Autómatas de Pila

Teoría de Lenguajes

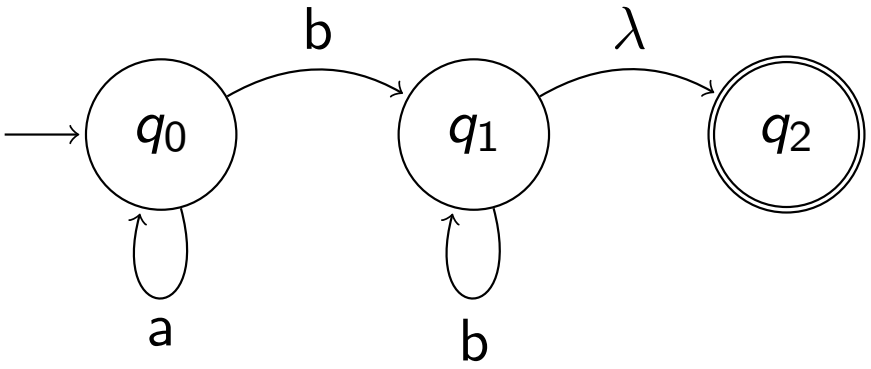
11 de Mayo de 2020
(día 53 de la cuarentena)

Introducción

- Lenguajes regulares
  - Son los más simples, el tipo 3 de la jerarquía de Chomsky
  - Reconocidos por autómatas finitos / expresiones regulares (formalismos equivalentes)
  - No tienen *memoria* (no pueden *contar*, salvo cosas como paridad)
  - Lema de pumping para demostrar que un lenguaje es **no** regular
- Lenguajes independientes (libres) del contexto
  - Tipo 2 en la jerarquía de Chomsky
  - Reconocidos por autómatas de pila y gramáticas libres de contexto
  - Mayor poder expresivo, la pila permite tener *memoria* y poder *contar* (ciertas cosas)

$L_1 = \{a^n b^n \mid n \geq 1\}$

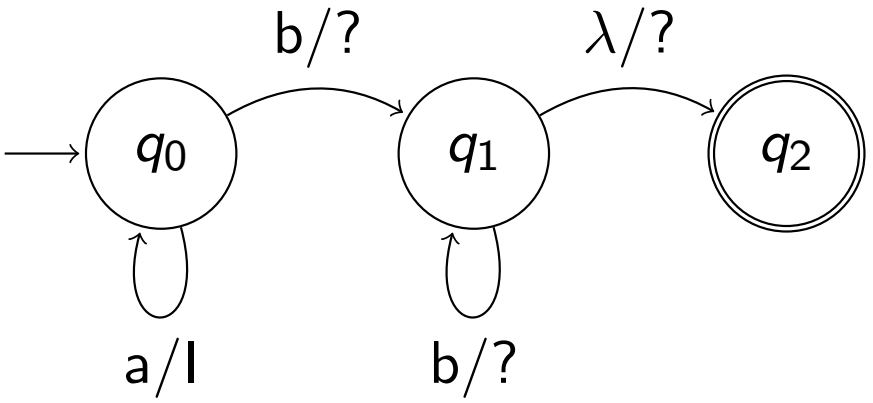
Con un AF podemos reconocer cadenas de a's seguidas de b's:



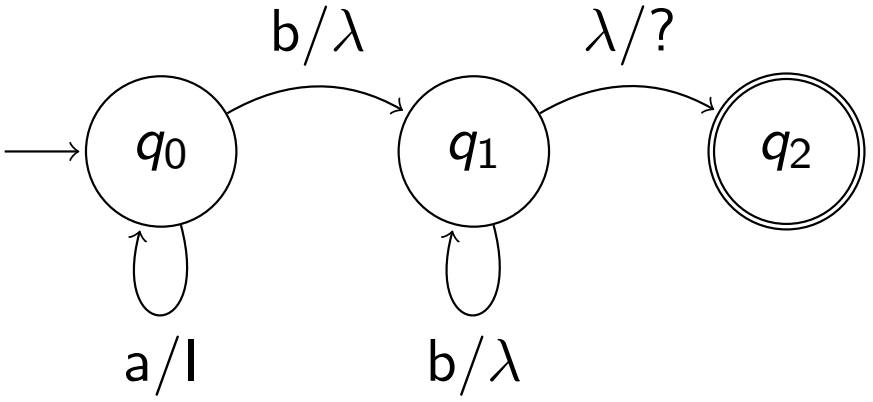
Pero esto reconoce $a^* b^+$ y no $a^n b^n$.
Vamos a usar la pila para contar n. Cada vez que leamos una a vamos a apilar un palote (I).

$L_1 = \{a^n b^n \mid n \geq 1\}$

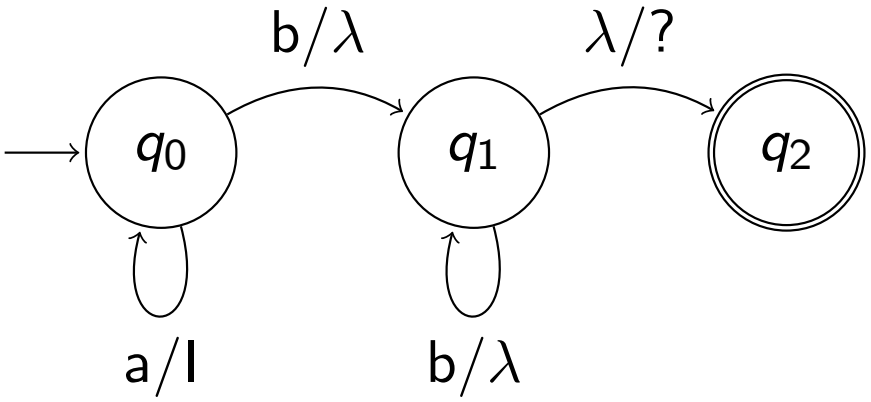
Vamos a usar la pila para contar n. Cada vez que leamos una a vamos a apilar un palote (I):



Luego, cada vez que leamos una b vamos a desapilar un palote:



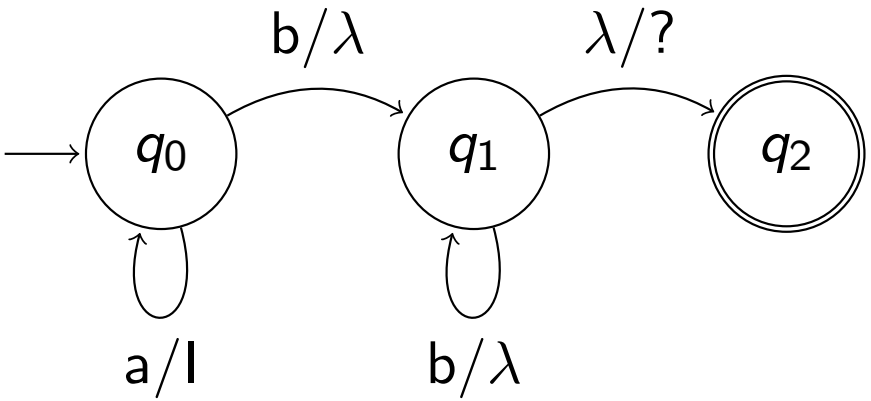
$L_1 = \{a^n b^n \mid n \geq 1\}$



Sólo queremos permitir la transición al estado final q_2 cuando $|a| == |b|$.
O sea, haber desapilado exactamente la misma cantidad de palotes que apilamos. O sea, tener la pila *vacia*.
Pero en los APs no podemos transicionar con la pila vacía!!

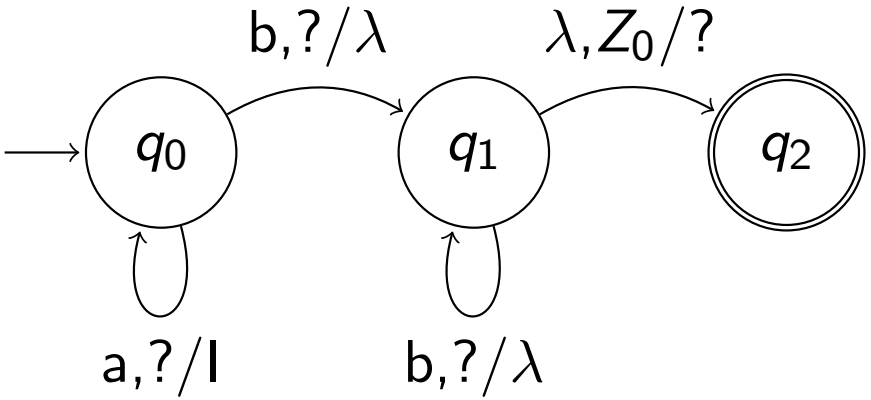
El formalismo define que hay un elemento inicial Z_0 en la pila.
Además, las transiciones se realizan según el tope de la pila.

$L_1 = \{a^n b^n \mid n \geq 1\}$

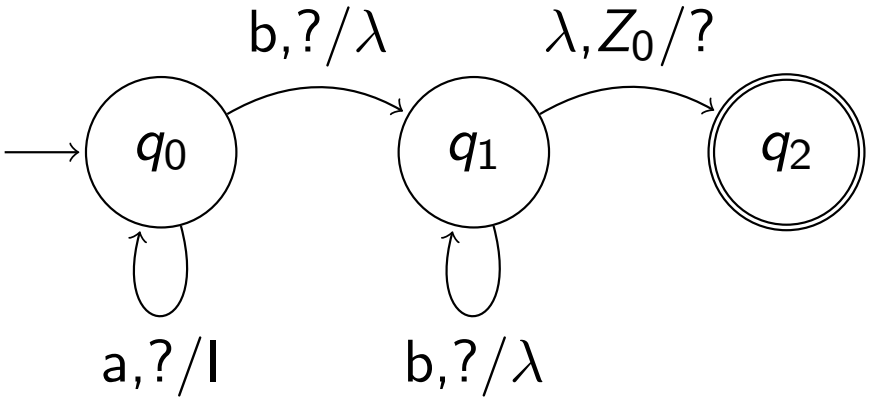


El formalismo define que hay un elemento inicial Z_0 en la pila. Además, las transiciones se realizan según el tope de la pila.

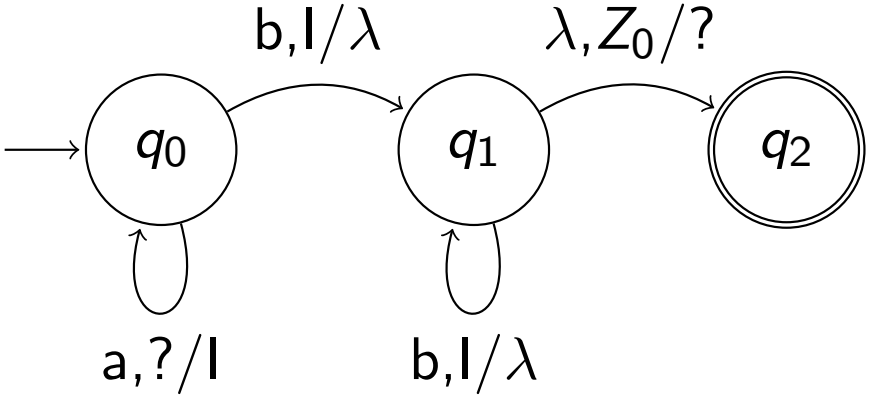
Si Z_0 esta en el tope de la pila, entonces ya desapilamos todos los palotes que habiamos apilado. Podemos entonces transicionar a q_2:



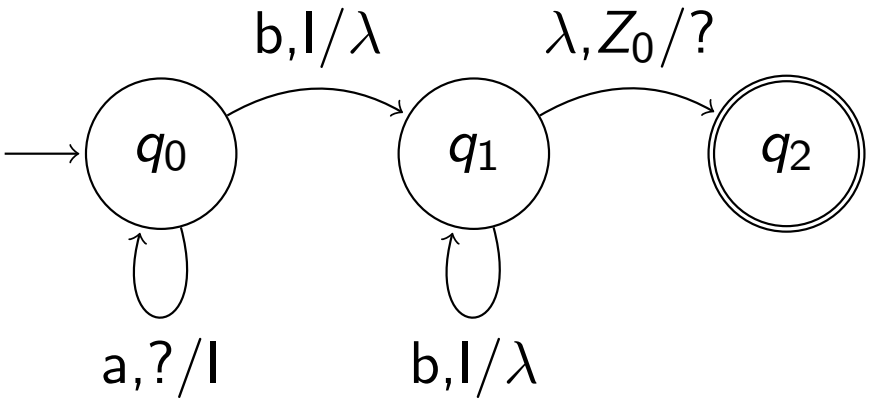
$L_1 = \{a^n b^n \mid n \geq 1\}$



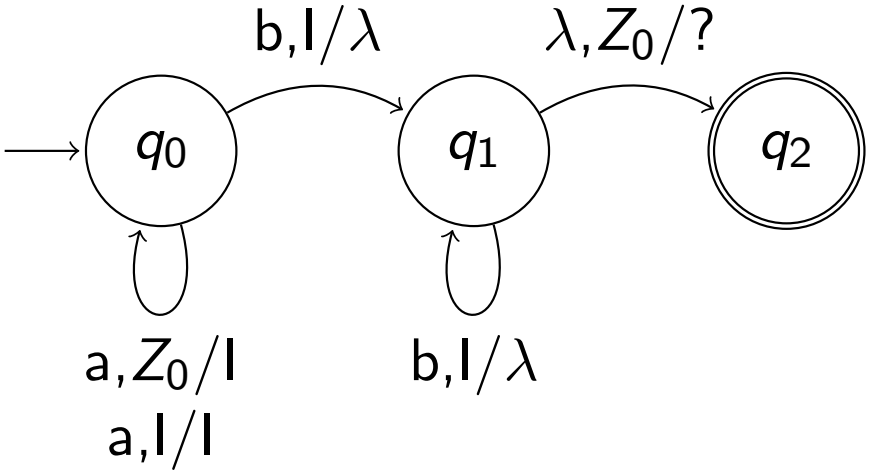
Permitimos leer b's sólo si hay palotes en la pila. Eso significa que anteriormente leimos (al menos igual cantidad de) a's:



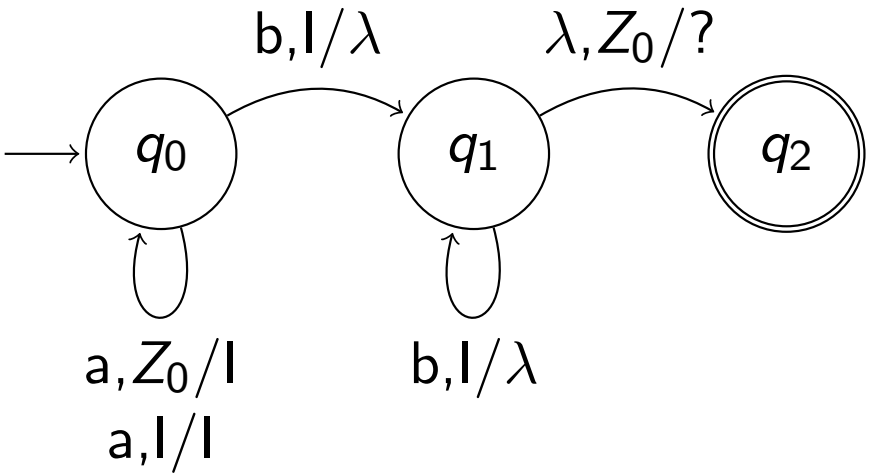
$L_1 = \{a^n b^n \mid n \geq 1\}$



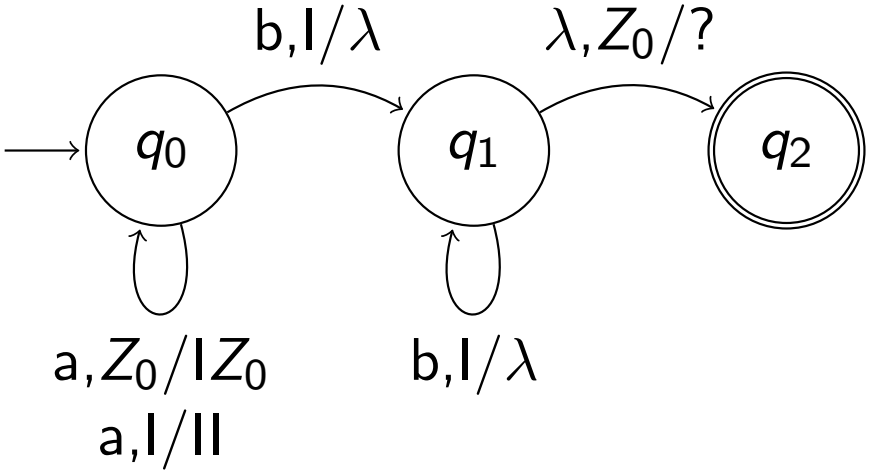
Al leer la primera a, vamos a tener Z_0 en la pila.
Pero necesitamos agregar otra transición para permitir leer más a's:



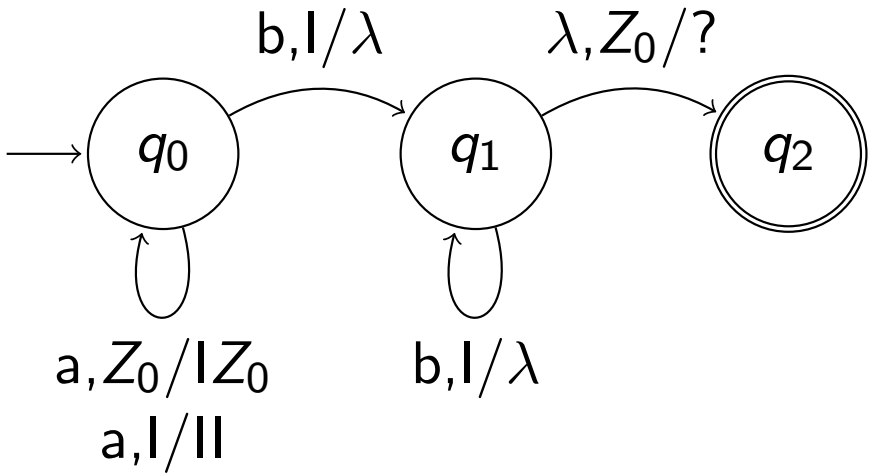
$L_1 = \{a^n b^n \mid n \geq 1\}$



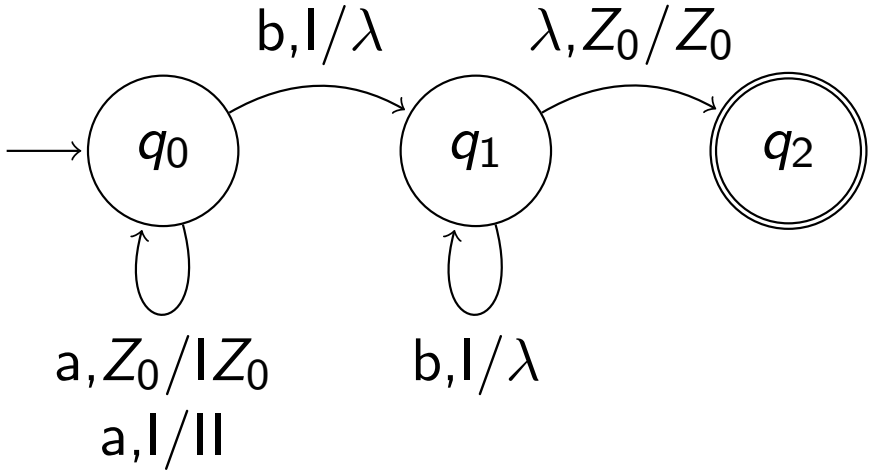
El formalismo nos dice que en cada transición el tope de la pila se saca. Esto significa que debemos volver a apilarlo si queremos conservarlo. Y que λ en este caso significa en realidad no apilar nada:



$L_1 = \{a^n b^n \mid n \geq 1\}$



Por último: en este AP una vez que llegamos al estado final q_2 no queremos seguir transicionando. Así que podemos vaciar la pila en la última transición o conservar Z_0 (elegimos esta opción):



Definición

Recordemos que un autómata finito se definía de la siguiente manera:

$$A =< Q, \Sigma, \delta, q_0, F >$$

Un autómata de pila M se define como:

$$M =< Q, \Sigma, \Gamma, \delta, q_0, Z_0, F >$$

Donde:

1. Q es un conjunto finito de estados
2. Σ es el alfabeto de entrada
3. Γ es el alfabeto de la pila
4. $q_0 \in Q$ es el estado inicial
5. $Z_0 \in \Gamma$ es el símbolo inicial de la pila
6. $F \subseteq Q$ es el conjunto de estados finales
7. δ es la función de transición: $Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow P(Q \times \Gamma^*)$

Configuración instantánea

En un autómata finito, teníamos como configuración instantánea un elemento de:

$$Q \times \Sigma^*$$

¿Cómo será una configuración instantánea en un autómata de pila?

$$(q, \alpha, \gamma) \in Q \times \Sigma^* \times \Gamma^*$$

donde:

1. q es el estado actual
2. α es la cadena de entrada que resta consumir
3. γ es el contenido de la pila

Lenguaje aceptado

Relación de transición entre configuraciones:

$\forall q_1, q_2 \in Q,\ a \in \Sigma,\ \alpha \in \Sigma^*,\ b \in \Gamma,\ \beta \in \Gamma^*$:

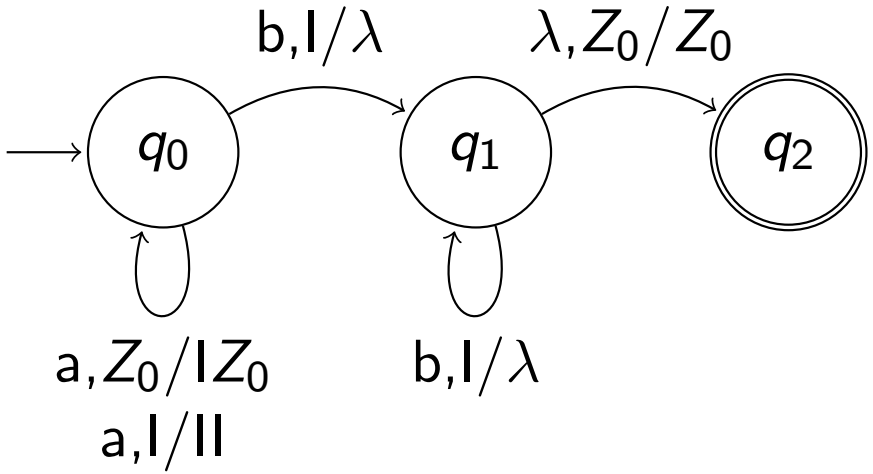
1. $(q_1, a\alpha, b\gamma) \vdash (q_2, \alpha, \beta\gamma) \iff (q_2, \beta) \in \delta(q_1, a, b)$
2. $(q_1, \alpha, b\gamma) \vdash (q_2, \alpha, \beta\gamma) \iff (q_2, \beta) \in \delta(q_1, \lambda, b)$

Lenguaje aceptado:

$$\alpha \in L(M) \iff \exists q_f \in F, \gamma \in \Gamma^* \mid (q_0, \alpha, Z_0) \vdash^* (q_f, \lambda, \gamma)$$

Notar que no nos importa que queda en la pila, es simplemente un $\gamma \in \Gamma^*$

$L_1 = \{a^n b^n \mid n \geq 1\}$



$$M_1 = < \{q_0, q_1, q_2\}, \{a, b\}, \{I, Z_0\}, \delta, q_0, Z_0, \{q_2\} >$$

$$(q_0, aabb, Z_0) \vdash (q_0, abb, IZ_0) \vdash (q_0, bb, IIZ_0)$$
$$\vdash (q_1, b, IZ_0) \vdash (q_1, \lambda, Z_0) \vdash (q_2, \lambda, Z_0) \checkmark$$

$$(q_0, aab, Z_0) \vdash (q_0, ab, IZ_0) \vdash (q_0, b, IIZ_0) \vdash (q_1, \lambda, IZ_0) \times$$

$$(q_0, abb, Z_0) \vdash (q_0, bb, IZ_0) \vdash (q_1, b, Z_0) \vdash (q_2, b, Z_0) \times$$

Lenguaje aceptado

Para los LIC aparece el concepto de *Lenguaje aceptado por pila vacía*: se acepta una cadena si existe una secuencia de transiciones por medio de la cual se consume toda la cadena dejando la pila vacía:

Lenguaje aceptado por pila vacía:

$$\alpha \in N(M) \iff \exists q_n \in Q \mid (q_0, \alpha, Z_0) \vdash^* (q_n, \lambda, \lambda)$$

Notar que en este caso no nos importa en que estado termina. Vale la pena recordar que si la pila se vacía antes de terminar de consumir la cadena, no se puede continuar transicionando (no hay tope de pila).

Lenguaje aceptado

En la teórica se demuestra que los lenguajes generados por **APND** por pila vacía son los mismos que los generados por **APND** por estado final, y que estos corresponden a todos los LIC

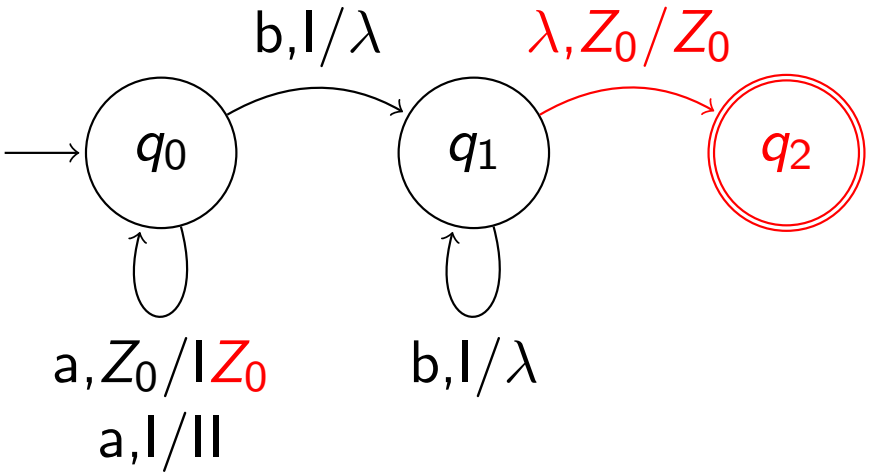
Lenguaje aceptado por estado final:

$$\alpha \in L(M) \iff \exists q_f \in F, \gamma \in \Gamma^* \mid (q_0, \alpha, Z_0) \vdash^* (q_f, \lambda, \gamma)$$

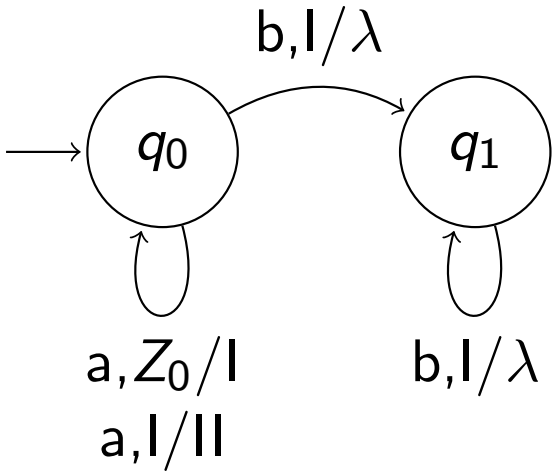
Lenguaje aceptado por pila vacía:

$$\alpha \in N(M) \iff \exists q_n \in Q \mid (q_0, \alpha, Z_0) \vdash^* (q_n, \lambda, \lambda)$$

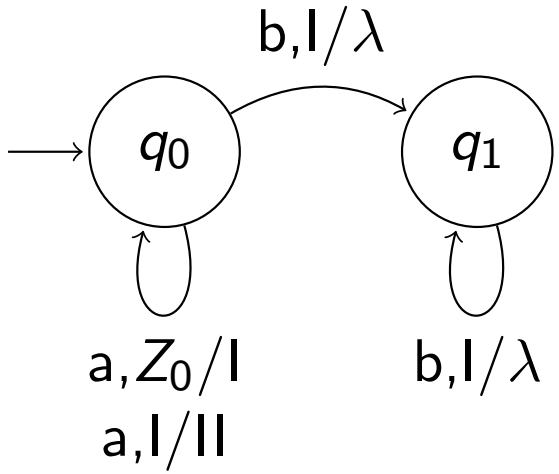
$L_1 = \{a^n b^n \mid n \geq 1\}$ por PV



Para vaciar la pila, necesitamos deshacernos de Z_0:



$L_1 = \{a^n b^n \mid n \geq 1\}$ por PV



$$M_2 =< \{q_0, q_1\}, \{a, b\}, \{I, Z_0\}, \delta, q_0, Z_0, \emptyset >$$

$(q_0, aabb, Z_0) \vdash (q_0, abb, I) \vdash (q_0, bb, II) \vdash (q_1, b, I) \vdash (q_1, \lambda, \lambda)$ ✓
$(q_0, aab, Z_0) \vdash (q_0, ab, I) \vdash (q_0, b, II) \vdash (q_1, \lambda, I)$ ×
$(q_0, abb, Z_0) \vdash (q_0, bb, I) \vdash (q_1, b, \lambda)$ ×

$L_2 = \{\omega\#\omega^r \mid \omega \in (0|1)^*\}$

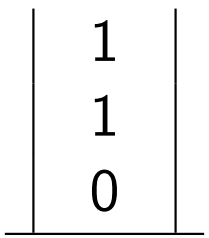
Ejemplos de cadenas:

$\alpha_1 = \lambda$ ✗ $\alpha_2 = \#$ ✓ $\alpha_3 = 0\#0$ ✓ $\alpha_4 = 01\#01$ ✗ $\alpha_5 = 01\#10$ ✓

El lenguaje se compone de cadenas *espejadas* por el símbolo numeral.

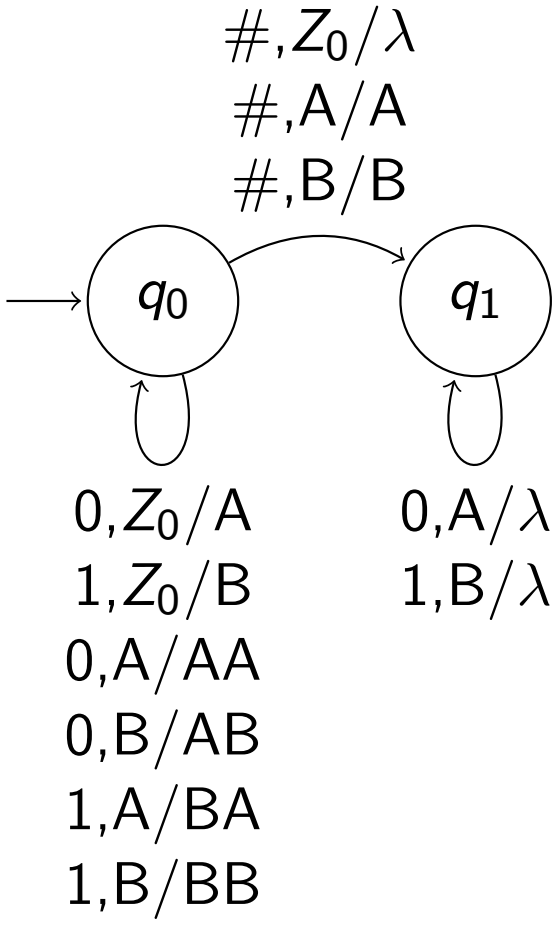
Una pila funciona de forma *LIFO (Last In, First Out)*

Usamos la pila para leer la reversa de una cadena ω apilada.
Si $\omega = 011$, entonces en la pila tenemos

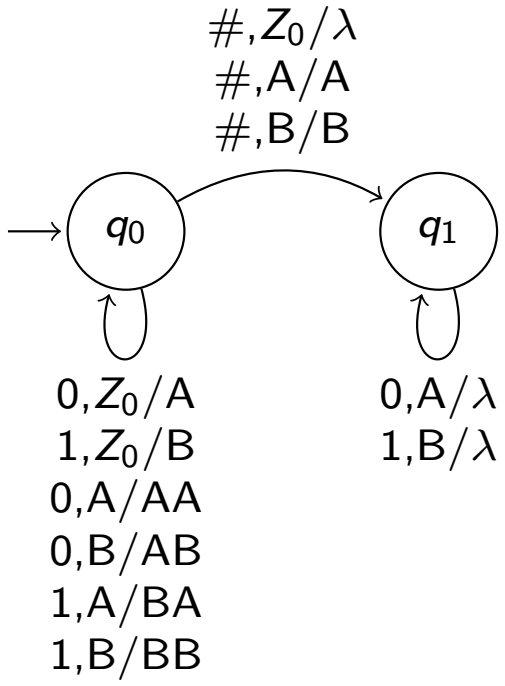


$L_2 = \{\omega\#\omega^r \mid \omega \in (0|1)^*\}$

1. Leemos ω y apilamos
2. Leemos $\#$ sin tocar la pila
3. Desapilamos verificando que coincida con ω^r



$L_2 = \{\omega\#\omega^r \mid \omega \in (0|1)^*\}$

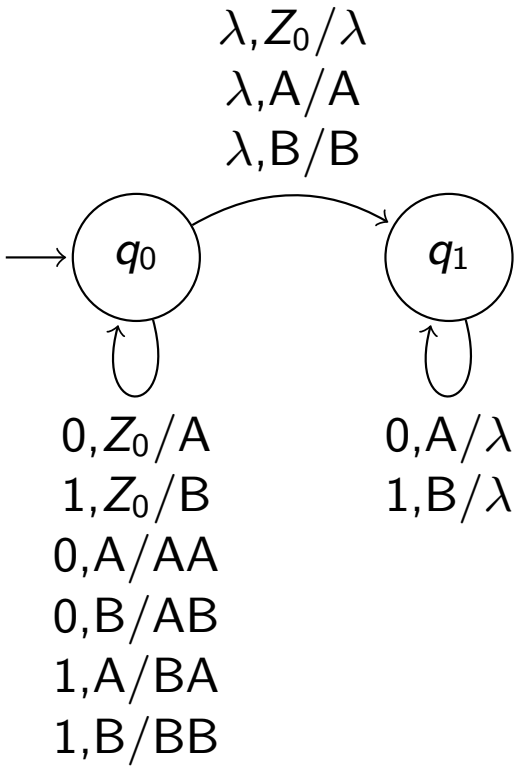


- Esta solución es por pila vacía. Si quisieramos hacerlo por estado final, qué deberíamos hacer?
- Por qué tenemos la transición $\#, Z_0/\lambda$?
- Que pasa con la cadena $\alpha = 01\#01$?

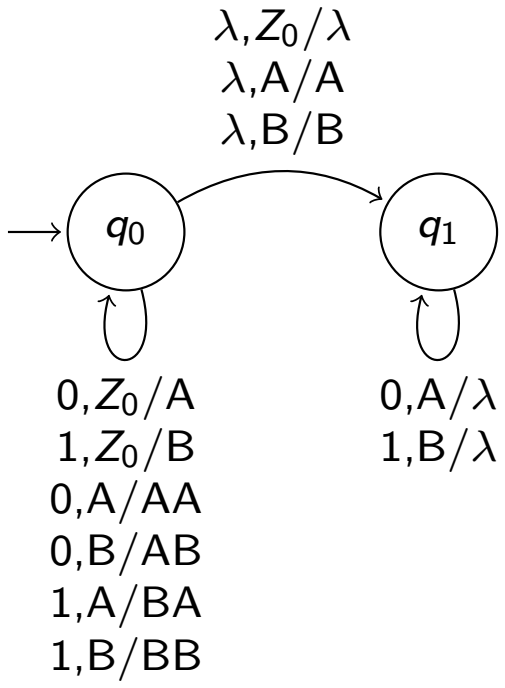
$L_3 = \{\omega\omega^r \mid \omega \in (0|1)^*\}$

Tenemos un lenguaje muy similar al anterior, pero ahora sin el #. Podemos probar reemplazando # por λ y ver que pasa, usando algunas cadenas como

$\alpha_1 = \lambda$ ✓ $\alpha_2 = 00$ ✓ $\alpha_3 = 1001$ ✓ $\alpha_4 = 1$ × $\alpha_5 = 10$ × $\alpha_6 = 1010$ ×



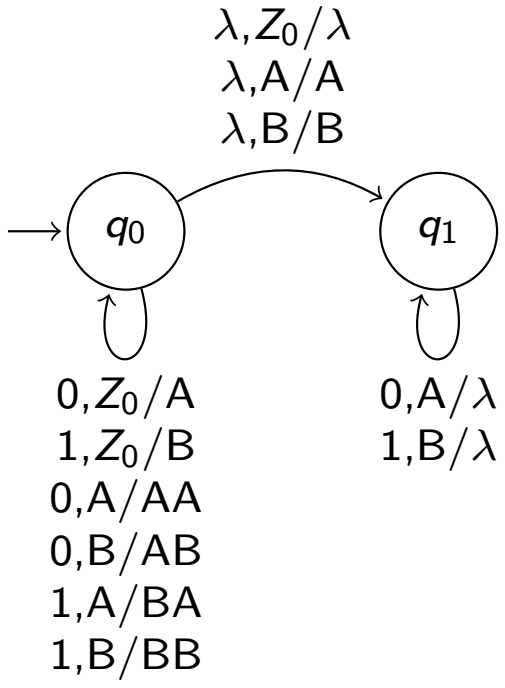
$L_3 = \{\omega\omega^r \mid \omega \in (0|1)^*\}$



Funciona ?? Sí ✓

Nota: tenemos un AP no deterministico. Cómo vemos eso? Por ejemplo estando en q_0 con B en el tope de la pila y teniendo un 1 por leer, puedo transicionar a q_0 por $1, B/BB$ o transicionar a q_1 por $\lambda, B/B$.

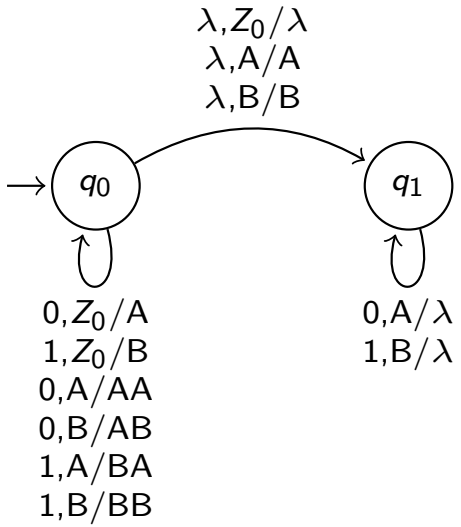
$L_3 = \{\omega\omega^r \mid \omega \in (0|1)^*\}$



Podemos eliminar el no-determinismo? No ×

Lo necesitamos porque no sabemos cuando terminamos de leer ω y comenzamos con ω^r (antes teníamos el # en el medio).

$L_3 = \{\omega\omega^r \mid \omega \in (0|1)^*\}$



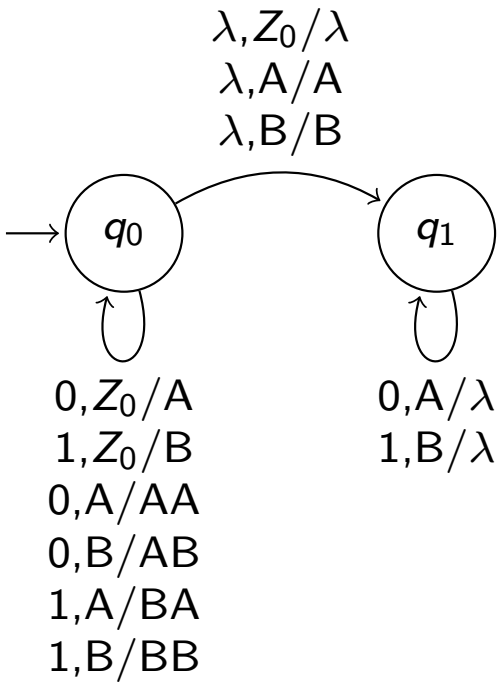
Recordemos que en un autómata no determinístico, necesitamos al menos una secuencia de transiciones validas para aceptar una cadena, y que no exista ninguna secuencia válida para rechazarla. Ejemplos:

- $\alpha_2 = 00 : (q_0, 00, Z_0) \vdash (q_1, 00, \lambda)$ ×
  $(q_0, 00, Z_0) \vdash (q_0, 0, A) \vdash (q_1, 0, A) \vdash (q_1, \lambda, \lambda)$ ✓
- $\alpha_5 = 10 : (q_0, 10, Z_0) \vdash (q_0, 0, B) \vdash (q_1, 0, B)$ ×
  $(q_0, 10, Z_0) \vdash (q_1, 10, \lambda)$ ×
  $(q_0, 10, Z_0) \vdash (q_0, 0, B) \vdash (q_0, \lambda, AB) \vdash (q_1, \lambda, AB)$ ×

AP determinístico

M autómata de pila es determinístico si $\forall q \in Q, a \in \Sigma, z \in \Gamma$ vale

1. $|\delta(q, a, z)| \leq 1$
2. $|\delta(q, \lambda, z)| \leq 1$
3. Si $|\delta(q, \lambda, z)| = 1$ entonces $|\delta(q, a, z)| = 0$



Item 1 ✓ *Item* 2 ✓ *Item* 3 ×

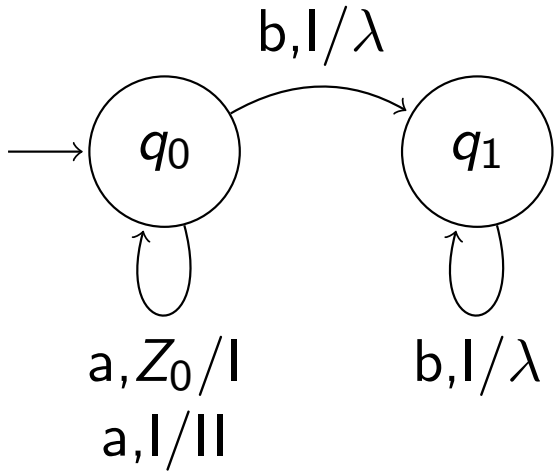
Equivalencia entre APD y APND ?

Mencionamos que no podemos construir un APD para $\{\omega\omega^r \mid \omega \in (0|1)^*\}$. Es decir, existen lenguajes independientes del contexto que no son generados por ningún APD. Esto implica que (a diferencia de los autómatas finitos) no existe una equivalencia entre APD y APND.

También dijimos que los lenguajes generados por pila vacia con APND son los mismos que los generados por estados finales con APND.
Qué pasará si nos restringimos a APD?

$L_4 = \{a^n b^m \mid 1 \leq n \leq m\}$

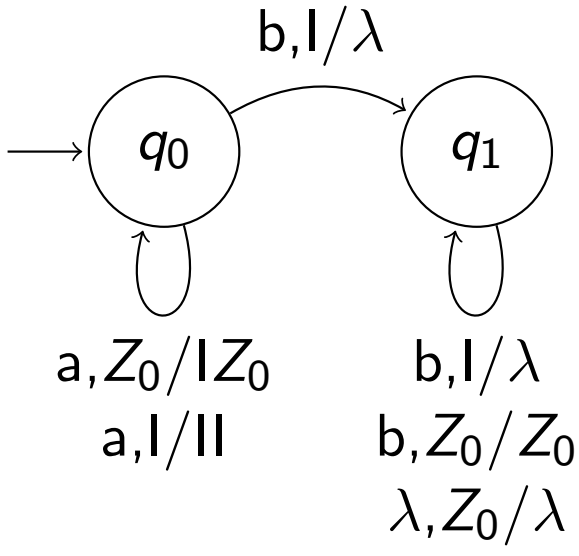
Intentemos construir un APD por pila vacía que acepte L_4, basandonos en el AP construido antes para $\{a^n b^n \mid 1 \leq n\}$ por pila vacía (que era determinístico):



Queremos que $|b| \geq |a|$, pero la pila se vacia cuando $|b| = |a|$!!!

$L_4 = \{a^n b^m \mid 1 \leq n \leq m\}$

Probemos manteniendo a Z_0 en la base de la pila:



Esto funciona pero es no determinístico!
Cuál es el *problema*? Los lenguajes generados por APD por pila vacía son libres de prefijos. Es decir, si $\alpha \in L$ entonces $\forall \beta \neq \lambda, \alpha\beta \notin L$

Lenguaje libre de prefijos

Recordemos la siguiente definición:

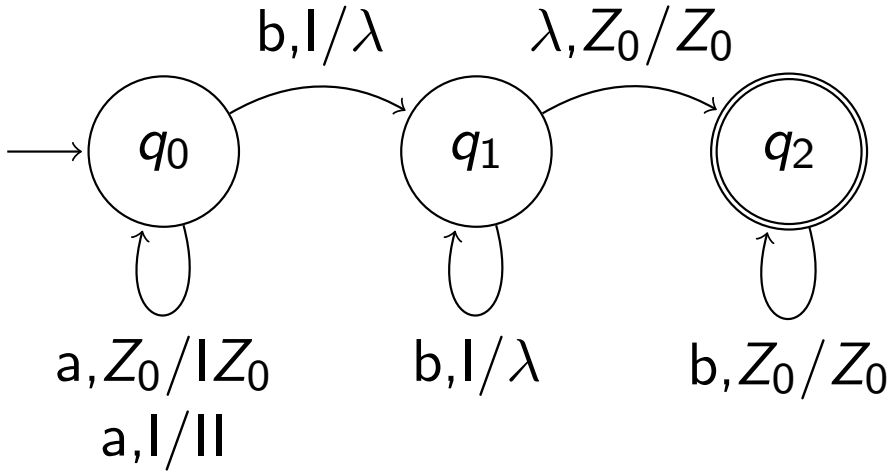
Lenguaje aceptado por pila vacía:

$$\alpha \in N(M) \iff \exists q_n \in Q \mid (q_0, \alpha, Z_0) \vdash^* (q_n, \lambda, \lambda)$$

Para aceptar una cadena α, esta debe haberse consumido entera y haberse vaciado la pila. Por lo tanto, cualquier otra cadena que tenga de prefijo a α no va a poder operar, se va a quedar *trabada*. Y al estar considerando un APD, no existe otra secuencia de transiciones.

$L_4 = \{a^n b^m \mid 1 \leq n \leq m\}$

Volviendo a nuestro lenguaje, podemos dar un APD por estados finales?

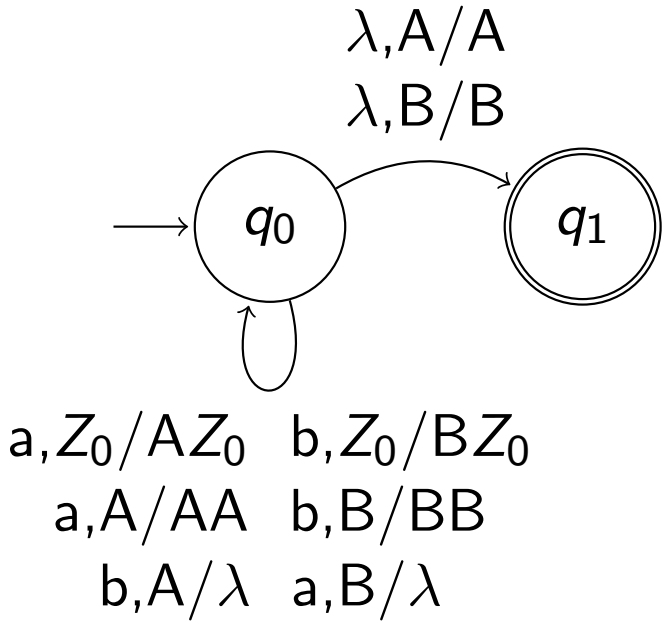


Concluimos que los lenguajes generados por APD por pila vacía **no son equivalentes** a los lenguajes generados por APD por estados finales.

$L_5 = \{\omega \mid \omega \in \{a, b\}^* \wedge |a| \neq |b|\}$

Nuestro lenguajes se compone de cadenas sobre $\Sigma = \{a, b\}^*$ con distinta cantidad de a's que b's. Ejemplos: $\alpha_1 = a$, $\alpha_2 = aba$, $\alpha_3 = abb$, $\alpha_4 = bbb$.

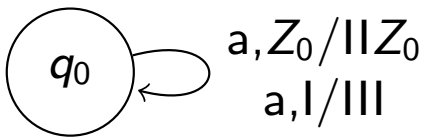
Pista: necesitamos dos símbolos en la pila y también conservar Z_0 para evitar que la pila se vacie.



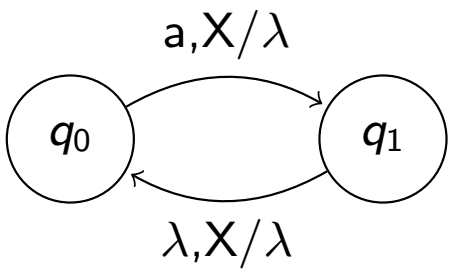
Otros ejercicios

- $L_6 = \{\omega \mid \omega \in \{a, b\}^* \wedge \forall$ prefijo β de ω vale $|\beta_a| \geq |\beta_b|\ \}$

- $L_7 = \{a^n b^m \mid 2n > m > 0\}$
  (pista: puedo apilar más de un simbolo por transición)



- $L_8 = \{a^n b^m \mid n > 2m > 0\}$
  (pista: para desapilar dos simbolos leyendo una sola letra necesito una transición a un estado nuevo leyendo y desapilando y luego volver leyendo λ y desapilando)



Lenguajes no independientes del contexto

- $L_1 = \{a^n b^n c^n \mid n \geq 1\}$
  Al terminar de leer las b's perdi la cuenta de n y no puedo verificarlo contra la cantidad de c's.

- $L_2 = \{\omega\omega \mid \omega \in (0|1)^*\}$
  Como ya vimos, los símbolos de ω quedan apilados al revés y no puedo darlos vuelta.

En ambos casos tener una segunda pila resolvería los inconvenientes, pero eso ya equivale a una máquina de Turing (correspondiente a lenguajes de tipo 0, llamados lenguajes *sin restricciones*).
En particular, L_1 y L_2 son lenguajes sensibles al contexto (tipo 1).

Como habrán visto en la teórica, existe un lema de pumping que se puede usar para probar que un lenguaje **no** es independiente del contexto.

Repaso / Conclusiones

- LIC (tipo 2): mayor poder expresivo, pueden contar ciertas cosas
- AP: formalismo, autómata con una pila, aceptan LIC
- Definición (tupla), configuración instantánea, relación entre CIs
- Lenguaje generado por estado final y por pila vacía
- Determinismo en APs
- Equivalencias (APD-APND, APND por EF-PV, APD por EF-PV)
- Ejercicios
- Lenguajes no independientes del contexto