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

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

PAUTA EXAMEN

Pregunta 1

Se busca demostrar que para todo autómata apilador $\mathcal{P}$, existe un autómata apilador alternativo $\mathcal{D}$ tal que $\mathcal{L}(\mathcal{P}) = \mathcal{L}(\mathcal{D})$.

Sea el autómata apilador $\mathcal{P} = (Q, \Sigma, \Gamma, \Delta, q_0, \perp, F)$. Construimos un PDA alternativo $\mathcal{D} = (Q', \Sigma, \Delta', q_0', F')$ tal que:

- $Q' = Q \cup \Gamma \cup \{q_0'\}$
- $F' = F$
- $\Delta' = \{(q_0', \epsilon, q_0\perp)\} \cup \{(pA, a, q\gamma) | (p, a, A, q, \gamma) \in \Delta\}$

Ahora, debemos demostrar la correctitud del PDA alternativo generado:

- ($\Rightarrow$) Sea una palabra w y la ejecución de aceptación $(q_0\perp, w) \overset{*}{\vdash}_{\mathcal{P}} (q_f, \epsilon)$ del autómata apilador $\mathcal{P}$ sobre w. Por la construcción del autómata apilador alternativo $\mathcal{D}$, tendremos la siguiente ejecución:

  $$(q_0', w) \vdash_{\mathcal{D}} (q_0\perp, w) \overset{*}{\vdash}_{\mathcal{D}} (q_f, \epsilon)$$

  Lo que es una ejecución de aceptación para $\mathcal{D}$.

- ($\Leftarrow$) Sea una palabra w y la ejecución de aceptación $(q_0', w) \overset{*}{\vdash}_{\mathcal{D}} (q_f, \epsilon)$ del autómata apilador alternativo $\mathcal{D}$. Por construcción, tenemos que:

  $$(q_0', w) \vdash_{\mathcal{D}} (q_0\perp, w) \overset{*}{\vdash}_{\mathcal{D}} (q_f, \epsilon)$$

  Lo que implica la existencia de la ejecución del autómata $\mathcal{P}$:

  $$(q_0\perp, w) \overset{*}{\vdash}_{\mathcal{P}} (q_f, \epsilon)$$

  La cual también es una ejecución de aceptación.

Finalmente, al haber demostrado la correctitud de lo generado, damos por concluida la demostración.
Dado lo anterior, la distribución de puntaje es la siguiente:

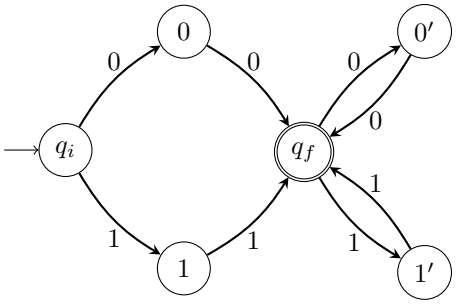
- **(4 puntos)** Por definir correctamente $\mathcal{D}$:
  - (1 punto) por definir correctamente $Q' = Q \cup \Gamma \cup \{q_0'\}$
  - (1 punto) por definir correctamente $F' = F$
  - (2 puntos) por definir correctamente $\Delta' = \{(q_0', \epsilon, q_0\perp)\} \cup \{(pA, a, q\gamma) | (p, a, A, q, \gamma) \in \Delta\}$
- **(1 punto)** Por demostrar correctitud en la dirección ($\Rightarrow$).
- **(1 punto)** Por demostrar correctitud en la dirección ($\Leftarrow$).

Pregunta 2

Se puede notar que el lenguaje L_1 es regular, y que el lenguaje L_2 no lo es. A continuación, se demuestran ambas afirmaciones por separado.

Pregunta 2.1

Una posible demostración de que L_1 es regular, es mediante el siguiente autómata:



La idea general para aceptar el lenguaje $L_1 = \{a_1 \dots a_{2n} \in \{0,1\}^* \mid n > 0 \wedge a_1 a_3 \dots a_{2n-1} = a_2 a_4 \dots a_{2n}\}$, es que cada vez que se lee una letra en una posición impar, se debe revisar que la letra inmediatamente siguiente sea la misma. Luego, el autómata siempre lee de a dos letras, esperando que sean iguales. Si lo son, vuelve al estado final.

Además, por la condición de que $n > 0$, el ϵ no debe ser aceptado, por lo que el estado inicial no puede ser final, y se añaden estados para revisar las primeras dos letras de la palabra.

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

- **(1 puntos)** Por explicación de autómata
- **(4 puntos)** Por autómata correcto
- **(1 puntos)** Por manejar caso de input $w = \epsilon$

Pregunta 2.2

Una posible demostración de que L_2 no es regular, es mediante la siguiente aplicación del contrapositivo del lema de bombeo. Sea un $N > 0$ arbitrario. Existe la siguiente palabra:

$$0^N 1 1 0^N \in L_2$$

La cual se puede dividir de la siguiente forma:

$$\begin{aligned} x &= \epsilon \\ y &= 0^N \\ z &= 1 1 0^N \end{aligned}$$

Entonces, para cualquier $y = 0^j 0^k 0^l$ con $u = 0^j$, $v = 0^k$, $w = 0^l$ y $k > 0$, podemos tomar $i = 2$ tal que:

$$x \cdot u \cdot v^2 \cdot w \cdot z = 0^{N+k} 1 1 0^N \notin L_2$$

Por lo tanto, queda demostrado por el contrapositivo del lema de bombeo que L_2 no es regular.

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

- **(2 puntos)** Por elección de palabra perteneciente a L_2
- **(2 puntos)** Por correcta elección de x, y, z y posibles casos de u, v, w
- **(2 puntos)** Por bombear y llegar a palabra que no pertenece a L_2

Pregunta 3

Pregunta 3.1

Se deben demostrar ambas direcciones. En primer lugar demostraremos la dirección *existe* $X \in V$ *y* $\alpha, \beta \in (V \cup \Sigma)^*$ *tal que* $X \overset{+}{\Rightarrow} \alpha X \beta \Rightarrow \mathcal{L}(\mathcal{G})$ *es infinito*. Sabemos que $X \overset{+}{\Rightarrow} \alpha X \beta$. Como $\mathcal{G}$ está en CNF, entonces necesariamente $\alpha \neq \varepsilon$ o $\beta \neq \varepsilon$ (no pueden ser vacios a la vez). También, como la gramática no tiene variables inútiles, sabemos que:

- $\exists \alpha', \beta'$ $S \overset{*}{\Rightarrow} \alpha' X \beta'$ (alcanzable)
- $\alpha \overset{*}{\Rightarrow} u$, $\beta \overset{*}{\Rightarrow} v$, $\alpha' \overset{*}{\Rightarrow} u'$, $\beta' \overset{*}{\Rightarrow} v'$, $X \overset{*}{\Rightarrow} w$ (generadoras)

Y como $\mathcal{G}$ está en CNF, $u \neq \varepsilon$ o $v \neq \varepsilon$. Luego, podemos realizar la construcción:

$$S \overset{*}{\Rightarrow} \alpha' X \beta' \overset{*}{\Rightarrow} \alpha' \alpha^i X \beta^i \beta' \overset{*}{\Rightarrow} u' u^i w v^i v'$$

Como $u \neq \varepsilon$ o $v \neq \varepsilon$, entonces el subconjunto generado con esta construcción es infinito.

Ahora demostraremos la otra dirección, es decir $\mathcal{L}(\mathcal{G})$ *es infinito* $\Rightarrow$ *existe* $X \in V$ *y* $\alpha, \beta \in (V \cup \Sigma)^*$ *tal que* $X \overset{+}{\Rightarrow} \alpha X \beta$. Como $\mathcal{L}(\mathcal{G})$ es un lenguaje infinito, entonces existe $w \in \mathcal{L}(\mathcal{G})$ tal que $|w| \geq 2^{|V|}$. Como la gramática está en forma normal de Chomsky, entonces al hacer el árbol de derivación de esta palabra necesariamente la altura del árbol será mayor a $|V|$. Por lo tanto existe una rama del árbol que tiene al menos $|V| + 1$ variables y por principio del palomar, hay una variable repetida en el árbol de derivación. Luego, existe X tal que

$$X \overset{+}{\Rightarrow} \alpha X \beta$$

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

- **(0.5 puntos)** En el caso converso, por argumentar que $u \neq \varepsilon$ o $v \neq \varepsilon$.
- **(0.5 puntos)** En el caso converso, por argumentar que X es alcanzable
- **(0.5 puntos)** En el caso converso, por establecer que X, α, β, α', β' son generadoras
- **(1 punto)** En el caso converso, por encontrar derivaciones para generar palabras infinitas
- **(0.5 puntos)** En el caso converso, por concluir que el lenguaje $\mathcal{L}(\mathcal{G})$ es infinito
- **(1 punto)** En el caso directo, por argumentar que como el lenguaje es infinito entonces existe una palabra suficientemente grande
- **(1 punto)** En el caso directo, por usar el arbol de derivacion de la palabra para acotar la cantidad de derivaciones
- **(1 punto)** En el caso directo, por usar principio del palomar para concluir que existe repetición de las palabras

Pregunta 3.2

Una posible solución para esta pregunta es entregar un contraejemplo. Sea la siguiente gramática:

$$S \to a$$
$$X \to XX$$

Está claro que el lenguaje generado por esta gramática es $\{a\}$ (lenguaje finito), pero sin embargo existe una variable X recursiva. Como existe al menos un caso en el que se incumple la propiedad del enunciado, entonces la propiedad no es necesariamente cierta si $\mathcal{G}$ tiene variables inútiles.

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

- **(4 puntos)** Por encontrar un contraejemplo
- **(2 puntos)** Por concluir que no se cumple la propiedad

Pregunta 4

Pregunta 4.1

Una posible solución para esta pregunta es modificar el algoritmo *eval-NFAonthefly* visto en clases.

#OutputNoVacio-AnnA($\mathcal{N} = (Q, \Sigma, \Lambda, \Delta, I, F)$, $d = a_0 a_1 \dots a_{n-1}$)

$S \leftarrow I$
for $i = 0$ to $n - 1$ **do**
  $S_{old} \leftarrow S$
  $S \leftarrow \varnothing$
  for $p \in S_{old}$ **do**
    $S \leftarrow S \cup \{q \mid (p, a_i, q) \in \Delta\}$
    $S \leftarrow S \cup \{q \mid \exists \lambda \in \Lambda.\, (p, a_i, \lambda, q) \in \Delta\}$
  end for
end for
return Check$(S \cap F \neq \varnothing)$

El algoritmo realiza lo pedido, ya que verifica si hay una ejecución de aceptación, con o sin anotación. Al igual que el algoritmo *eval-NFAonthefly*, se inicializa tomando los estados iniciales, luego se itera sobre la palabra y se guardan los estados a los que se llega desde el estado actual al leer esa letra. Además, este algoritmo revisa los estados a los que se llega desde el estado actual, leyendo la letra actual y si tiene una anotación. Finalmente, retorna si se llego a un estado final o no, que es lo pedido.

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

- **(2 puntos)** Por construir el algoritmo basado en *eval-NFAonthefly*.
- **(3 puntos)** Por ampliar las transiciones a las que tienen marca.
- **(1 punto)** Por explicación de correctitud.

Pregunta 4.2

Una posible solución para esta pregunta es modificar el algoritmo *eval-NFAonthefly* visto en clases de la siguiente forma:

#OutputNoTrivial-AnnA($\mathcal{N} = (Q, \Sigma, \Lambda, \Delta, I, F)$, $d = a_0 a_1 \dots a_{n-1}$)

$S \leftarrow I \times \{0\}$
for $i = 0$ to $n - 1$ **do**
  $S_{old} \leftarrow S$
  $S \leftarrow \varnothing$
  for $(p, 0) \in S_{old}$ **do**
    $S \leftarrow S \cup \{(q, 0) \mid (p, a_i, q) \in \Delta\}$
    $S \leftarrow S \cup \{(q, 1) \mid \exists \lambda \in \Lambda. (p, a_i, \lambda, q) \in \Delta\}$
  end for
  for $(p, 1) \in S_{old}$ **do**
    $S \leftarrow S \cup \{(q, 1) \mid (p, a_i, q) \in \Delta\}$
    $S \leftarrow S \cup \{(q, 1) \mid \exists \lambda \in \Lambda. (p, a_i, \lambda, q) \in \Delta\}$
  end for
end for
return Check$(S \cap (F \times \{1\}) \neq \varnothing)$

El algoritmo realiza lo pedido, ya que verifica si hay una ejecución de aceptación con alguna anotación. Para esto, el algoritmo mantiene pares (p, b) donde p es un estado y $b \in \{0, 1\}$ donde 0 significa que hay una ejecución que ha llegado hasta p pero que no tiene una anotación, y 1 significa que la ejecución tiene al menos una anotación (y por lo tanto, el resultado si llega a un estado final es distinto de ϵ). Al igual que el algoritmo *eval-NFAonthefly*, se inicializa tomando los estados iniciales y lo marca con 0, que significa que no ha habiado una anotación hasta ese momento. Luego se itera sobre la palabra y se consideran dos casos. Si hay un par $(p, 0)$ (el primer for) entonces se agregan los pares $(q, 0)$, si hay una transición sin anotación de p a q, y los pares $(q, 1)$ si hay una transición con anotación. En cambio, si hay un par $(p, 1)$ (el segundo for) entonces se agregan todos los pares $(q, 1)$ si hay una transición de p a q, sin importar si es de anotación o no. Finalmente, el algoritmo verifica si hay un par $(p, 1)$ con $p \in F$ en los estados S. Si es así, retorna `TRUE` ya que significa que hay una ejecución que entrega una anotación distinta de ϵ que es lo pedido.

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

- **(1 punto)** Por inicializar $S = I \times \{0\}$
- **(1 punto)** Por no marcar los estados a los que ha llegado sin marcas previas.
- **(1 punto)** Por marcar los estados a los que ha llegado sin marcas previas, pero tienen marca en esa transición.
- **(1 punto)** Por marcar los estados a los que ha llegado con marcas previas y mantenerles las marcas.
- **(1 punto)** Por devolver **Check**$(S \cap (F \times \{1\}) \neq \varnothing)$.
- **(1 punto)** Por explicación de correctitud.