

Autómatas de Pila

Pablo Verdes

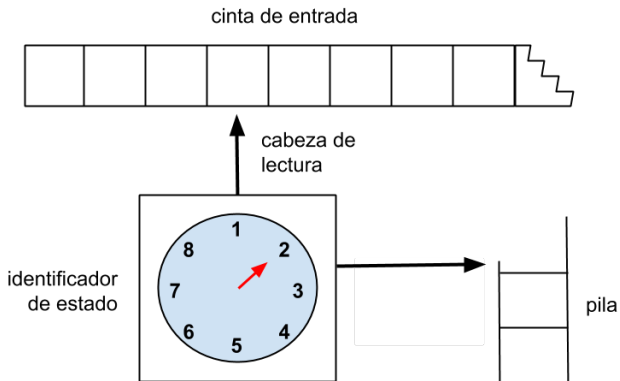
LCC

8 de junio de 2017

Introducción

- En la unidad anterior estudiamos los AEF y su poder de reconocimiento de lenguajes —en particular, sus límites.
- Allí, intentamos aumentar la capacidad de reconocimiento de los AEF introduciendo no determinismo en sus sistemas de transiciones.
- Sin embargo, encontramos que los AEFND reconocen el mismo conjunto de lenguajes que los AEF, esto es, el conjunto de los lenguajes regulares \mathcal{L}_3 .
- En esta unidad estudiaremos una generalización de los AEFND, llamada **autómatas de pila**.
- El ingrediente principal en este tipo de autómatas es la presencia de una **memoria** (pila) donde se puede almacenar información a medida que se procesa una palabra de entrada.
- Recordemos que el problema de los AEF era que no podían ‘recordar’ la cantidad de símbolos leídos.

Introducción

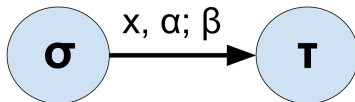


Introducción

- Los símbolos que pueden almacenarse en esta pila constituyen un conjunto finito. Este conjunto puede incluir, o no:
 - ▶ los símbolos del alfabeto de la máquina
 - ▶ símbolos adicionales que la máquina utiliza como marcadores internos, por ej. para separar secciones que tengan interpretaciones distintas o para indicar que la pila está vacía (usualmente, el símbolo #).
- Las transiciones que ejecutan los AP deben ser variantes de la siguiente secuencia básica:
 - 1 leer un símbolo de la entrada
 - 2 extraer un símbolo de la pila
 - 3 insertar un símbolo en la pila
 - 4 pasar a un nuevo estado

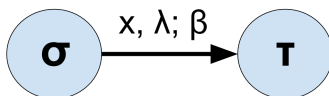
Autómatas de pila

- A este proceso se lo representa con la notación $(\sigma, x, \alpha; \beta, \tau)$, donde:
 - ▶ σ es el estado actual
 - ▶ x es el símbolo del alfabeto que se lee en la entrada
 - ▶ α es el símbolo que se extrae de la pila
 - ▶ β es el símbolo que se inserta en la pila
 - ▶ τ es el nuevo estado al que pasa el autómata
- Gráficamente: arcos con etiquetas $x, \alpha; \beta$

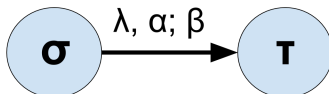


Autómatas de pila

- Se obtienen variantes de este proceso básico de transición permitiendo que las transiciones lean, extraigan o inserten la cadena vacía.
- Ejemplos:
 - ▶ En el estado σ : si se lee x de la entrada, no extraer nada de la pila, insertar β en la pila y pasar al estado τ .



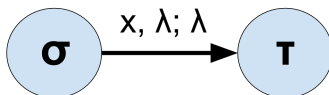
- ▶ En el estado σ : no leer nada de la entrada, extraer α de la pila, insertar β en la pila y pasar al estado τ .



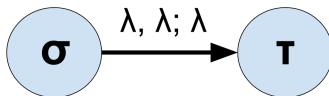
Autómatas de pila

- Ejemplos:

- ▶ En el estado σ : si se lee x de la entrada, no extraer ni insertar nada en la pila y pasar al estado τ (este tipo de transición ignora la pila: se reduce al tipo de transición de un AEF).



- ▶ En el estado σ : no leer nada de la entrada, no extraer ni insertar nada en la pila y pasar al estado τ (transición espontánea).



Autómatas de pila

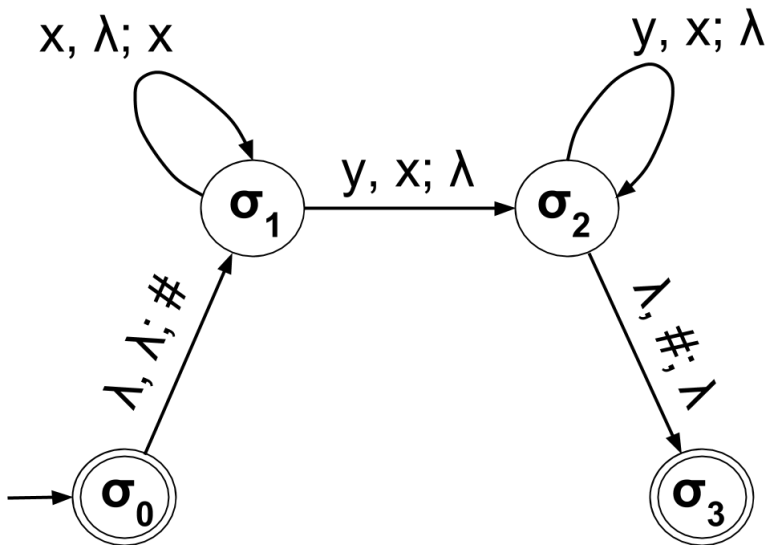
Definición: Un **autómata de pila (AP)** es una tupla

$$A = (S, \Sigma, \Gamma, T, \sigma, Ac)$$

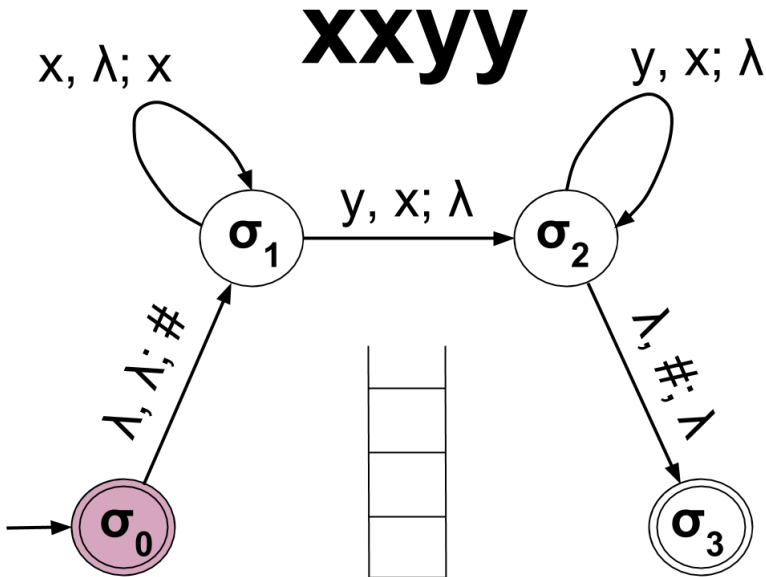
donde:

- S es un conjunto finito de **estados**
- Σ es un conjunto finito de **símbolos de entrada**
- Γ es un conjunto finito de **símbolos de pila**
- $T \subseteq S \times (\Sigma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \times (\Gamma \cup \{\lambda\}) \times S$
es una **relación de transición**
- $\sigma \in S$ es el **estado inicial**
- $Ac \subseteq S$ es un conjunto de **estados de aceptación**

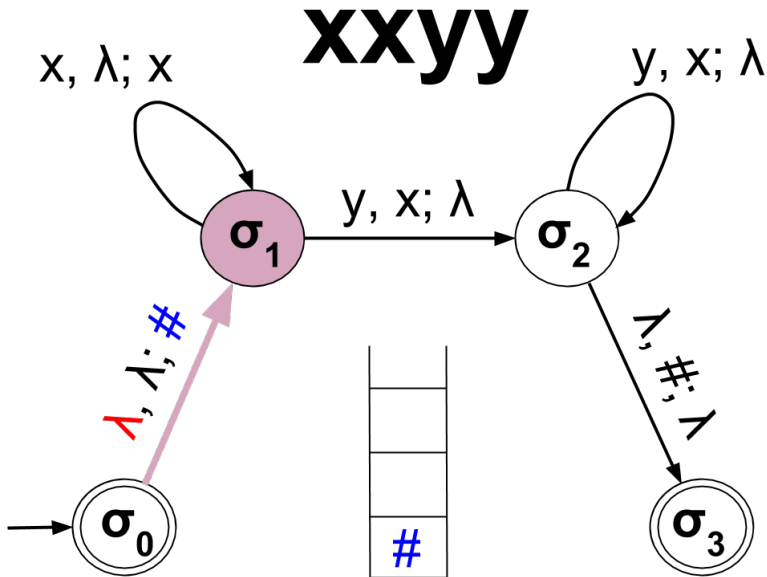
Ejemplo



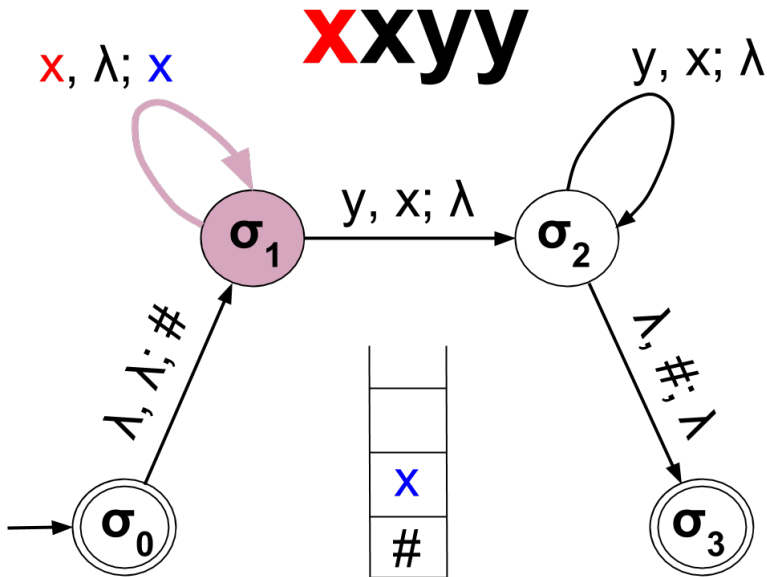
Ejemplo



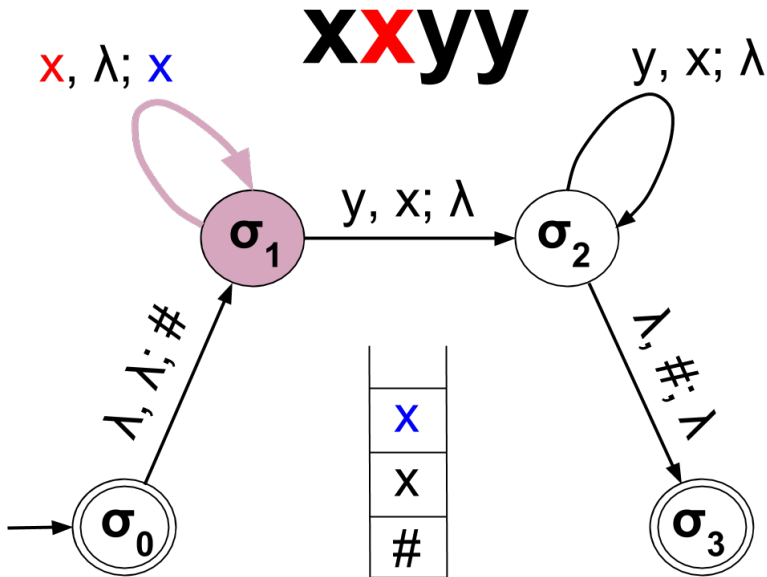
Ejemplo



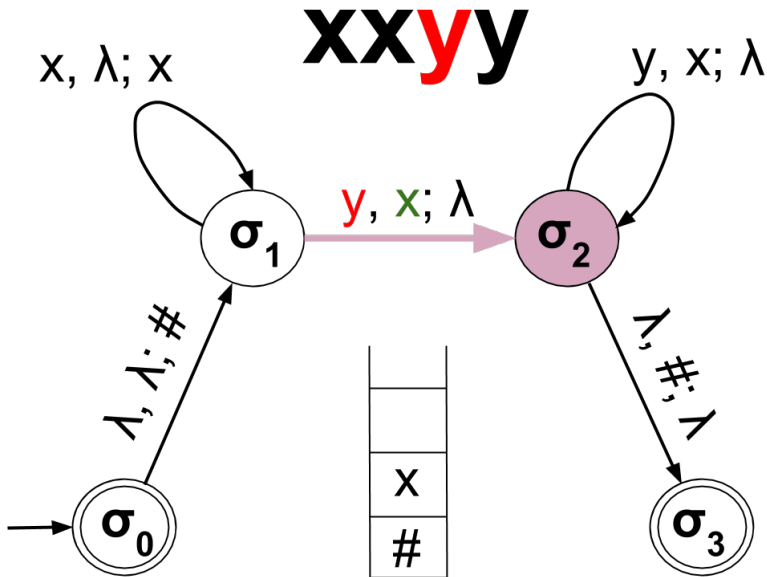
Ejemplo



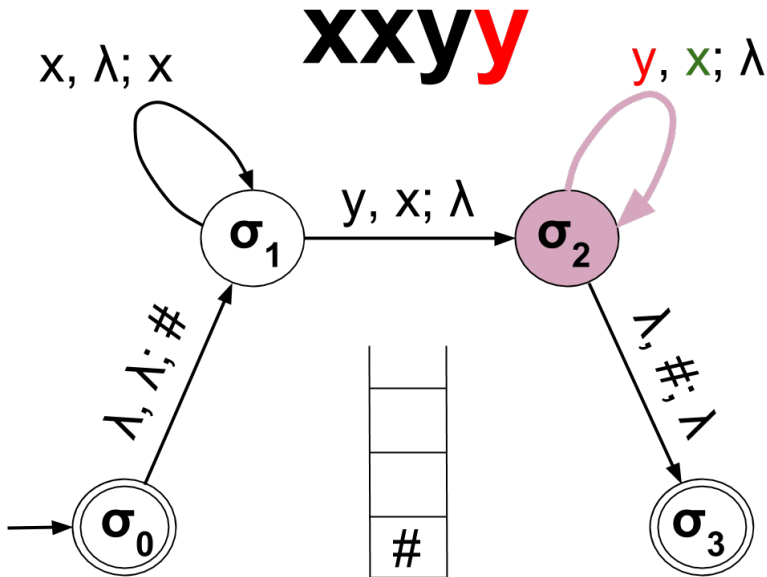
Ejemplo



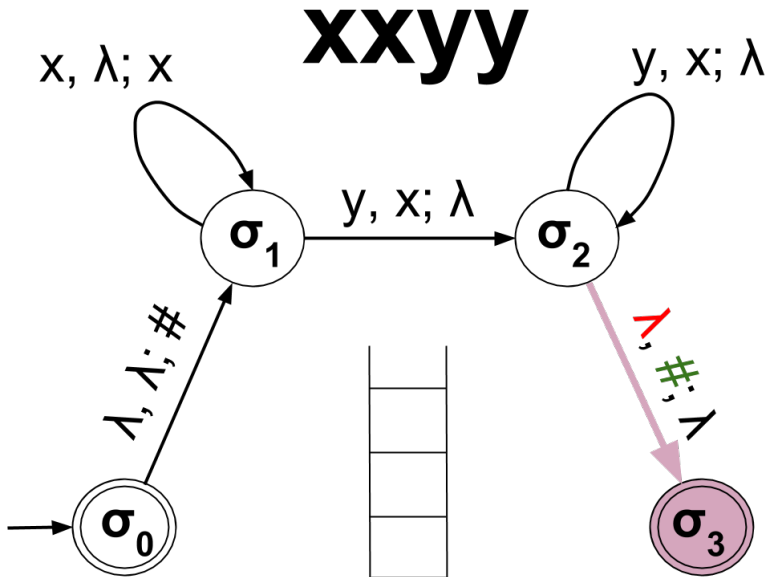
Ejemplo



Ejemplo



Ejemplo



AP como aceptadores de lenguajes

- **Definición:** Una **configuración** de un autómata de pila $A = (S, \Sigma, \Gamma, T, \sigma, Ac)$ es un elemento de $\mathcal{C}_A = S \times \Sigma^* \times \Gamma^*$.
- La idea es que una configuración (σ, x, γ) indica que el autómata A está en el estado σ , le falta leer la cadena x de la entrada, y el contenido completo de la pila (de arriba hacia abajo) es γ . Esta información es suficiente para predecir lo que ocurrirá en el futuro.
- **Definición:** Para un AP $A = (S, \Sigma, \Gamma, T, \sigma, Ac)$, la relación **lleva en un paso** $\Rightarrow_A \subseteq \mathcal{C}_A \times \mathcal{C}_A$ se define de la siguiente manera:

$$(\sigma, xw, \alpha\gamma) \Rightarrow_A (\tau, w, \beta\gamma) \quad \text{sii} \quad (\sigma, x, \alpha; \beta, \tau) \in T$$

- **Definición:** La relación **lleva en uno o más pasos** \Rightarrow_A^* se define recursivamente a partir de la relación lleva en un paso \Rightarrow_A de manera análoga a lo hecho para la función de transición f de los AEF.
- No usaremos el subíndice A cuando sea obvio del contexto.

AP como aceptadores de lenguajes

- **Definición:** Sea $A = (S, \Sigma, \Gamma, T, \sigma, Ac)$ un AP. El **lenguaje aceptado** por A es el conjunto

$$\mathcal{AC}(A) = \{w \in \Sigma^* \mid (\sigma, w, \lambda) \Rightarrow^* (\tau, \lambda, \gamma), \gamma \in \Gamma^*, \tau \in Ac\}$$

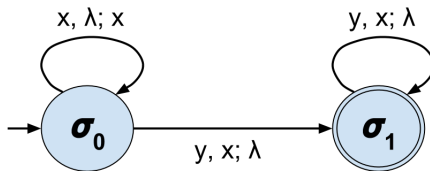
- Es decir, una cadena w será aceptada por un AP si, arrancando desde su estado inicial y con la pila vacía, es posible que el autómata llegue a un estado de aceptación después de leer toda la cadena.
- Nótese que esto no quiere decir que la máquina deba llegar a un estado de aceptación al leer el último carácter de w . ¿Por qué?
- Porque después de leer el último carácter de la entrada, el AP podría ejecutar varias transiciones de la forma $(\sigma, \lambda, \alpha; \beta, \tau)$, cambiando de estado (y tal vez operando sobre la pila, por ejemplo para vaciarla) pero sin leer ningún carácter de la entrada.

AP como aceptadores de lenguajes

- Consideremos un autómata de pila que sólo tiene transiciones del tipo $(\sigma, x, \lambda; \lambda, \tau)$.
- Los pasos de este tipo ignoran que la máquina tiene una pila. Sólo dependen del estado y símbolo de entrada actuales.
- Vemos entonces que los AEFND son un caso particular de AP.
- Por lo tanto, los lenguajes regulares son un subconjunto de los lenguajes aceptados por los AP: $\mathcal{L}_3 \subseteq \mathcal{AC}(AP)$.
- Más aún, acabamos de ver que el lenguaje $\{x^n y^n \mid n \in \mathbb{N}_0\}$ es aceptado por un AP, y como por el lema del bombeo sabemos que no es regular, concluimos que $\mathcal{L}_3 \subset \mathcal{AC}(AP)$.
- Veamos otro ejemplo.

AP como aceptadores de lenguajes

- Ejemplo:



$$\mathcal{AC}(A) = \{xy, x^n y, x^n y^k \text{ si } k \leq n\} = \{x^n y^k \mid n, k \in \mathbb{N}, k \leq n\}$$

- Obs. que si $k < n$ el autómata no vacía su pila.
- Si quisiéramos componer AP, por ej. para concatenar lenguajes, necesitaríamos que al llegar al estado de aceptación del primer AP la pila estuviera vacía.
- Veamos que siempre se puede retocar un AP de manera de aceptar el mismo lenguaje pero vaciando su pila.

AP como aceptadores de lenguajes

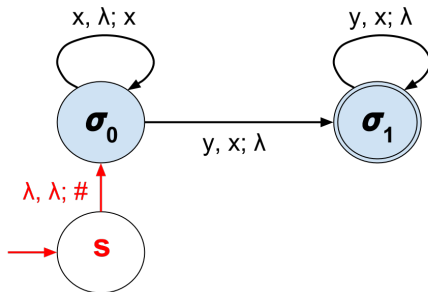
Teorema: Para cada $A \in AP$, existe $A' \in AP$ que vacía su pila y tal que

$$\mathcal{AC}(A') = \mathcal{AC}(A).$$

D/ Sea $A = (S, \Sigma, \Gamma, T, \sigma, Ac)$ un AP.

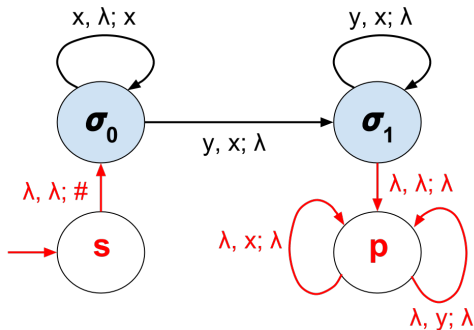
La idea para construir A' es la siguiente:

- El estado inicial de A deja de serlo. Introducimos un nuevo estado inicial y una transición del nuevo al anterior que lo único que hace es insertar en la pila un marcador $\#$ (sup. que $\# \notin \Gamma$).



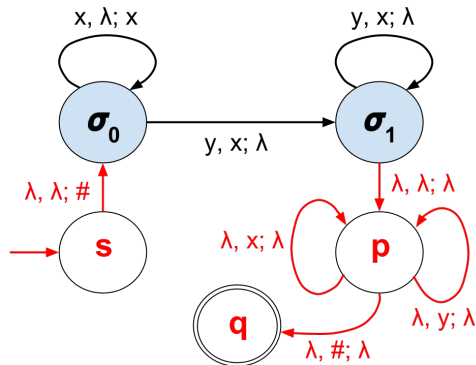
AP como aceptadores de lenguajes

- Los estados de aceptación de A dejan de serlo. Introducimos un nuevo estado, p , junto con las transiciones que permiten a la máquina pasar de cada uno de los antiguos estados de aceptación al nuevo estado p sin leer, extraer o insertar un símbolo.
- Vaciamos la pila sin salir del estado p introduciendo transiciones de la forma $p \xrightarrow{\lambda, x; \lambda} p$ para cada x en $\Gamma - \{\#\}$.



AP como aceptadores de lenguajes

- Agregamos un nuevo (y único) estado de aceptación q y la transición $p \xrightarrow{\lambda, \#; \lambda} q$ que borra el marcador $\#$, vaciando completamente la pila.



- A continuación, formalizamos estas ideas.

AP como aceptadores de lenguajes

Dado $A = (S, \Sigma, \Gamma, T, \sigma, Ac)$, definimos $A' = (S', \Sigma, \Gamma', T', \sigma', Ac')$ donde:

- $S' = S \cup \{s, p, q\}$, donde $s, p, q \notin S$
- $\Gamma' = \Gamma \cup \{\#\}$, donde $\# \notin \Gamma$
- $\sigma' = s$
- $Ac' = \{q\}$
-

$$T' = T \cup \{(s, \lambda, \lambda; \#, \sigma)\} \quad (1)$$

$$\cup \{(\tau, \lambda, \lambda; \lambda, p) \mid \tau \in Ac\} \quad (2)$$

$$\cup \{(p, \lambda, \alpha; \lambda, p) \mid \alpha \in \Gamma - \{\#\}\} \quad (3)$$

$$\cup \{(p, \lambda, \#; \lambda, q)\} \quad (4)$$

AP como aceptadores de lenguajes

Ahora debemos mostrar que

$$\mathcal{AC}(A) = \mathcal{AC}(A').$$

① Veamos que $\mathcal{AC}(A) \subseteq \mathcal{AC}(A')$:

Sea $\alpha \in \mathcal{AC}(A)$. Sabemos que, partiendo del estado inicial σ con la pila vacía, α nos lleva en uno o más pasos a un estado de aceptación. En términos de configuraciones, podemos entonces escribir:

$$(\sigma, \alpha, \lambda) \Rightarrow^* (\tau, \lambda, \gamma) \quad \text{donde} \quad \tau \in Ac, \gamma \in \Gamma^*$$

Por lo tanto, en A' tenemos la derivación:

$$\begin{aligned} (s, \alpha, \lambda) &\Rightarrow^{(1)} (\sigma, \alpha, \#) \Rightarrow^* (\tau, \lambda, \gamma\#) \Rightarrow^{(2)} (p, \lambda, \gamma\#) \\ &\Rightarrow^{(3)*} (p, \lambda, \#) \Rightarrow^{(4)} (q, \lambda, \lambda) \end{aligned}$$

Como $q \in Ac'$, concluimos que $\alpha \in \mathcal{AC}(A')$ y la pila queda vacía.

② El caso $\mathcal{AC}(A) \supseteq \mathcal{AC}(A')$ se demuestra análogamente. □

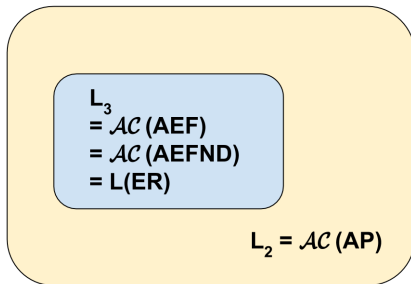
AP como aceptadores de lenguajes

- **Teorema 1:** Sea G una gramática independiente de contexto. Entonces existe un autómata de pila A tal que $L(G) = \mathcal{AC}(A)$.

D/ ver Brookshear, pág. 85.

- **Teorema 2:** Sea A un autómata de pila. Entonces existe una gramática independiente de contexto G tal que $L(G) = \mathcal{AC}(A)$.

D/ ver Brookshear, pág. 90.



Límites de los lenguajes independientes de contexto

- ¿Existen lenguajes fuera de \mathcal{L}_2 ? La respuesta es *sí*.
- Un ejemplo es $\{a^n b^n c^n \mid n \in \mathbb{N}\}$. Para demostrarlo usaremos el:
- **Lema de bombeo:** Sea $L \in \mathcal{L}_2$. Si L es infinito existe $\alpha \in L$ de la forma $\alpha = xuyvz$, donde $uv \neq \lambda$, tal que $xu^n yv^n z \in L \quad \forall n \in \mathbb{N}$.

D/ Sea $L \in \mathcal{L}_2$ dado. Sabemos que existe una gramática independiente de contexto $G = (N, T, P, \sigma)$ tal que $L(G) = L$.

Informalmente, la idea de la demostración es la siguiente:

- ▶ L infinito $\Rightarrow L$ contiene palabras de longitud arbitrariamente grande
- ▶ Palabras largas requieren derivaciones largas (de muchos pasos).
- ▶ Si la palabra es suficientemente larga, en algún momento forzosamente tendremos que volver a aplicar alguna de las reglas de producción.
- ▶ La aparición de ese bucle es el que permite bombear caracteres.

Límites de los lenguajes independientes de contexto

Sea m el número máximo de símbolos de $N \cup T$ que aparecen en el lado derecho de las reglas de producción de G : $m = \max\{|\delta|, A \rightarrow \delta \in P\}$.

Obs. que m es la mayor cantidad de símbolos por los que se puede reemplazar un no terminal. Por lo tanto, al aplicar i reglas de producción cualesquiera, la longitud de la cadena resultante es a lo sumo m^i .

Llamemos k a la cantidad de símbolos no terminales de G : $k = |N|$.

Sea $\alpha \in L$ una cadena tal que $|\alpha| > m^k$, que existe pues L es infinito.

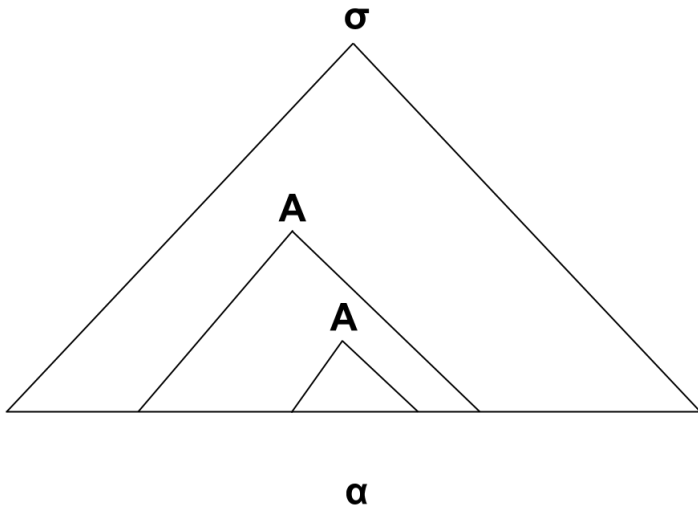
¿Cuántas veces se deben aplicar las reglas de producción para derivar α ?
Llamémoslo i .

$m^i \geq |\alpha| > m^k \Rightarrow i > k$, es decir, para formar α debieron aplicarse más de k reglas de producción (debieron expandirse más de k no terminales).

Como hay k no terminales, por lo menos uno debió expandirse 2 veces.
Llamémoslo A .

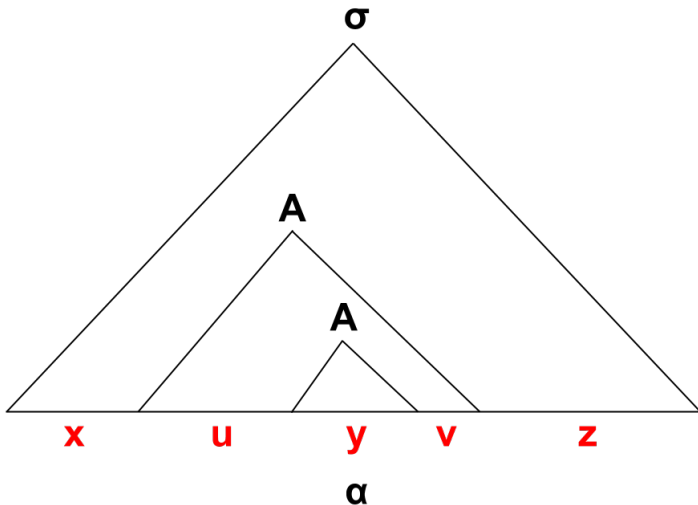
Límites de los lenguajes independientes de contexto

Representando esquemáticamente la derivación de α como un árbol:



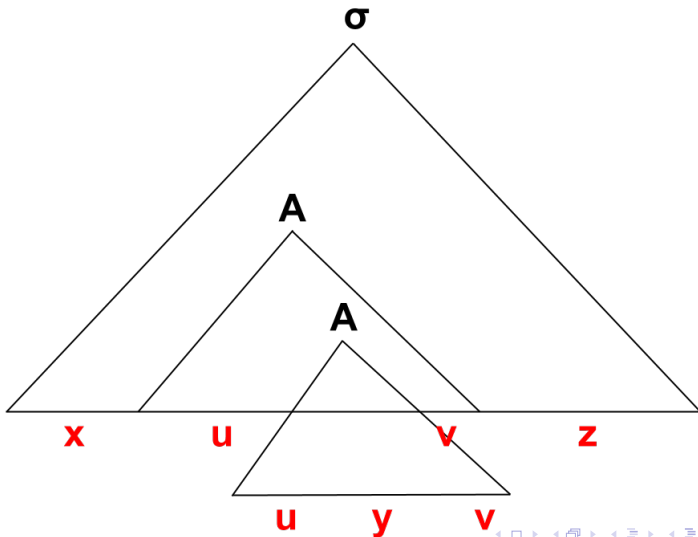
Límites de los lenguajes independientes de contexto

Representando esquemáticamente la derivación de α como un árbol:



Límites de los lenguajes independientes de contexto

Representando esquemáticamente la derivación de α como un árbol:



Límites de los lenguajes independientes de contexto

Corolario: $L = \{a^n b^n c^n \mid n \in \mathbb{N}\} \notin \mathcal{L}_2$

D/ Por el absurdo, supongamos que $L \in \mathcal{L}_2$.

- Como L es infinito, podemos aplicar el lema de bombeo y concluir que existe una cadena $xuyvz \in L$ tal que $xu^n yv^n z \in L \ \forall n \in \mathbb{N}$.
- $uv \neq \lambda$. Sin perder generalidad, supongamos que sea $u \neq \lambda$.
- Hay 2 posibilidades:
 - 1 u está formado por un sólo símbolo (a, b ó c): al bombearlo se obtiene una palabra que no mantiene la igualdad entre exponentes y por lo tanto $\notin L$
 - 2 u está formado por más de un símbolo (por ejemplo, $u = ab$): al bombearlo se obtiene una palabra que altera el orden de los símbolos y por lo tanto $\notin L$
- Absurdo, luego $L \notin \mathcal{L}_2$. □