Tema 2: Lenguajes Independientes del Contexto

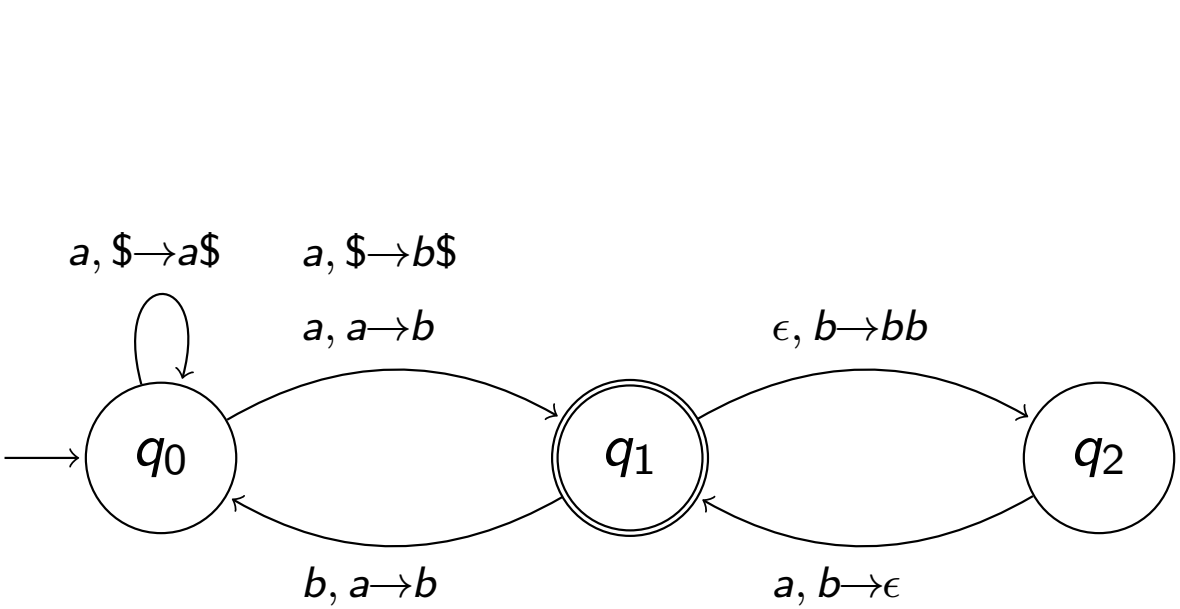
Lección 2.2

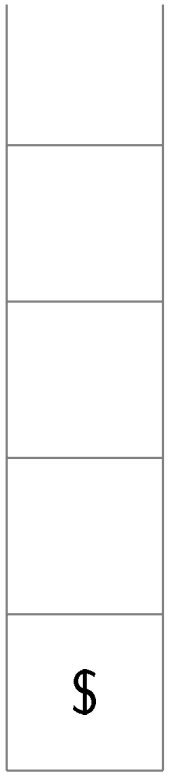
Autómatas de Pila (AdP)

Jordi Bernad

Índice

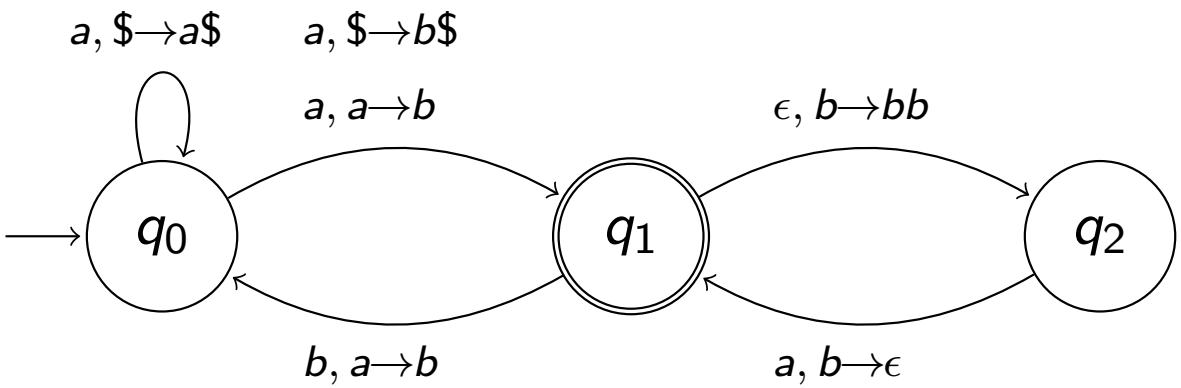
Un ejemplo de autómata de pila





- Un autómata de pila (AdP) es una autómata no determinista junto con una pila.
- El fondo de la pila está marcado por un símbolo especial, $

Un ejemplo de autómata de pila

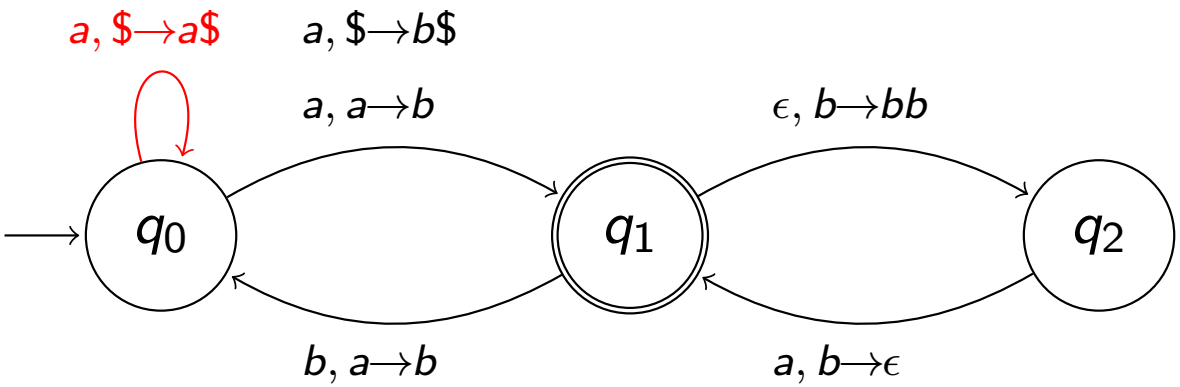


- Las transiciones entre estados

$$s_1, s_2 \rightarrow z$$

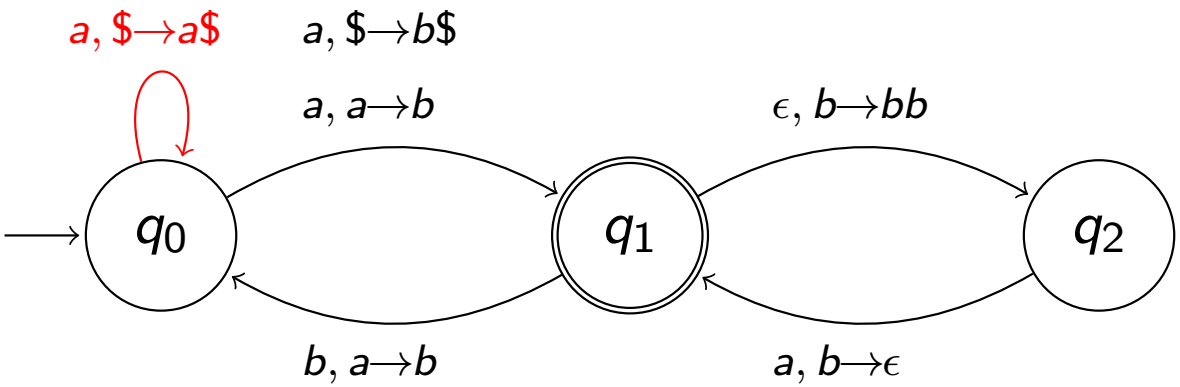
si lees de la entrada s_1 y en la cima de la pila hay s_2, cambia de estado y sustituye la cima de la pila por z

Ejemplo de transición



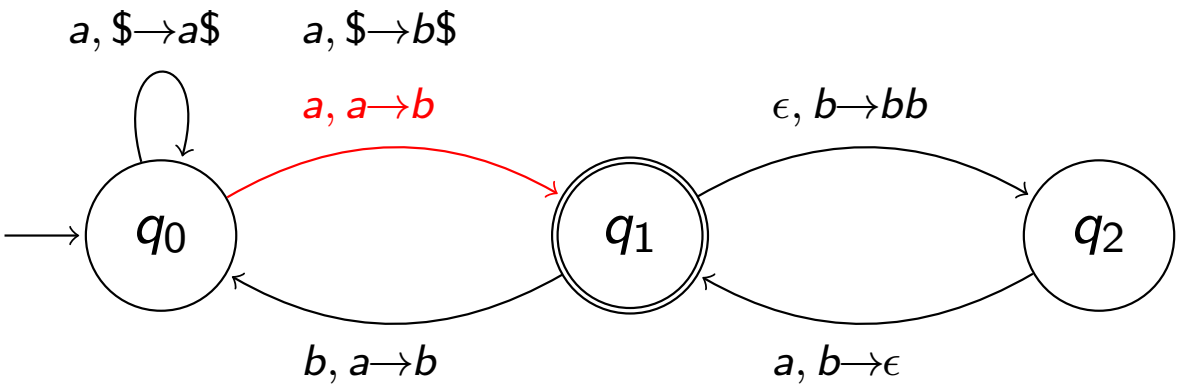
- Transición $a, \$ \rightarrow a\$$
- Si estamos en el estado q_0, se lee de la entrada una a y en la pila hay un \$ (pila vacía),

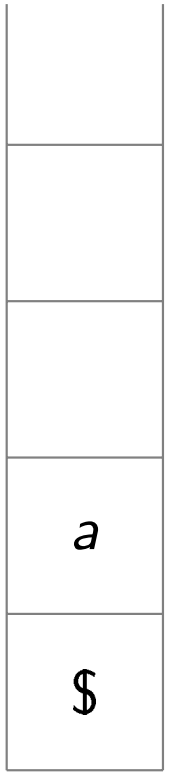
Ejemplo de transición



- Transición $a, \$ \rightarrow a\$$
- Si estamos en el estado q_0, se lee de la entrada una a y en la pila hay un \$ (pila vacía),
- Cambia al estado q_0 y sustituye en la pila \$ por $a\$$: apila una a

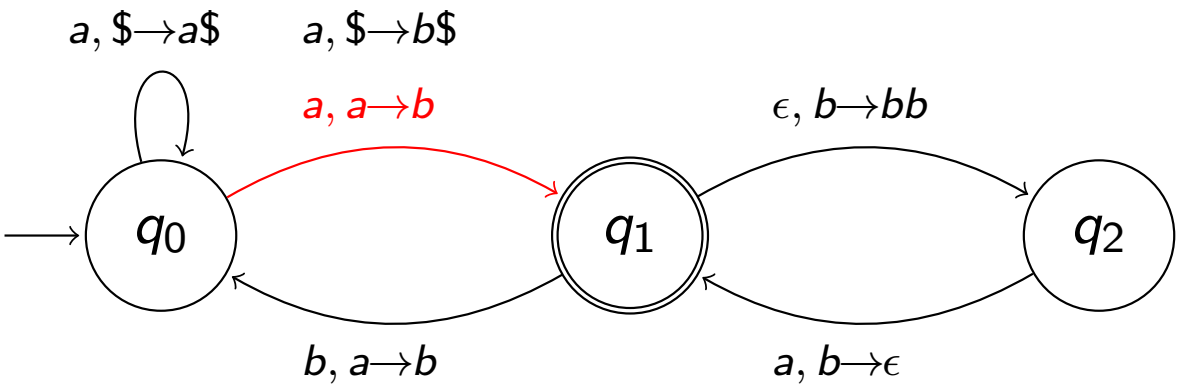
Ejemplo de transición

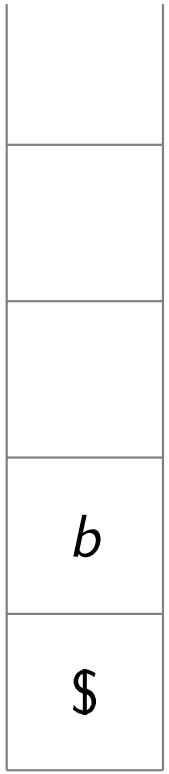




- Transición $a, a \rightarrow b$
- Si estamos en el estado q_0, se lee de la entrada a y en la pila hay a,

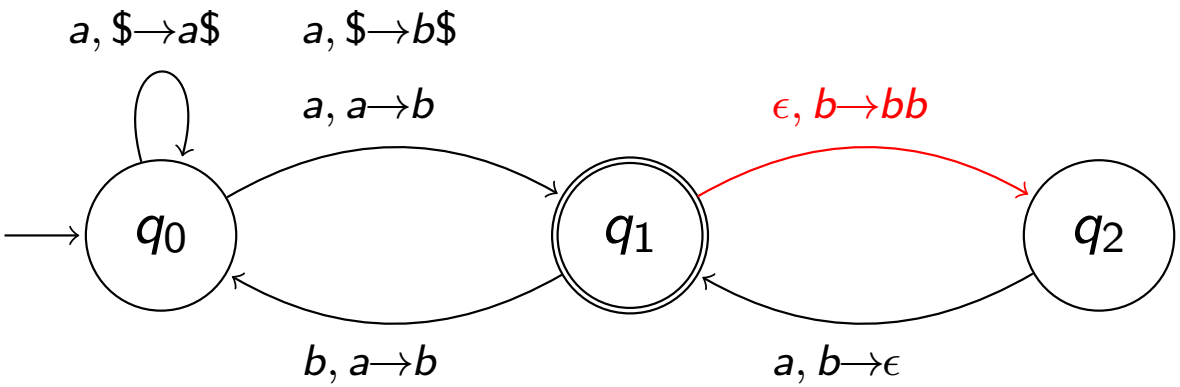
Ejemplo de transición





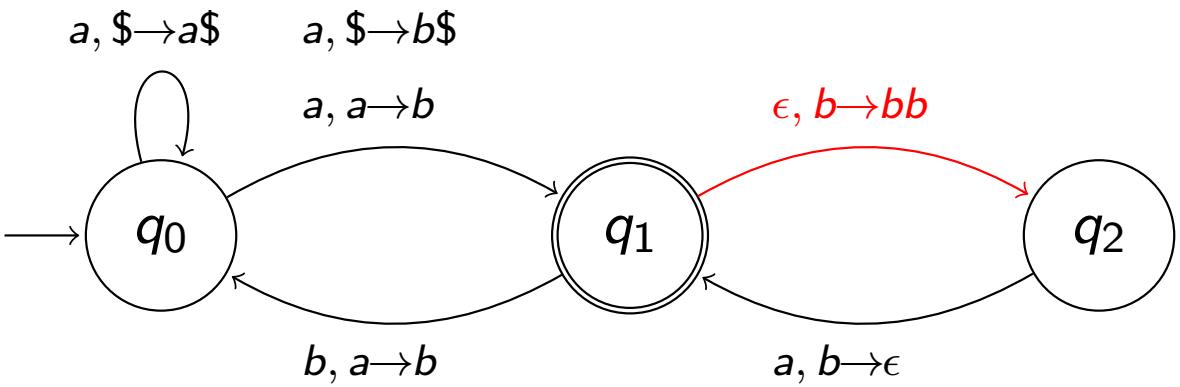
- Transición $a, a \rightarrow b$
- Si estamos en el estado q_0, se lee de la entrada a y en la pila hay a,
- Cambia al estado q_1 y sustituye en la pila a por b: desapila y apila b

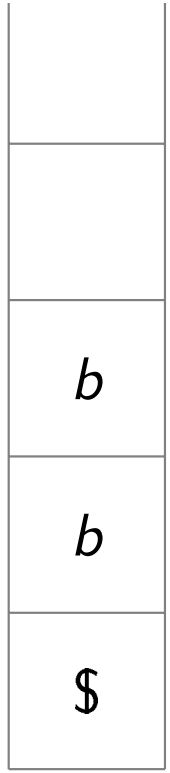
Ejemplo de transición



- Transición $\epsilon, b \rightarrow bb$
- Si estamos en el estado q_1, sin consumir ningún símbolo de la entrada, si en la pila hay b,

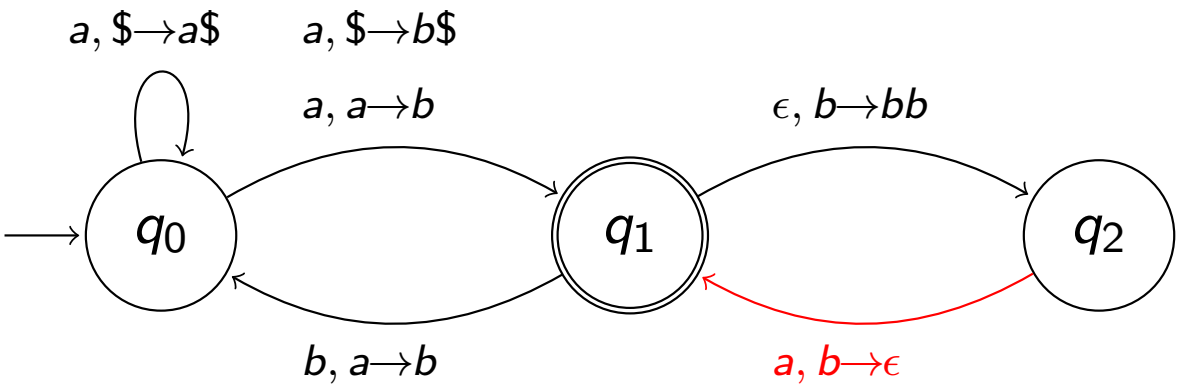
Ejemplo de transición

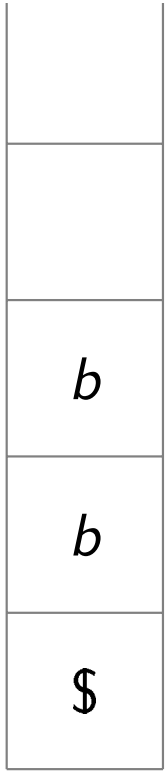




- Transición $\epsilon, b \rightarrow bb$
- Si estamos en el estado q_1, sin consumir ningún símbolo de la entrada, si en la pila hay b,
- Cambia al estado q_2 y sustituye en la pila b por bb: apila b

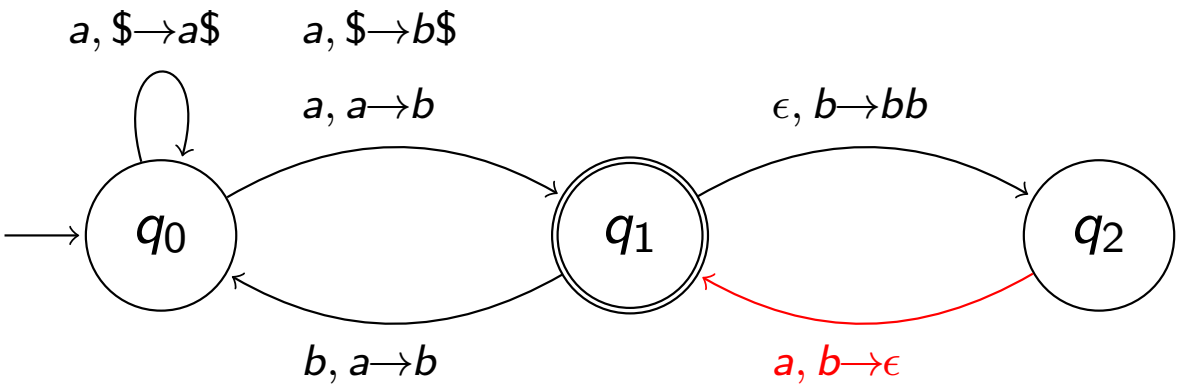
Ejemplo de transición





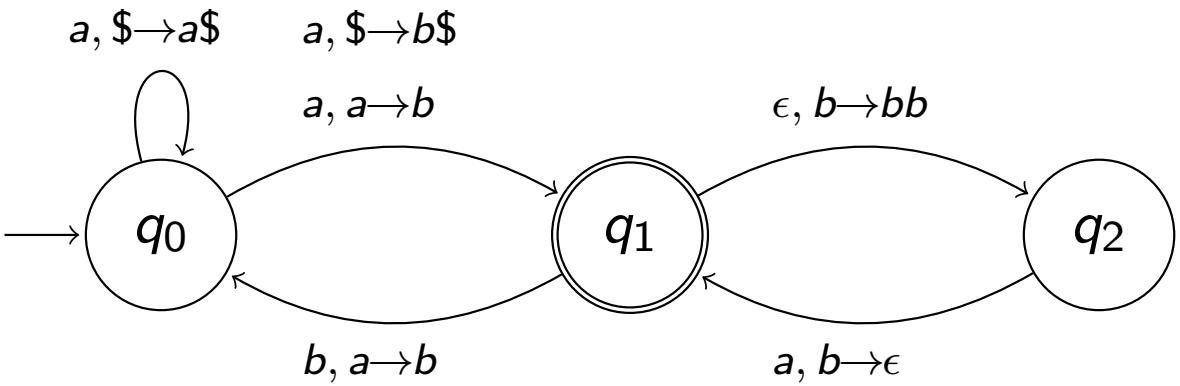
- Transición $a, b \rightarrow \epsilon$
- Si estamos en el estado q_2, se lee de la entrada a, y en la pila hay b,

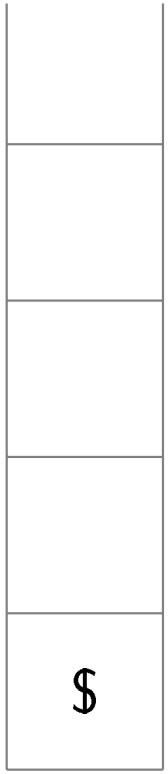
Ejemplo de transición



- Transición $a, b \rightarrow \epsilon$
- Si estamos en el estado q_2, se lee de la entrada a, y en la pila hay b,
- Cambia al estado q_1 y sustituye en la pila b por ϵ: desapila

Computación





- Una entrada es aceptada si al finalizar, alguna de sus computaciones acaba en un estado final.
- Entrada: *aabba* rechazada.
- Entrada: *bbb* rechazada.
- Entrada: *aa* aceptada

Definición formal de autómata de pila

Definición

Un **autómata de pila (AdP)** es $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ tal que

- Q es el conjunto finito de estados
- Σ es el alfabeto de entrada
- Γ es el alfabeto de pila que incluye el símbolo \$
- $q_0 \in Q$ es el estado inicial
- $F \subseteq Q$ es el conjunto de los estados finales o de aceptación.

Definición formal de autómata de pila

Definición

Un **autómata de pila (AdP)** es $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ tal que

- Q es el conjunto finito de estados
- Σ es el alfabeto de entrada
- Γ es el alfabeto de pila que incluye el símbolo \$
- $q_0 \in Q$ es el estado inicial
- $F \subseteq Q$ es el conjunto de los estados finales o de aceptación.
- $\delta : Q \times (\Sigma \cup \epsilon) \times \Gamma \to \mathcal{P}(Q \times \Gamma^*)$ es la función de transición $\delta(q, a, b) = R$ quiere decir que si estoy en el estado q, leo el símbolo a en la entrada y b en la cima puedo ir a cualquiera de los estados q' con $(q', c) \in R$ cambiando la cima a c

Definición formal computación

Definición

Dado un AdP $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, una **computación** de M con entrada $w = a_1 \dots a_m$ con $a_i \in \Sigma \cup \epsilon$ es

$$(q_0, \$)(q_1, y_1) \dots (q_m, y_m)$$

de forma que q_i es un estado e y_i es la cima de la pila en el instante i cumpliendo

$$q_{i+1} \in \delta(q_i, a_{i+1}, y_i)$$

Una **computación aceptadora** de M con entrada w es una computación $(q_0, \$)(q_1, y_1) \dots (q_m, y_m)$ tal que $q_m \in F$. Decimos que la cadena w es **aceptada** por M.

Lenguaje reconocido por un AdP

Definición

Sea $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ un AdP. El lenguaje reconocido por M, $L(M)$, es el conjunto de palabras aceptadas por el autómata

$$L(M) = \{w \in \Sigma^* \mid w \text{ aceptada por } M\}$$

Representación de AdP

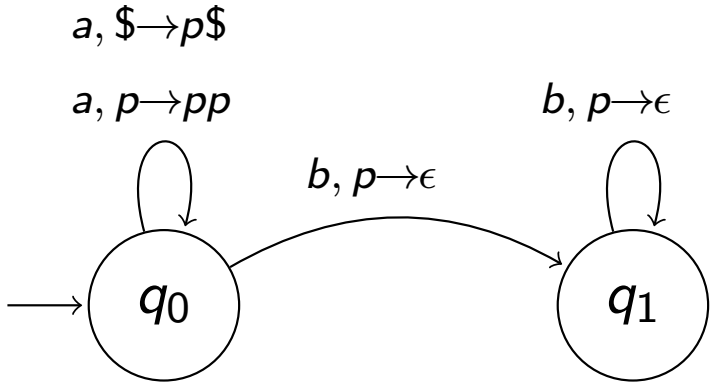
- También utilizaremos la tabla de transiciones para representar un AdP

Entrada	a			b			ϵ		
Pila	a	b	\$	a	b	\$	a	b	\$
q_0	(q_1, b)		$(q_0, a\$), (q_1, b\$)$						
q_1				(q_0, b)				(q_2, bb)	
q_2		(q_1, ϵ)							

Un AdP que reconozca $\{a^n b^n \mid n \geq 1\}$

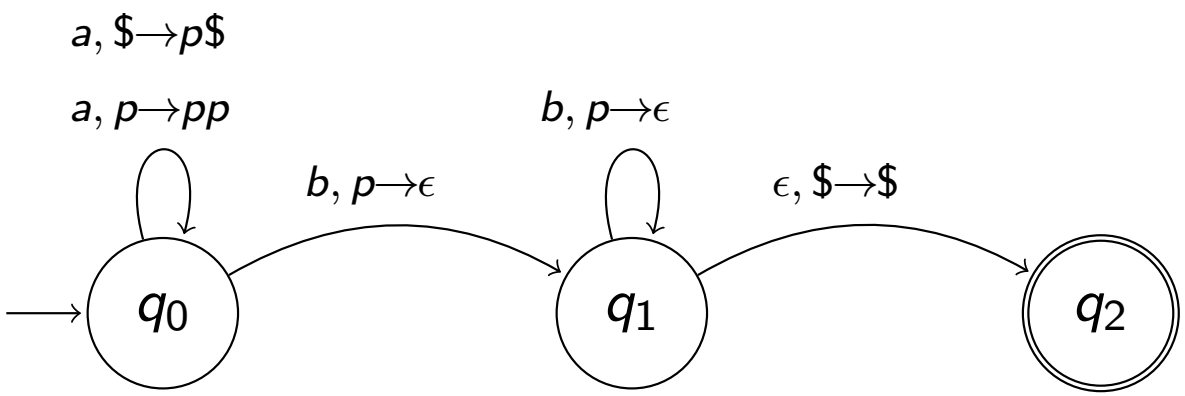
- ¿Cómo diseñamos un autómata que reconozca $L = \{a^n b^n \mid n \geq 1\}$?
- Podemos usar la pila para saber cuántas a hay al inicio de la cadena.
- Por cada a al inicio de la cadena, guardamos en la pila un símbolo p.
- Cuando empezamos a leer las b, desapilamos una p por cada b.
- Si al acabar la entrada, la pila está vacía: aceptamos. Si no, rechazamos.

Un AdP que reconozca $\{a^n b^n \mid n \geq 1\}$



- En q_0, apilamos p por cada a.
- Si aparece una b, desapilamos y pasamos al estado q_1
- En el estado q_1, desapilamos por cada b
- q_1 no puede ser estado final. Si no aceptaríamos, *aaab*, *aaabb*
- ¿Cómo sabemos si la pila está vacía?

Un AdP que reconozca $\{a^n b^n \mid n \geq 1\}$



- En q_0, apilamos p por cada a.
- Si aparece una b, desapilamos y pasamos al estado q_1
- En el estado q_1, desapilamos por cada b
- q_1 no puede ser estado final. Si no aceptaríamos, $aaab$, $aaabb$
- ¿Cómo sabemos si la pila está vacía?
- Añadiendo el estado q_2 con transición $\epsilon, \$ \rightarrow \$$

AdP y Lenguajes independientes del contexto

- ¿Qué relación hay entre los lenguajes incontextuales y los AdP?

Teorema

Un lenguaje es independiente del contexto si y solo si existe un AdP que lo reconoce.

AdP y Lenguajes independientes del contexto

- Dos resultados a demostrar

Lema

Si G es una gramática independiente del contexto, entonces existe un AdP, M, tal que $L(G) = L(M)$

Lema

Si M es un AdP, entonces existe una gramática independiente del contexto, G, tal que $L(M) = L(G)$

Crear AdP que reconoce lenguaje de una gramática

- Partimos de una gramática $G = (N, \Sigma, R, S)$

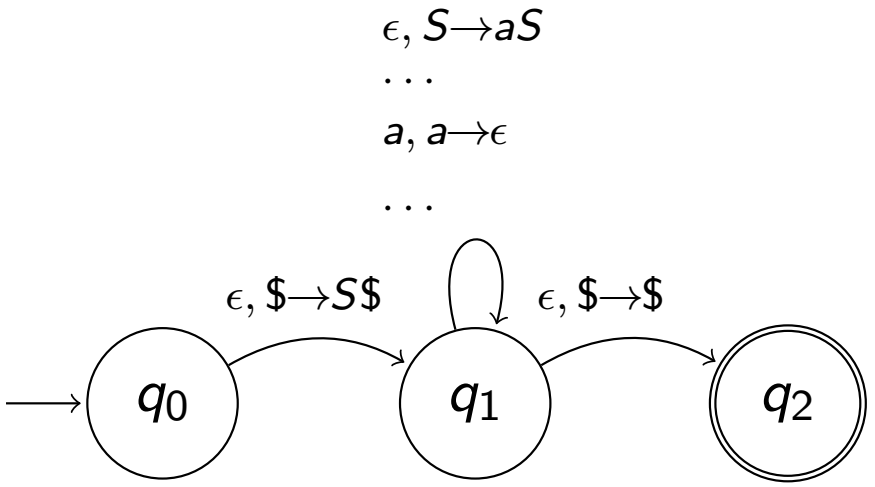
$$S \rightarrow aS|T|Sa$$
$$T \rightarrow bTc|bc$$

- Definimos un AdP cuyo alfabeto de la pila sea la unión de los terminales y no terminales: $\Gamma = N \cup \Sigma \cup \{\$\}$
- Para una entrada w, utilizaremos la pila para hacer la derivación $S \Rightarrow^* w$

Crear AdP que reconoce lenguaje de una gramática

$S \to aS|T|Sa$

$T \to bTc|bc$

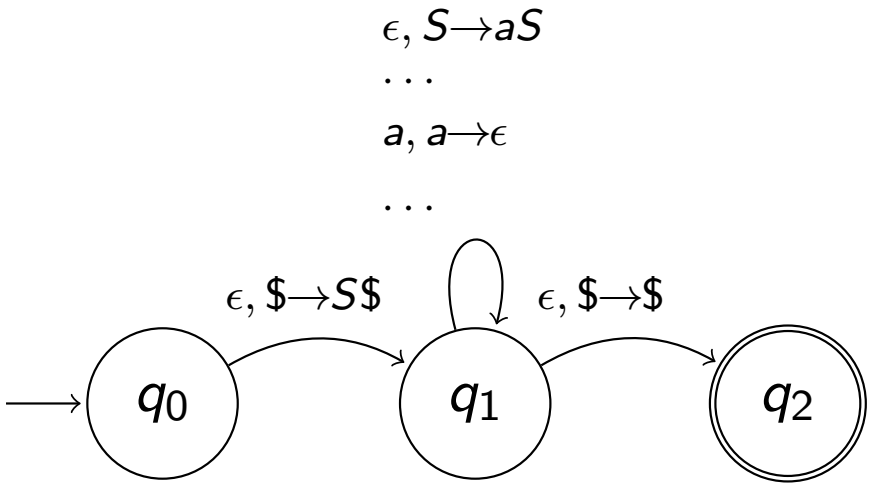


1. Desde un estado inicial del AdP q_0, definimos la transición $\epsilon, \$ \to S\$$ a un estado q_1.
   Esto es, metemos en la pila la variable inicial S
2. Para cada regla $A \to z$, definimos la transición $\epsilon, A \to z$ que lleva del estado q_1 a q_1.
   Esto es, hacemos la sustitución de una derivación sobre la pila.
3. Por cada símbolo de la entrada se define la transición $a, a \to \epsilon$.
   Esto es, quitamos de la pila un símbolo de la entrada.
4. Aceptamos la entrada si al finalizar tenemos la pila vacía: estado q_2

Crear AdP que reconoce lenguaje de una gramática

$S \rightarrow aS|T|Sa$

$T \rightarrow bTc|bc$



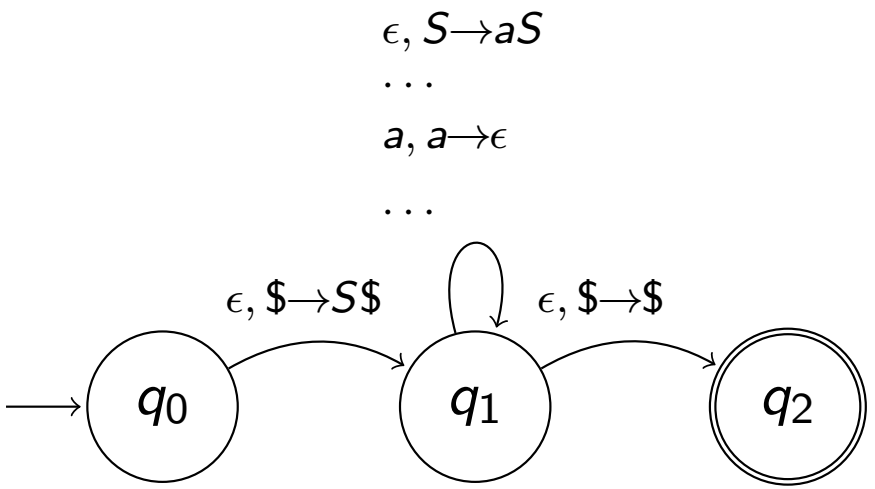
- Por ejemplo, para la entrada *abcaa*, la pila evolucionaría según la derivación

$$S \Rightarrow aS \Rightarrow aSa \Rightarrow aSaa \Rightarrow aTaa \Rightarrow abcaa$$

Crear AdP que reconoce lenguaje de una gramática

$S \rightarrow aS|T|Sa$

$T \rightarrow bTc|bc$



- Por ejemplo, para la entrada *abcaa*, la pila evolucionaría según la derivación

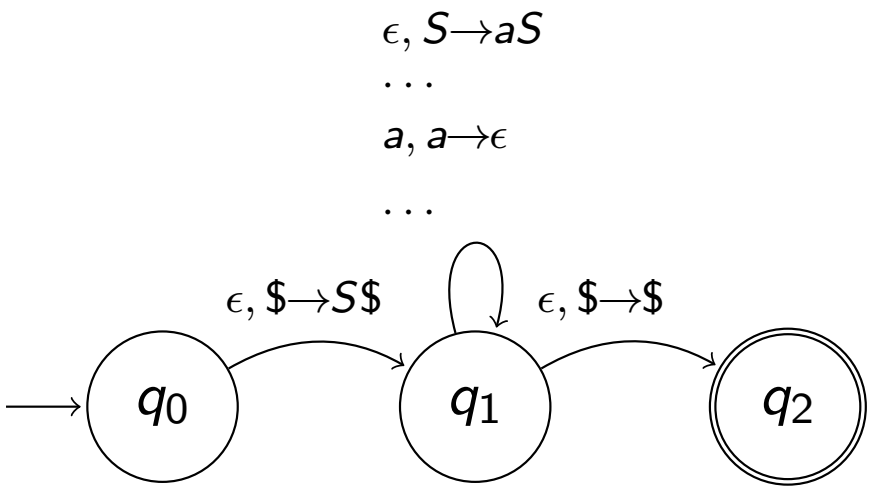
$$S \Rightarrow aS \Rightarrow aSa \Rightarrow aSaa \Rightarrow aTaa \Rightarrow abcaa$$

Transición $\epsilon, \$ \rightarrow S\$$

Crear AdP que reconoce lenguaje de una gramática

$S \rightarrow aS|T|Sa$

$T \rightarrow bTc|bc$



- Por ejemplo, para la entrada *abcaa*, la pila evolucionaría según la derivación

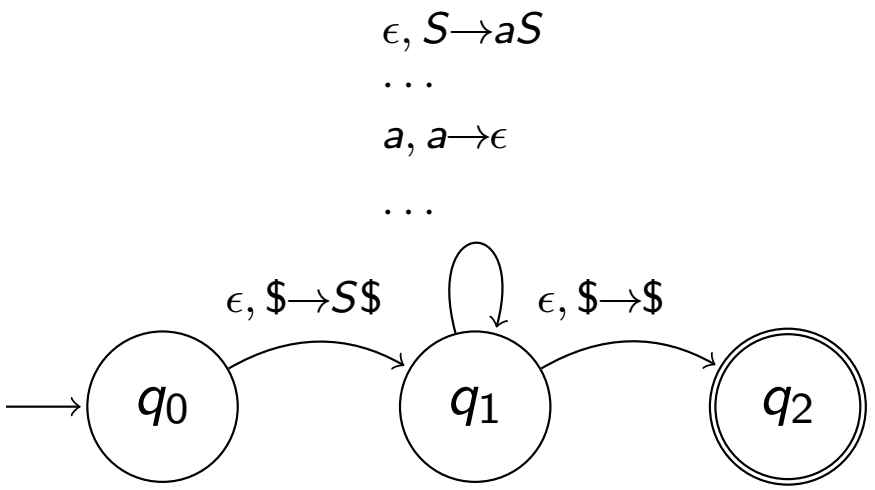
$$S \Rightarrow aS \Rightarrow aSa \Rightarrow aSaa \Rightarrow aTaa \Rightarrow abcaa$$

Transición $\epsilon, S \rightarrow aS$

Crear AdP que reconoce lenguaje de una gramática

$S \rightarrow aS|T|Sa$

$T \rightarrow bTc|bc$



- Por ejemplo, para la entrada *abcaa*, la pila evolucionaría según la derivación

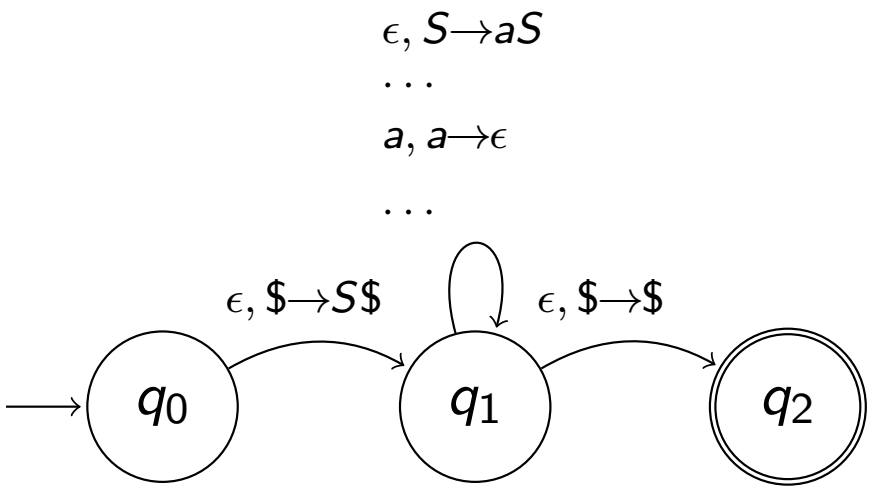
$$S \Rightarrow aS \Rightarrow aSa \Rightarrow aSaa \Rightarrow aTaa \Rightarrow abcaa$$

Transición $a, a \rightarrow \epsilon$

Crear AdP que reconoce lenguaje de una gramática

$S \rightarrow aS|T|Sa$

$T \rightarrow bTc|bc$



- Por ejemplo, para la entrada *abcaa*, la pila evolucionaría según la derivación

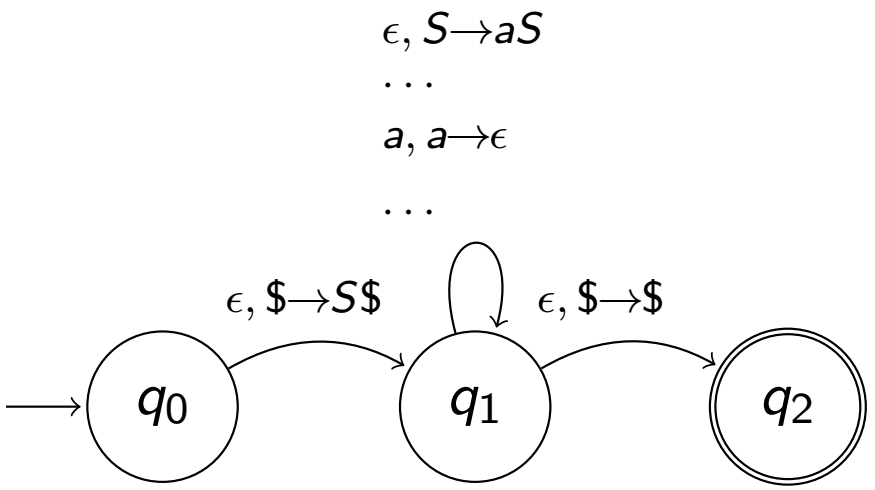
$$S \Rightarrow aS \Rightarrow aSa \Rightarrow aSaa \Rightarrow aTaa \Rightarrow abcaa$$

Transición $\epsilon, S \rightarrow Sa$

Crear AdP que reconoce lenguaje de una gramática

$S \rightarrow aS|T|Sa$

$T \rightarrow bTc|bc$



- Por ejemplo, para la entrada *abcaa*, la pila evolucionaría según la derivación

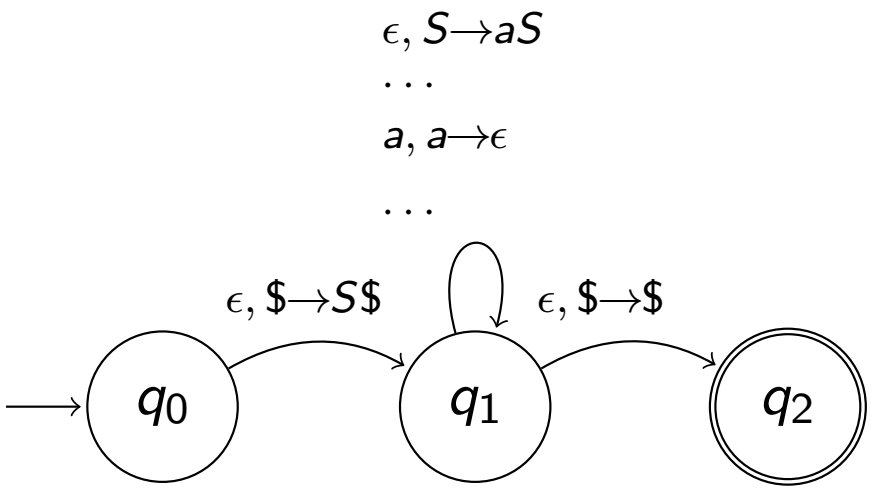
$$S \Rightarrow aS \Rightarrow aSa \Rightarrow aSaa \Rightarrow aTaa \Rightarrow abcaa$$

Transición $\epsilon, S \rightarrow Sa$

Crear AdP que reconoce lenguaje de una gramática

$S \rightarrow aS|T|Sa$

$T \rightarrow bTc|bc$



- Por ejemplo, para la entrada *abcaa*, la pila evolucionaría según la derivación

$$S \Rightarrow aS \Rightarrow aSa \Rightarrow aSaa \Rightarrow aTaa \Rightarrow abcaa$$

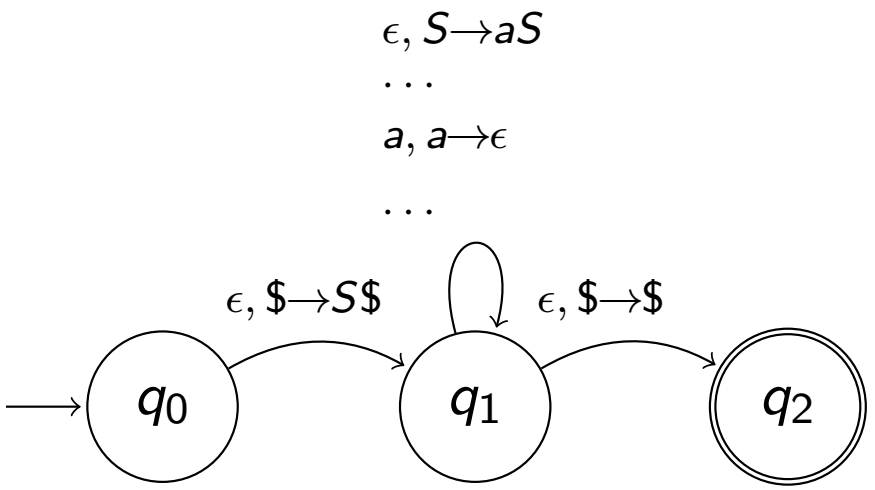
				T	a	a	\$

Transición $\epsilon, S \rightarrow T$

Crear AdP que reconoce lenguaje de una gramática

$S \rightarrow aS|T|Sa$

$T \rightarrow bTc|bc$



- Por ejemplo, para la entrada *abcaa*, la pila evolucionaría según la derivación

$$S \Rightarrow aS \Rightarrow aSa \Rightarrow aSaa \Rightarrow aTaa \Rightarrow abcaa$$

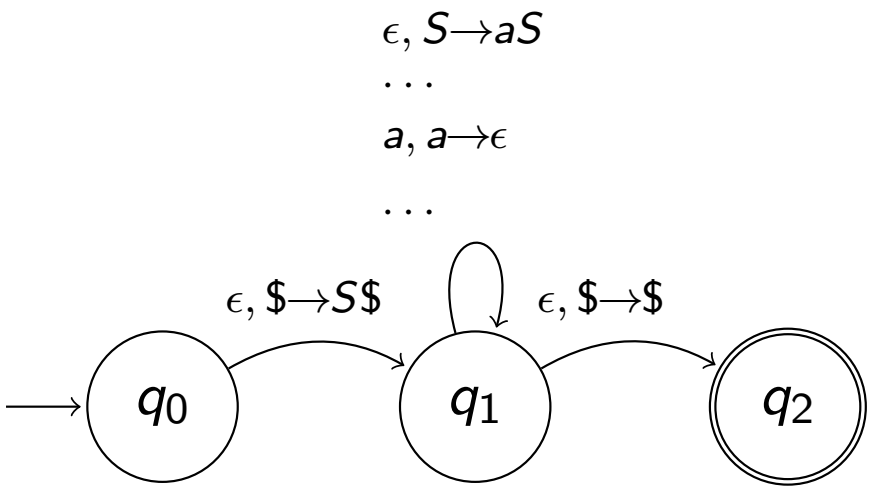
			b	c	a	a	\$

Transición $\epsilon, T \rightarrow bc$

Crear AdP que reconoce lenguaje de una gramática

$S \rightarrow aS|T|Sa$

$T \rightarrow bTc|bc$



- Por ejemplo, para la entrada *abcaa*, la pila evolucionaría según la derivación

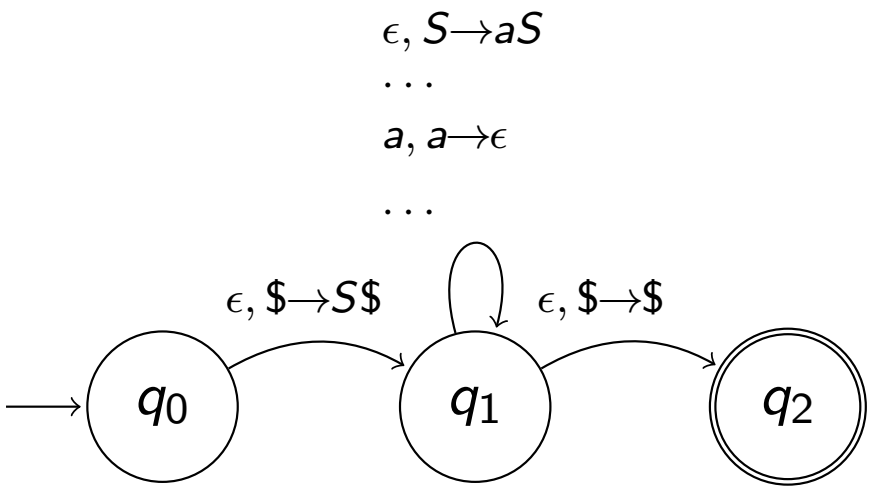
$$S \Rightarrow aS \Rightarrow aSa \Rightarrow aSaa \Rightarrow aTaa \Rightarrow abcaa$$

Transición $b, b \rightarrow \epsilon$

Crear AdP que reconoce lenguaje de una gramática

$S \rightarrow aS|T|Sa$

$T \rightarrow bTc|bc$



- Por ejemplo, para la entrada *abcaa*, la pila evolucionaría según la derivación

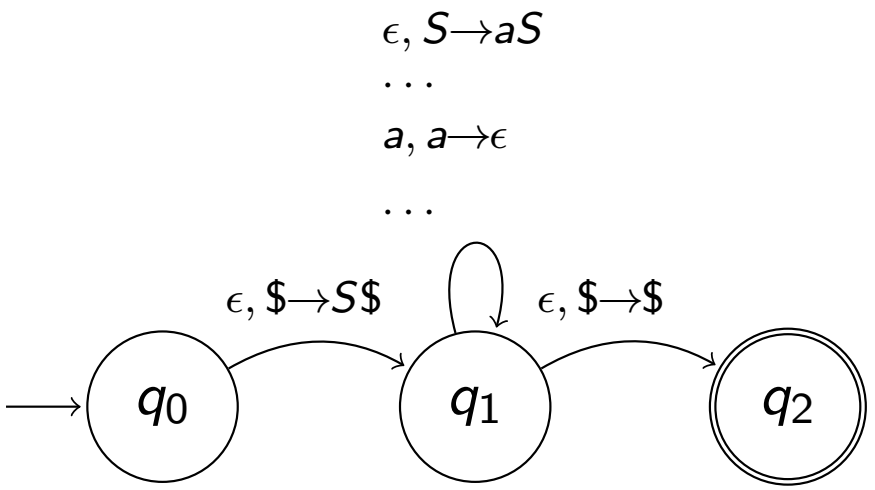
$$S \Rightarrow aS \Rightarrow aSa \Rightarrow aSaa \Rightarrow aTaa \Rightarrow abcaa$$

Transición $c, c \rightarrow \epsilon$

Crear AdP que reconoce lenguaje de una gramática

$S \rightarrow aS|T|Sa$

$T \rightarrow bTc|bc$



- Por ejemplo, para la entrada *abcaa*, la pila evolucionaría según la derivación

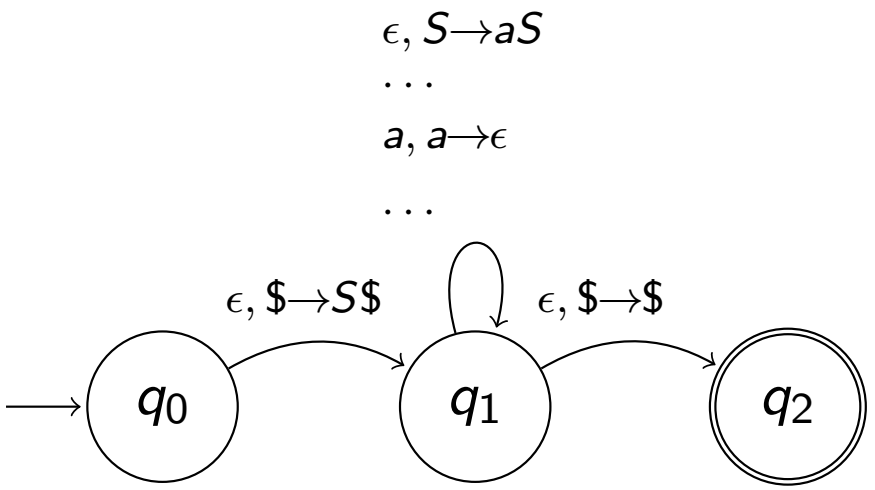
$$S \Rightarrow aS \Rightarrow aSa \Rightarrow aSaa \Rightarrow aTaa \Rightarrow abcaa$$

Transición $a, a \rightarrow \epsilon$

Crear AdP que reconoce lenguaje de una gramática

$S \rightarrow aS|T|Sa$

$T \rightarrow bTc|bc$



- Por ejemplo, para la entrada *abcaa*, la pila evolucionaría según la derivación

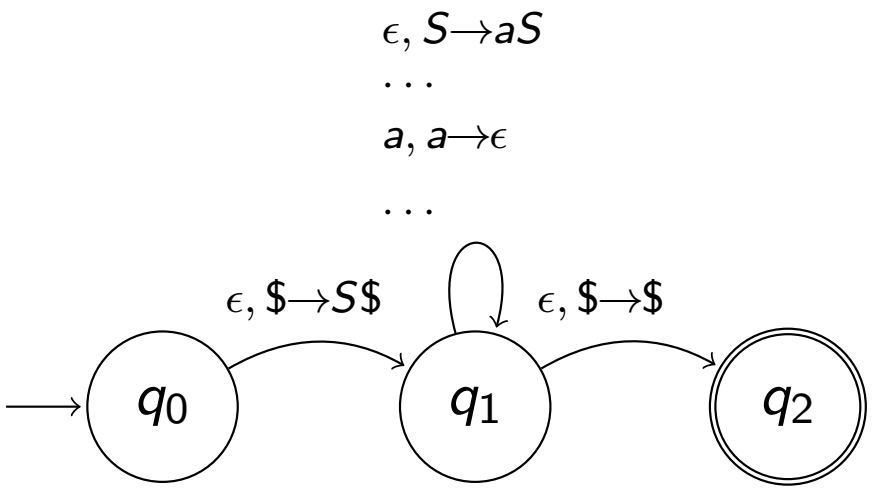
$$S \Rightarrow aS \Rightarrow aSa \Rightarrow aSaa \Rightarrow aTaa \Rightarrow abcaa$$

Transición $a, a \rightarrow \epsilon$

Crear AdP que reconoce lenguaje de una gramática

$S \rightarrow aS|T|Sa$

$T \rightarrow bTc|bc$



- Por ejemplo, para la entrada *abcaa*, la pila evolucionaría según la derivación

$$S \Rightarrow aS \Rightarrow aSa \Rightarrow aSaa \Rightarrow aTaa \Rightarrow abcaa$$

Transición $\epsilon, \$ \rightarrow \$$ aceptada

Dar la gramática de un AdP

- Definimos una gramática que tenga un noterminal $A_{p,q}$ para cada par de estados del autómata $p, q \in Q$

Dar la gramática de un AdP

- Definimos una gramática que tenga un noterminal $A_{p,q}$ para cada par de estados del autómata $p, q \in Q$
- $A_{p,q}$ genera todas las entradas que llevan del estado p con pila \$ al estado q con pila \$

Dar la gramática de un AdP

- Definimos una gramática que tenga un noterminal $A_{p,q}$ para cada par de estados del autómata $p, q \in Q$
- $A_{p,q}$ genera todas las entradas que llevan del estado p con pila \$ al estado q con pila \$
- Se añaden las reglas:

Dar la gramática de un AdP

- Definimos una gramática que tenga un noterminal $A_{p,q}$ para cada par de estados del autómata $p, q \in Q$
- $A_{p,q}$ genera todas las entradas que llevan del estado p con pila \$ al estado q con pila \$
- Se añaden las reglas:
  - Si los finales son $F = \{q_1, \ldots, q_r\}$,

$$S \to A_{q_0,q_1} \mid \ldots A_{q_0,q_r}$$

Dar la gramática de un AdP

- Definimos una gramática que tenga un noterminal $A_{p,q}$ para cada par de estados del autómata $p, q \in Q$
- $A_{p,q}$ genera todas las entradas que llevan del estado p con pila \$ al estado q con pila \$
- Se añaden las reglas:
  - Si los finales son $F = \{q_1, \ldots, q_r\}$,

    $$S \to A_{q_0,q_1} \mid \ldots A_{q_0,q_r}$$

  - Se añade la regla

    $$A_{p,q} \to aA_{r,s}b$$

    cuando la entrada a lleva de p a r apilando u y la entrada b lleva de s a q desapilando u

Dar la gramática de un AdP

- Definimos una gramática que tenga un noterminal $A_{p,q}$ para cada par de estados del autómata $p, q \in Q$
- $A_{p,q}$ genera todas las entradas que llevan del estado p con pila \$ al estado q con pila \$
- Se añaden las reglas:
  - Si los finales son $F = \{q_1, \ldots, q_r\}$,

    $$S \to A_{q_0,q_1} \mid \ldots A_{q_0,q_r}$$

  - Se añade la regla

    $$A_{p,q} \to aA_{r,s}b$$

    cuando la entrada a lleva de p a r apilando u y la entrada b lleva de s a q desapilando u
  - Para todos los estados p, q, r, se añade la regla

    $$A_{p,q} \to A_{p,r}A_{r,q}$$

Dar la gramática de un AdP

- Definimos una gramática que tenga un noterminal $A_{p,q}$ para cada par de estados del autómata $p, q \in Q$
- $A_{p,q}$ genera todas las entradas que llevan del estado p con pila \$ al estado q con pila \$
- Se añaden las reglas:
  - Si los finales son $F = \{q_1, \ldots, q_r\}$,

    $$S \to A_{q_0,q_1} \mid \ldots A_{q_0,q_r}$$

  - Se añade la regla

    $$A_{p,q} \to aA_{r,s}b$$

    cuando la entrada a lleva de p a r apilando u y la entrada b lleva de s a q desapilando u
  - Para todos los estados p, q, r, se añade la regla

    $$A_{p,q} \to A_{p,r}A_{r,q}$$

  - para cualquier estado p se añade la regla

    $$A_{p,p} \to \epsilon$$

Dar la gramática de un AdP

- Estado final q_1

En. Pila.	a			b			ϵ
	a	b	\$	a	b	\$	\$
q_0	(q_0, aa)	(q_0, ϵ)	$(q_0, a\$)$	(q_0, ϵ)	(q_0, bb)	$(q_0, b\$)$	$(q_1, \$)$

$S \rightarrow$ A_{q_0,q_1}

Dar la gramática de un AdP

- Estado final q_1

En. Pila.	a			b			ϵ
	a	b	\$	a	b	\$	\$
q_0	(q_0, aa)	(q_0, ϵ)	$(q_0, a\$)$	(q_0, ϵ)	(q_0, bb)	$(q_0, b\$)$	$(q_1, \$)$

$$
\begin{aligned}
S &\rightarrow A_{q_0,q_1} \\
A_{q_0,q_0} &\rightarrow aA_{q_0,q_0}b \\
A_{q_0,q_0} &\rightarrow bA_{q_0,q_0}a
\end{aligned}
$$

Dar la gramática de un AdP

- Estado final q_1

En. Pila.	a			b			ϵ
	a	b	\$	a	b	\$	\$
q_0	(q_0, aa)	(q_0, ϵ)	$(q_0, a\$)$	(q_0, ϵ)	(q_0, bb)	$(q_0, b\$)$	$(q_1, \$)$

$$
\begin{aligned}
S &\rightarrow A_{q_0,q_1} \\
A_{q_0,q_0} &\rightarrow aA_{q_0,q_0}b \\
A_{q_0,q_0} &\rightarrow bA_{q_0,q_0}a \\
A_{q_0,q_0} &\rightarrow A_{q_0,q_0}A_{q_0,q_0} \mid A_{q_0,q_1}A_{q_1,q_0} \\
A_{q_0,q_1} &\rightarrow A_{q_0,q_0}A_{q_0,q_1} \mid A_{q_0,q_1}A_{q_1,q_1} \\
A_{q_1,q_0} &\rightarrow A_{q_1,q_0}A_{q_0,q_0} \mid A_{q_1,q_1}A_{q_1,q_0} \\
A_{q_1,q_1} &\rightarrow A_{q_1,q_0}A_{q_0,q_1} \mid A_{q_1,q_1}A_{q_1,q_1}
\end{aligned}
$$

Dar la gramática de un AdP

- Estado final q_1

En. Pila.	a			b			ϵ
	a	b	\$	a	b	\$	\$
q_0	(q_0, aa)	(q_0, ϵ)	$(q_0, a\$)$	(q_0, ϵ)	(q_0, bb)	$(q_0, b\$)$	$(q_1, \$)$

$$
\begin{aligned}
S &\rightarrow A_{q_0,q_1} \\
A_{q_0,q_0} &\rightarrow aA_{q_0,q_0}b \\
A_{q_0,q_0} &\rightarrow bA_{q_0,q_0}a \\
A_{q_0,q_0} &\rightarrow A_{q_0,q_0}A_{q_0,q_0} \mid A_{q_0,q_1}A_{q_1,q_0} \\
A_{q_0,q_1} &\rightarrow A_{q_0,q_0}A_{q_0,q_1} \mid A_{q_0,q_1}A_{q_1,q_1} \\
A_{q_1,q_0} &\rightarrow A_{q_1,q_0}A_{q_0,q_0} \mid A_{q_1,q_1}A_{q_1,q_0} \\
A_{q_1,q_1} &\rightarrow A_{q_1,q_0}A_{q_0,q_1} \mid A_{q_1,q_1}A_{q_1,q_1} \\
A_{q_0,q_0} &\rightarrow \epsilon \\
A_{q_1,q_1} &\rightarrow \epsilon
\end{aligned}
$$

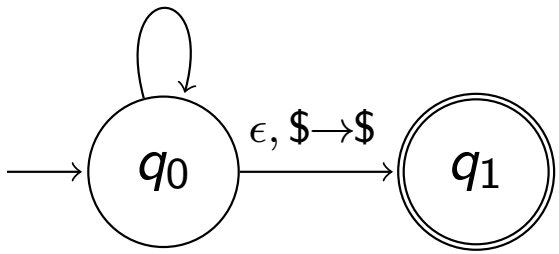
Conclusión

- Si tenemos que demostrar que un lenguaje es incontextual, podemos definir una gramática o un AdP
- Naturaleza del problema recursiva: usar gramaticas.
  $L = \{w = w^R \mid w \in \{a, b\}^*\}$ es incontextual

$$S \rightarrow aSa|bSb|a|b|\epsilon$$

- Naturaleza del problema iterativa: usar AdP
  $L = \{w \in \{a, b\}^* \mid |w|_a = |w|_b\}$ es incontextual

$$\begin{array}{ll} a, \$ \rightarrow a\$ & b, \$ \rightarrow b\$ \\ a, a \rightarrow aa & a, b \rightarrow \epsilon \\ b, b \rightarrow bb & b, a \rightarrow \epsilon \end{array}$$



Autómatas de Pila deterministas

- Recordar que para los lenguajes regulares existía una equivalencia entre autómatas finitos deterministas y no deterministas.
- Los AdP son no deterministas. ¿Existe siempre algún AdP determinista equivalente?

Autómatas de Pila deterministas

- Recordar que para los lenguajes regulares existía una equivalencia entre autómatas finitos deterministas y no deterministas.
- Los AdP son no deterministas. ¿Existe siempre algún AdP determinista equivalente?
- No siempre. Es decir, los lenguajes reconocidos por los autómatas de pilas deterministas son un subconjunto propio de los lenguajes reconocidos por los autómatas de pila.
- $L = \{ww^R \mid w \in \{a, b\}^*\}$, no existe un autómata de pila determinista que lo reconozca.

Lenguajes no incontextuales

- ¿Son todos los lenguajes independientes del contexo?

Lenguajes no incontextuales

- ¿Son todos los lenguajes independientes del contexo?
- Existen lenguajes que no son LIC.
- $L = \{a^n b^n c^n \mid n \geq 0\}$
- ¿Cómo podemos saber si un lenguaje no es LIC?

Lenguajes no incontextuales

- ¿Son todos los lenguajes independientes del contexo?
- Existen lenguajes que no son LIC.
- $L = \{a^n b^n c^n \mid n \geq 0\}$
- ¿Cómo podemos saber si un lenguaje no es LIC?
- Lema de bombeo para LICs

Lema de bombeo para LICs

- Se trata de un lema que nos permite demostrar que algunos lenguajes no son incontextuales.

Lema de bombeo para LICs

- Se trata de un lema que nos permite demostrar que algunos lenguajes no son incontextuales.
- Está basado en los bucles de las gramáticas, por ejemplo $S \to aSb$ o bien $S \to Sb$

Lema de bombeo para LICs

- Se trata de un lema que nos permite demostrar que algunos lenguajes no son incontextuales.
- Está basado en los bucles de las gramáticas, por ejemplo $S \to aSb$ o bien $S \to Sb$
- Si tienes una derivación de la palabra

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaaabbbb$$

puedes repetir el bucle $S \to aSb$ y generar palabras más largas usando $S \Rightarrow^* a^n S b^n$

Lema de bombeo para LICs

- Se trata de un lema que nos permite demostrar que algunos lenguajes no son incontextuales.
- Está basado en los bucles de las gramáticas, por ejemplo $S \to aSb$ o bien $S \to Sb$
- Si tienes una derivación de la palabra

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaaabbbb$$

puedes repetir el bucle $S \to aSb$ y generar palabras más largas usando $S \Rightarrow^* a^n S b^n$

- Lo mismo si tienes reglas $A \to aBa$ y $B \to bA$

$$S \Rightarrow AB \Rightarrow aBaB \Rightarrow abAaB$$

puedes repetir el bucle $A \Rightarrow^* abAa$ y generar palabras más largas usando $A \Rightarrow^* (ab)^n A a^n$

Lema de bombeo para LICs

Lema de bombeo

Si L es un lenguaje independiente de contexto infinito, existe un número N tal que cualquier $w \in L$ con $|w| \geq N$, se puede dividir en $w = uvxyz$ cumpliendo

- $uv^i xy^i z \in L$, para todo $i \geq 0$
- $|vy| \geq 1$
- $|vxy| \leq N$

Otra forma del lema de bombeo para LICs

Dado un lenguaje infinito L,

- si $\forall N \; \exists w$ con $w \in L$, $|w| \geq N$
- tal que $\forall u, v, x, y, z$ con $w = uvxyz$, $|vy| \geq 1$ y $|vxy| \leq N$
- $\exists i$ con $uv^i xy^i z \notin L$

entonces L no es incontextual

$A = \{a^n b^n c^n\}$ no es LIC

$A = \{a^n b^n c^n\}$ no es LIC

- Para todo N existe una palabra en A, $|w| \geq N$
  - $w = a^N b^N c^N$ $|w| = 3N \geq N$

$A = \{a^n b^n c^n\}$ no es LIC

- Para todo N existe una palabra en A, $|w| \geq N$
  - $w = a^N b^N c^N$ $|w| = 3N \geq N$
- tal que para cualquier partición de w, u, v, x, y, z con $w = uvxyz$, $|vy| \geq 1$ y $|vxy| \leq N$,
  - la partición puede ser

$$vxy = a^r b^s \qquad v = a^t, y = a^l b^s, t + l + s \geq 1 \qquad (1)$$

$$vxy = a^r b^s \qquad v = a^t b^l, y = b^j, t + l + j \geq 1 \qquad (2)$$

$$vxy = b^r c^s \qquad v = b^t, y = b^l c^s, t + l + s \geq 1 \qquad (3)$$

$$vxy = b^r c^s \qquad v = b^t c^l, y = c^j, t + l + j \geq 1 \qquad (4)$$

$A = \{a^n b^n c^n\}$ no es LIC

- Para todo N existe una palabra en A, $|w| \geq N$
  - $w = a^N b^N c^N$ $|w| = 3N \geq N$
- tal que para cualquier partición de w, u, v, x, y, z con $w = uvxyz$, $|vy| \geq 1$ y $|vxy| \leq N$,
  - la partición puede ser

$$vxy = a^r b^s \qquad v = a^t, y = a^l b^s, t + l + s \geq 1 \qquad (1)$$

$$vxy = a^r b^s \qquad v = a^t b^l, y = b^j, t + l + j \geq 1 \qquad (2)$$

$$vxy = b^r c^s \qquad v = b^t, y = b^l c^s, t + l + s \geq 1 \qquad (3)$$

$$vxy = b^r c^s \qquad v = b^t c^l, y = c^j, t + l + j \geq 1 \qquad (4)$$

- $\exists i$ con $uv^i xy^i z \notin A$
  - caso (1): para $i = 2$,

$$uv^2 xy^2 z = a^{N-r} a^{2t} a^{r-t-l} a^l b^s a^l b^s b^{N-s} c^N = a^{N+t} b^s a^l b^N c^N$$

si $s \geq 1$, hay b entre a; y si $s = 0$, hay más a que b (o c).

$A = \{a^n b^n c^n\}$ no es LIC

- Para todo N existe una palabra en A, $|w| \geq N$
  - $w = a^N b^N c^N$ $|w| = 3N \geq N$
- tal que para cualquier partición de w, u, v, x, y, z con $w = uvxyz$, $|vy| \geq 1$ y $|vxy| \leq N$,
  - la partición puede ser

$$vxy = a^r b^s \qquad v = a^t, y = a^l b^s, t + l + s \geq 1 \qquad (1)$$

$$vxy = a^r b^s \qquad v = a^t b^l, y = b^j, t + l + j \geq 1 \qquad (2)$$

$$vxy = b^r c^s \qquad v = b^t, y = b^l c^s, t + l + s \geq 1 \qquad (3)$$

$$vxy = b^r c^s \qquad v = b^t c^l, y = c^j, t + l + j \geq 1 \qquad (4)$$

- $\exists i$ con $uv^i xy^i z \notin A$
  - caso (1): para $i = 2$,

$$uv^2 xy^2 z = a^{N-r} a^{2t} a^{r-t-l} a^l b^s a^l b^s b^{N-s} c^N = a^{N+t} b^s a^l b^N c^N$$

    si $s \geq 1$, hay b entre a; y si $s = 0$, hay más a que b (o c).
  - Hay que mirar los otros tres casos

$A = \{a^n b^n c^n\}$ no es LIC

- Para todo N existe una palabra en A, $|w| \geq N$
  - $w = a^N b^N c^N$ $|w| = 3N \geq N$
- tal que para cualquier partición de w, u, v, x, y, z con $w = uvxyz$, $|vy| \geq 1$ y $|vxy| \leq N$,
  - la partición puede ser

$$vxy = a^r b^s \qquad v = a^t, y = a^l b^s, t + l + s \geq 1 \qquad (1)$$

$$vxy = a^r b^s \qquad v = a^t b^l, y = b^j, t + l + j \geq 1 \qquad (2)$$

$$vxy = b^r c^s \qquad v = b^t, y = b^l c^s, t + l + s \geq 1 \qquad (3)$$

$$vxy = b^r c^s \qquad v = b^t c^l, y = c^j, t + l + j \geq 1 \qquad (4)$$

- $\exists i$ con $uv^i xy^i z \notin A$
  - caso (1): para $i = 2$,

$$uv^2 xy^2 z = a^{N-r} a^{2t} a^{r-t-l} a^l b^s a^l b^s b^{N-s} c^N = a^{N+t} b^s a^l b^N c^N$$

    si $s \geq 1$, hay b entre a; y si $s = 0$, hay más a que b (o c).
  - Hay que mirar los otros tres casos
- luego A no es incontextual.

$A = \{ww \mid w \in \{0,1\}^*\}$ no es LIC

$A = \{ww \mid w \in \{0,1\}^*\}$ no es LIC

- Para todo N existe una palabra en A, $|w| \geq N$
  - $w = 0^N 1^N 0^N 1^N$, $|w| = 4N \geq N$

$A = \{ww \mid w \in \{0,1\}^*\}$ no es LIC

- Para todo N existe una palabra en A, $|w| \geq N$
  - $w = 0^N 1^N 0^N 1^N$, $|w| = 4N \geq N$
- tal que para cualquier partición de w, u, v, x, y, z con $w = uvxyz$, $|vy| \geq 1$ y $|vxy| \leq N$,
  - Caso 1: vxy cae en el primer $0^N 1^N$ de w
  - Caso 2: vxy cae en el segundo $0^N 1^N$ de w
  - Caso 3: vxy está en el medio, $vxy = 1^k 0^l$

$A = \{ww \mid w \in \{0,1\}^*\}$ no es LIC

- Para todo N existe una palabra en A, $|w| \geq N$
  - $w = 0^N 1^N 0^N 1^N$, $|w| = 4N \geq N$
- tal que para cualquier partición de w, u, v, x, y, z con $w = uvxyz$, $|vy| \geq 1$ y $|vxy| \leq N$,
  - Caso 1: vxy cae en el primer $0^N 1^N$ de w
  - Caso 2: vxy cae en el segundo $0^N 1^N$ de w
  - Caso 3: vxy está en el medio, $vxy = 1^k 0^l$
- $\exists i$ con $uv^i xy^i z \notin A$
  - caso 1: para $i = 2$, la primera parte de uv^2xy^2z empieza por 0 y la segunda por 1
  - caso 2: para $i = 2$, la primera parte de uv^2xy^2z acaba por 0 y la segunda por 1
  - caso 3: para $i = 0$, $uxz = 0^N 1^r 0^s 1^N$, con $r \neq N$ y $s \neq N$

$A = \{ww \mid w \in \{0,1\}^*\}$ no es LIC

- Para todo N existe una palabra en A, $|w| \geq N$
  - $w = 0^N 1^N 0^N 1^N$, $|w| = 4N \geq N$
- tal que para cualquier partición de w, u, v, x, y, z con $w = uvxyz$, $|vy| \geq 1$ y $|vxy| \leq N$,
  - Caso 1: vxy cae en el primer $0^N 1^N$ de w
  - Caso 2: vxy cae en el segundo $0^N 1^N$ de w
  - Caso 3: vxy está en el medio, $vxy = 1^k 0^l$
- $\exists i$ con $uv^i xy^i z \notin A$
  - caso 1: para $i = 2$, la primera parte de uv^2xy^2z empieza por 0 y la segunda por 1
  - caso 2: para $i = 2$, la primera parte de uv^2xy^2z acaba por 0 y la segunda por 1
  - caso 3: para $i = 0$, $uxz = 0^N 1^r 0^s 1^N$, con $r \neq N$ y $s \neq N$
- luego A no es incontextual.

Lenguajes no incontextuales

- También se pueden usar las propiedades de clausura para demostrar que un lenguaje no es incontextual.
- Si A es LIC y B regular, entonces $A \cap B$ es LIC.

Lenguajes no incontextuales

- También se pueden usar las propiedades de clausura para demostrar que un lenguaje no es incontextual.
- Si A es LIC y B regular, entonces $A \cap B$ es LIC.
- ¿$A = \{w \mid |w|_a = |w|_b = |w|_c\}$ no es LIC?

Lenguajes no incontextuales

- También se pueden usar las propiedades de clausura para demostrar que un lenguaje no es incontextual.
- Si A es LIC y B regular, entonces $A \cap B$ es LIC.
- ¿$A = \{w \mid |w|_a = |w|_b = |w|_c\}$ no es LIC?
- $A \cap a^*b^*c^* = \{a^n b^n c^n\}$.

Lenguajes no incontextuales

- También se pueden usar las propiedades de clausura para demostrar que un lenguaje no es incontextual.
- Si A es LIC y B regular, entonces $A \cap B$ es LIC.
- ¿$A = \{w \mid |w|_a = |w|_b = |w|_c\}$ no es LIC?
- $A \cap a^*b^*c^* = \{a^n b^n c^n\}$.
- Como $\{a^n b^n c^n\}$ no es LIC, y $a^*b^*c^*$ es regular, A no es LIC

La intersección de LICs

- En la lección anterior comentamos que la intersección de LICs no es necesariamente LIC.
- Veamos un ejemplo.

La intersección de LICs

- En la lección anterior comentamos que la intersección de LICs no es necesariamente LIC.
- Veamos un ejemplo.
- $L = \{a^n b^n \mid n \geq 0\}$ es LIC

La intersección de LICs

- En la lección anterior comentamos que la intersección de LICs no es necesariamente LIC.
- Veamos un ejemplo.
- $L = \{a^n b^n \mid n \geq 0\}$ es LIC
- $A = L \cdot c^*$ también es LIC (concatenación de LICs)

La intersección de LICs

- En la lección anterior comentamos que la intersección de LICs no es necesariamente LIC.
- Veamos un ejemplo.
- $L = \{a^n b^n \mid n \geq 0\}$ es LIC
- $A = L \cdot c^*$ también es LIC (concatenación de LICs)
- Por la misma razón, $B = a^* \cdot \{b^n c^n \mid n \geq 0\}$ es LIC.

La intersección de LICs

- En la lección anterior comentamos que la intersección de LICs no es necesariamente LIC.
- Veamos un ejemplo.
- $L = \{a^n b^n \mid n \geq 0\}$ es LIC
- $A = L \cdot c^*$ también es LIC (concatenación de LICs)
- Por la misma razón, $B = a^* \cdot \{b^n c^n \mid n \geq 0\}$ es LIC.
- Pero la intersección no es LIC

$$A \cap B = \{a^n b^n c^n \mid n \geq 0\}$$

Referencias

- Autómatas de Pila. Kelly: Capítulo 3, secciones 3.7-3.8
- Lenguajes no incontextuales/Lema de bombeo. Sipser: Capítulo 2, sección 2.3