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

IIC2223 — Teoría de Autómatas y Lenguajes Formales — 2' 2018

PAUTA I3

Pregunta 1

Sea $G = (V, \Sigma, P, S)$ una gramática con $\epsilon \notin \mathcal{L}(G)$. Demostraremos lo pedido por construcción de la gramática equivalente G'.

Primero, sea $G^* = (V, \Sigma, P^*, S)$ donde P^* es un conjunto de producciones que extiende a P, que es el menor conjunto que lo contiene y cerrado bajo clausura de las siguientes dos reglas:

1. $X \to Y,\ Y \to \gamma \in P^*$ entonces $X \to \gamma \in P^*$.
2. $Z \to \alpha X \beta,\ X \to \epsilon \in P^*$ entonces $Z \to \alpha\beta \in P^*$.

Luego, sea $G' = (V, \Sigma, P', S)$ el resultado de eliminar de G^* todas las producciones en vacio ($X \to \epsilon$) y unitarias ($X \to Y$).

Resta demostrar que $\mathcal{L}(G) = \mathcal{L}(G')$. Esto se puede separar en dos partes, demostrando lo mismo mediante G^*: $\mathcal{L}(G) = \mathcal{L}(G^*)$ y $\mathcal{L}(G^*) = \mathcal{L}(G')$.

Primero $\mathcal{L}(G) = \mathcal{L}(G^*)$:

- $\mathcal{L}(G) \subseteq \mathcal{L}(G^*)$: Para todo $w \in \mathcal{L}(G)$, existe un árbol de derivación de G para w. Como todas las producciones de P pertenecen a P^*, entonces ese árbol de derivación también es válido para G^*.
- $\mathcal{L}(G^*) \subseteq \mathcal{L}(G)$: Para todo $w \in \mathcal{L}(G^*)$, existe un árbol de derivación de G^* para w. Este puede utilizar producciones que no aparezcan originalmente en P, pero por construcción, cada regla $p \in P^*$ puede generarse a partir de una serie de producciones $p_1, \ldots, p_m \in P$ tal que si $\alpha \overset{G^*}{\Rightarrow} \beta$ mediante p, entonces $\alpha \overset{G}{\Rightarrow} \beta$ mediante composición de $p_1, \ldots, p_m$. Para cada producción usada en el árbol de w que no pertenece a P, se puede desenvolver como descrito y generar un árbol equivalente para w pero de G. Luego, $w \in \mathcal{L}(G)$.

Luego, queda demostrar que $\mathcal{L}(G') = \mathcal{L}(G^*)$. Este puede demostrarse mediante el argumento de que todo $w \in \mathcal{L}(G^*)$, su árbol de derivación de tamaño mínimo no utliza producciones en vacio ni unitarias. Sea $w \in \mathcal{L}(G^*)$, y sea T su árbol de derivación de tamaño mínimo. Demostraremos lo propuesto por contradicción:

- Supongamos T utiliza alguna producción unitaria $X \to Y$. Como T es válido, debe haber un subárbol con raiz en Y, que deriva por alguna regla $Y \to \gamma$. Por construcción, sabemos que $X \to \gamma$ pertenece a P^*, luego, es posible construir T' equivalente que utiliza esta última producción para colgar el subárbol originalmente de Y, desde X. T' también deriva a w pero es de menor tamaño que T al tener un nodo menos. Luego T no era de tamaño mínimo, llegando a una contradicción por nuestra suposición.

- Similarmente, supongamos T utiliza alguna producción en vacio $X \to \epsilon$. Luego, por la aparición de X en T, debe utilizar una producción que mencione a X: $Z \to \alpha X \beta$. Nuevamente, por construcción, la producción $Z \to \alpha\beta \in P^*$. Es posible construir entonces T', que reemplaza $Z \to \alpha X \beta$ por $Z \to \alpha\beta$, eliminando una hoja ϵ. Como la hoja es ϵ, esta no altera la palabra que deriva, w, y es de menor tamaño que T. Nuevamente esto es una contradicción por nuestra suposición.

Con esto, es posible terminar la demostración:

- $\mathcal{L}(G') \subseteq \mathcal{L}(G^*)$: Como todo w con árbol de derivación de G' utiliza producciones que también están en G^*, w también tiene un árbol de derivación valido en G^*.
- $\mathcal{L}(G^*) \subseteq \mathcal{L}(G')$: Para todo $w \in \mathcal{L}(G^*)$, este tiene al menos un árbol de derivación, y alguno de estos debe ser mínimo. Como este no utiliza producciones en vacio o unitarias, entonces el árbol también es valido en G'.

Como G^*: $\mathcal{L}(G) = \mathcal{L}(G^*)$ y $\mathcal{L}(G^*) = \mathcal{L}(G')$, entonces demostramos que $\mathcal{L}(\mathcal{G}) = \mathcal{L}(G')$, donde G' no posee producciones unitarias ni en vacio.

Considerado esto, la distribución de puntaje es:

- **(2 puntos)**: Por la construcción de G^*.
- **(1 punto)**: Por la construcción de G'.
- **(1 punto)**: Por demostrar que $\mathcal{L}(G) = \mathcal{L}(G^*)$.
- **(2 puntos)**: Por demostrar que $\mathcal{L}(G') = \mathcal{L}(G^*)$.

Alternativamente, considerando que muchas soluciones construyeron directamente G' y demostró directamente $\mathcal{L}(\mathcal{G}) = \mathcal{L}(G')$, se consideró:

- **(3 puntos)**: Por la construcción de G'.
- **(1.5 puntos)**: Por demostrar que $\mathcal{L}(G) \subseteq \mathcal{L}(G')$.
- **(1.5 puntos)**: Por demostrar que $\mathcal{L}(G') \subseteq \mathcal{L}(G^*)$.

Pregunta 2

Pregunta 2.1

En esta pregunta se debía simplemente aplicar el algoritmo CKY. La tabla final después de aplicar el algoritmo es

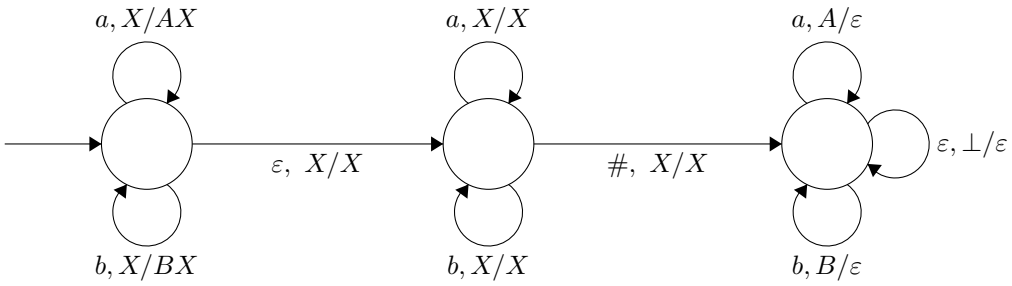
i=4	{S}				
i=3	∅	{T}			
i=2	{S}	∅	{S}		
i=1	∅	{T}	∅	{T}	
i=0	{S}	{B}	{S}	{B}	{S}
	a	b	a	b	a

Dado lo anterior, el puntaje asignado es el siguiente:

- **(2 puntos)** Por ejecutar correctamente el algoritmo.
- **(1 punto)** Por la explicación.

Pregunta 2.2

Un posible autómata apilador es



Lo que hace es leer un prefijo de u y guardarlo en el stack. Luego revisa que el reverso del prefijo leído sea igual a lo que está después del símbolo #. El no-determinismo hace que el autómata acepte siempre que exista un prefijo de u que haga match con el reverso de lo que hay después del separador #.

Dado lo anterior, el puntaje asignado es el siguiente:

- **(2.5 puntos)** Por construir correctamente el autómata.
- **(0.5 puntos)** Por la explicación.

Pregunta 3

Dado un autómata apilador $\mathcal{P} = (P, \Sigma, \Gamma, \Delta, p_0, \perp)$ y una expresión regular R es claro que existe un autómata determinista $\mathcal{A} = (Q, \Gamma, \delta, q_0, F)$ que cumple $\mathcal{L}(\mathcal{A}) = \mathcal{L}(R)$. Definimos un nuevo autómata apilador

$$\mathcal{P}' = (P \uplus Q \uplus \{p_0'\},\ \Sigma,\ \Gamma \uplus \{\perp'\},\ \Delta',\ p_0',\ \perp'),$$

tal que:

- $\Delta \subseteq \Delta'$, es decir, todas las transiciones de $\mathcal{P}$ están presentes en $\mathcal{P}'$
- $p_0' \perp' \xrightarrow{\epsilon} p_0 \perp\perp'$
- $\forall p \in P.\ \forall A \in \Gamma.\ pA \xrightarrow{\epsilon} q_0 A$
- $\forall q_1, q_2 \in Q.\ \forall A \in \Gamma.\ \delta(q_1, A) = q_2 \implies q_1 A \xrightarrow{\epsilon} q_2$
- $\forall q_f \in F.\ q_f \perp' \xrightarrow{\epsilon} q_f$

La idea es crear un autómata apilador que primero coloque el símbolo de inicio de stack $\perp$ del autómata $\mathcal{P}$ (distinto al símbolo de inicio de stack $\perp'$ del nuevo autómata) en su stack, para luego comienzar a simular las transiciones del autómata $\mathcal{P}$. En un punto cualquiera de la ejecución se decide de manera no-determinista comenzar a simular el autómata $\mathcal{A}$ consumiendo los símbolos presentes en el stack. Finalmente, si se llega a un $q_f \in F$ y se consumió por completo el stack de $\mathcal{P}$, aceptamos por stack vacío. Es importante notar que no es necesario definir un conjunto de estados finales para $\mathcal{P}$ y $\mathcal{P}'$ ya que no trabajamos con aceptación por estado final para estos dos autómatas.

Con la definición para $\mathcal{P}'$ anterior, solo queda demostrar que $\mathcal{L}_R(\mathcal{P}) = \mathcal{L}_\perp(\mathcal{P}')$. Tenemos que $w \in \mathcal{L}_\perp(\mathcal{P}')$ si, y solo si, existe una ejecución ρ de $\mathcal{P}'$ sobre $w = a_1 \cdots a_n$ tal que

$$\rho: \qquad p_0' \perp' \xrightarrow{\epsilon} p_0 \perp\perp' \xrightarrow{b_1} p_1\gamma_1 \xrightarrow{b_2} \cdots \xrightarrow{b_m} p_m\gamma_m \xrightarrow{\epsilon} q_0\gamma_m \xrightarrow{\epsilon} q_1\gamma_{m+1} \xrightarrow{\epsilon} \cdots \xrightarrow{\epsilon} q_l \perp' \xrightarrow{\epsilon} q_l$$

tal que

- $b_1, \ldots, b_m \in \Sigma \cup \{\epsilon\}$
- $p_i \in P \ \forall i \in \{1, \ldots, m\}$
- $q_i \in Q \ \forall i \in \{1, \ldots, l\}$
- $\exists (p_{i-1}, b_i, A_i, p_i, \alpha_i) \in \Delta$ que satisface:
  - $\text{top}(\gamma_{i-1}) = A_i$
  - $\gamma_i = \alpha_i \cdot \text{tail}(\gamma_{i-1})$

  $\forall i \in [1, m]$
- $\delta(q_{i-1}, A_i) = q_i \ \forall i \in \{1, \ldots, l\}$ con $\text{top}(\gamma_{m+i-1}) = A_i$
- $b_1 \cdots b_m = a_1 \cdots a_n$
- $q_l \in F$

lo anterior es cierto si, y solo si:

$$\rho': \qquad p_0 \perp \xrightarrow{b_1} p_1\gamma_1 \xrightarrow{b_2} \cdots \xrightarrow{b_m} p_m\gamma_m$$

es una ejecución de $\mathcal{P}$ sobre w, y

$$\rho'': \qquad q_0 \xrightarrow{\text{top}(\gamma_m)} q_1 \xrightarrow{\text{top}(\gamma_{m+1})} \cdots \xrightarrow{\text{top}(\gamma_{m+l})} q_l$$

es una ejecución de aceptación $\mathcal{A}$ sobre γ_m, ya que $\gamma_m = \text{top}(\gamma_m) \cdot \text{top}(\gamma_{m+1}) \cdots \text{top}(\gamma_{m+l})$ y $q_l \in F$. Dicho esto, con ρ, ρ' y ρ'' recién descritos:

$$\begin{aligned}
w \in \mathcal{L}_\perp(\mathcal{P}') &\Leftrightarrow \rho \text{ es ejecución de aceptación por stack vacío de } \mathcal{P}' \text{ sobre } w \\
&\Leftrightarrow \rho' \text{ es ejecución de } \mathcal{P} \text{ sobre } w \text{ y } \rho'' \text{ es ejecución de aceptación } \mathcal{A} \text{ sobre } \gamma_m \\
&\Leftrightarrow \rho' \text{ es ejecución de } \mathcal{P} \text{ sobre } w \text{ y } \gamma_m \in \mathcal{L}(\mathcal{A}) \\
&\Leftrightarrow \rho' \text{ es ejecución de } \mathcal{P} \text{ sobre } w \text{ y } \gamma_m \in \mathcal{L}(R) \\
&\Leftrightarrow \rho' \text{ es ejecución de aceptación por } R\text{-stack de } \mathcal{P} \text{ sobre } w \\
&\Leftrightarrow w \in \mathcal{L}_R(\mathcal{P})
\end{aligned}$$

Dado lo anterior, el puntaje asignado es el siguiente:

- **(1 punto)** Por mencionar correctamente que se debe agregar un nuevo símbolo inicial de stack $\perp'$ al nuevo autómata $\mathcal{P}'$.

- **(1 punto)** Por decir que la relación de transición Δ original será un subconjunto de la nueva relación de transición Δ'.
- **(1.5 puntos)** Por la idea de que se debe realizar un salto no-determinista desde los estados P (del autómata original) a los estados Q (del autómata que define R).
- **(1.5 puntos)** Por la idea de que se debe simular el autómata $\mathcal{A}$ (que define R) consumiendo el stack.
- **(1 punto)** Por demostrar que $\mathcal{L}_R(\mathcal{P}) = \mathcal{L}_\perp(\mathcal{P}')$.

Pregunta 4

Pregunta 4.1

Similar a la demostración del lema de bombeo para gramáticas libres de contexto, lo primero que debemos hacer es elegir un N que nos ayude a demostrar lo pedido. En este caso, podemos notar que si una gramática es lineal, entonces su arbol de derivación tiene solo una variable por nivel. Por lo tanto, sea m la máxima cantidad de terminales que puede producir una variable en nuestra gramática, definimos nuestro N como $N = (|V| + 1) \cdot m$.

Luego, se debe mencionar que se pueden eliminar las producciones unitarias de la gramática y esta seguiría siendo lineal, por lo que de ahora en adelante asumiremos que no existen producciones unitarias en nuestra gramática.

Si el lenguaje de nuestra gramática es finito, entonces el lema se cumple trivialmente, por lo que solo nos interesa el caso donde el lenguaje es infinito. Sea z una palabra que pertenece a nuestra gramática tal que $|z| \geq N$. Como el arbol de derivación tiene solo una variable por nivel, la cantidad máxima de terminales producidas por cada variable es m, y $|z| > N$, se tiene que el arbol de derivación de z pasa por al menos $|V| + 1$ variables.

Sea X la primera variable que se repite de **arriba hacia abajo** en el arbol de derivación. En el peor caso, habrán $|V|$ variables antes de la segunda X, y por lo tanto sean u, y las terminales que quedan a la izquierda y a la derecha justo antes de llegar a la primera X respectivamente, y v, x las terminales que se producen a la izquierda y a la derecha desde la primera X hasta antes de llegar a la segunda X. Se tiene que:

$$|uv| \leq |V| \cdot m \leq N$$
$$|xy| \leq |V| \cdot m \leq N$$

Y como no hay producciones unitarias, X debe producir alguna terminal y por lo tanto $vx \neq \epsilon$. Finalmente se debía mencionar que era posible intercambiar los arboles de derivacion de la variable repetida X y la palabra obtenida seguiría perteneciendo al lenguaje.

Dado lo anterior, el puntaje asignado es el siguiente:

- **(1 punto)** Por elegir y justificar correctamente el N.
- **(1 punto)** Por usar correctamente el N para mostrar que se repiten variables.
- **(1 punto)** Por el resto de la demostración.

Pregunta 4.2

La solución más simple consistía en usar el lema de bombeo entregado en la parte 4.1. Para esto, elegíamos la palabra $a^N b^N c^N d^N$ y luego notabamos que para cualquier división posible, uv solo contendrá letras a y xy

solo contendrá letras d. Como $vx \neq \epsilon$, v o x deben contener al menos una letra. Por lo tanto, al bombear con cualquier número distinto de 1, estaríamos modificando la cantidad de letras a o d (o ambas). En cualquier caso, la palabra resultante no pertenece al lenguaje, y entonces se concluye que el lenguaje no es lineal.

Dado lo anterior, el puntaje asignado es el siguiente:

- **(1 punto)** Por escoger la palabra correctamente.
- **(1 punto)** Por descomponer correctamente la palabra y explicar por que no se puede descomponer de otra forma.
- **(1 punto)** Por aplicar el bombeo y entregar una palabra que no se encuentra en el lenguaje.