que $L = \mathcal{L}(\mathcal{G})$ y $\mathcal{G}$ es LR(k) para algún k. Dado que el lenguaje L es regular, sabemos que existirá un DFA $A = (Q, \Sigma, \delta, q_0, F)$ tal que $L = \mathcal{L}(A)$; por lo que podemos construir la gramática $\mathcal{G} = (Q, \Sigma, P, \{q_0\})$ a partir de A con:

$$P = \{p \to aq \mid p, q \in Q,\, a \in \Sigma \,\wedge\, \delta(p, a) = q\} \;\;\cup\;\; \{p \to \varepsilon \,|\, p \in F\}$$

Como se vio en clases, esta gramática definirá el mismo lenguaje que el autómata en el que se basa, es decir $L = \mathcal{L}(A) = \mathcal{L}(\mathcal{G})$, por lo que ahora debemos demostrar que existe un k tal que $\mathcal{G}$ es LR(k).
Una gramática libre de contexto será LR(k), si para toda derivación:

- $\alpha \cdot \beta \cdot v_1 \underset{rm}{\Longleftarrow} \alpha \cdot X \cdot v_1 \overset{*}{\underset{rm}{\Longleftarrow}} S'$
- $\alpha \cdot \beta \cdot v_2 \underset{rm}{\Longleftarrow} \alpha' \cdot Y \cdot v_2' \overset{*}{\underset{rm}{\Longleftarrow}} S'$
- $v_1|_k = v_2|_k$

se cumple que $\alpha = \alpha'$, $X = Y$ y $v_2 = v_1$. Observando ahora las producciones de la gramática, podemos ver que tendremos solamente dos opciones para sus derivaciones, y son las siguientes:

- $S' \overset{*}{\underset{rm}{\Longrightarrow}} a_1a_2...a_k \cdot p \cdot \varepsilon \underset{rm}{\Longrightarrow} a_1a_2...a_k \cdot aq \cdot \varepsilon$, dónde $\alpha = a_1a_2...a_k$, $X = p$, $v_1 = \varepsilon$ y $\beta = aq$. Luego, tomando una derivación diferete tenemos: $S' \overset{*}{\underset{rm}{\Longrightarrow}} a_1a_2...a_j \cdot p' \cdot \varepsilon \underset{rm}{\Longrightarrow} a_1a_2...a_k \cdot aq \cdot \varepsilon$, dónde $\alpha' = a_1a_2...a_j$, $Y = p'$, $v_2 = \varepsilon$, $v_2' = \varepsilon$ y α, β deben ser los mismos que en la derivación anterior. Debido a la construcción de la gramática, cada producción puede generar una sola letra, lo que significa $\alpha = \alpha'$. Además, como está basada en un DFA, solamente existe un estado p que leyendo la letra a llegue al estado q, lo que sigifica que $p = p$, esto es, $X = Y$. Finalmente, tenemos que $v_1 = v_2 = v_2' = \varepsilon$, lo que significa no solo que la gramática será LR(k) para algún k, si no también que podría serlo para todo $k \geq 0$, pues $\varepsilon|_k = \varepsilon$ para todo $k \geq 0$.

- $S' \overset{*}{\underset{rm}{\Longrightarrow}} a_1a_2...a_k \cdot p \cdot \varepsilon \underset{rm}{\Longrightarrow} a_1a_2...a_k \cdot \varepsilon \cdot \varepsilon$, dónde $\alpha = a_1a_2...a_k$, $X = p$, $v_1 = \varepsilon$ y $\beta = \varepsilon$.

  A primera vista, pareciera que este caso es análogo al anterior, pues podemos tomar una derivación de la forma: $S' \overset{*}{\underset{rm}{\Longrightarrow}} a_1a_2...a_j \cdot p' \cdot \varepsilon \underset{rm}{\Longrightarrow} a_1a_2...a_k \cdot \varepsilon \cdot \varepsilon$, dónde $\alpha' = a_1a_2...a_j$, $Y = p'$, $v_2 = \varepsilon$ y α, β deben ser los mismos que en la derivación anterior; y luego podemos utilizar el mismo argumeto para $\alpha = \alpha'$, $v_1 = v_2 = v_2'$ y $p = p'$. Sin embargo existe otra derivación posible. Consideremos la siguiente derivación:

  $$S' \overset{*}{\underset{rm}{\Longrightarrow}} a_1a_2...a_kb_1b_2...b_n \cdot p'' \cdot \varepsilon \underset{rm}{\Longrightarrow} a_1a_2...a_k \cdot \varepsilon \cdot b_1b_2...b_n$$

  dónde $\alpha' = a_1a_2...a_kb_1b_2...b_n$, $Y = p''$, $v_2' = \varepsilon$, $v_2 = b_2...b_n$ y α, β deben ser los mismos que en la derivación anterior.

  Pareciera que este caso no aporta a la demostración, pues es claro que $v_1 \neq v_2$ y $v_2 \neq v_2'$, sin embargo sí se cumple que $v_1|_0 = v_2|_0$ y, puesto que $v_2 \neq v_2'$, tendremos que esta gramática no es LR(0). Ahora bien, como si tenemos que $v_1|_1 \neq v_2|_1$, esta propiedad si se cumple para LR(1).

Por lo tanto, la gramática $\mathcal{G}$ será LR(1).

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

- **(2 puntos)** Por construir correctamente la gramática.
- **(3 puntos)** Por demostrar que la gramática es LR(k) para algún k, usando las producciones con $\beta \neq \varepsilon$.
  - **(1 puntos)** Por utilizar correctamente $\mathcal{G}$ para construir una derivación de la forma $S' \overset{*}{\underset{rm}{\Longrightarrow}} a_1a_2...a_k \cdot p \cdot \varepsilon$.
  - **(2 puntos)** Por demostrar que las demás derivaciones cumplen con $\alpha = \alpha'$, $X = Y$, $v_2 = v_2'$.
- **(1 punto)** Por considerar el caso $\beta = \varepsilon$, incluso si no se descarta que la gramática sea LR(0).

Pregunta 4

Para un autómata finito no-determinista $\mathcal{A} = (Q, \Sigma, \Delta, I, F)$, considere el conjunto:

$$F^* = \{q \mid \text{existe una ejecución de aceptación } \rho \text{ desde } q \text{ sobre } A\}$$

es decir todos los estados tal que desde ellos se puede llegar a algún estado final con alguna ejecución. Luego el algoritmo propuesto es el siguiente.

Algorithm 1: Determina si $w \in \texttt{first}_k(\mathcal{L}(\mathcal{A}))$

```
Input  : A = (Q, Σ, Δ, I, F), w = a_1 ... a_n, k
Output: true si, y solo si, w esta en first_k(L(A))
F* := DFS(A);
if |w| > k then
 | return false
end
S := I, i := 1;
while i ≤ |w| & i ≤ k do
 | S := next(Δ, S, a_i);
 | i := i + 1;
end
return S ∩ F* ≠ ∅
```

Donde $\texttt{DFS}(A)$ es una búsqueda en profundidad de a partir cada estado del autómata hasta encontrar un estado final lo que retornará solo los estados para los cuales se pueda alcanzar un estado final desde ellos. Además `next` se define como se vió en clases para el algoritmo de evaluación de un NFA.

Finalmente, se debe justificar la correctitud del algoritmo y del tiempo del algoritmo que es:

$$\mathcal{O}((|\Delta| + |Q|) \cdot min(k, |w|) + |\Delta| \cdot |Q|).$$

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

- **(1 punto)** Por utilizar la evaluación `on_the_fly` para el automata.
- **(1 punto)** Por utilizar k para no revisar la palabra completa.
- **(2 puntos)** Por utilizar F^* o bien la noción de estados que alcanzan estados finales.
- **(1 punto)** Por explicar la correctitud del algoritmo.
- **(1 punto)** Por justificar el que su algoritmo cumple con el tiempo pedido.