Autómata Push-Down (APD)

Jaime A. Pavlich Mariscal

Limitación de Lenguajes Regulares

- Supongamos el siguiente lenguaje:

$$B = \{0^n 1^n | n \geq 0\}$$

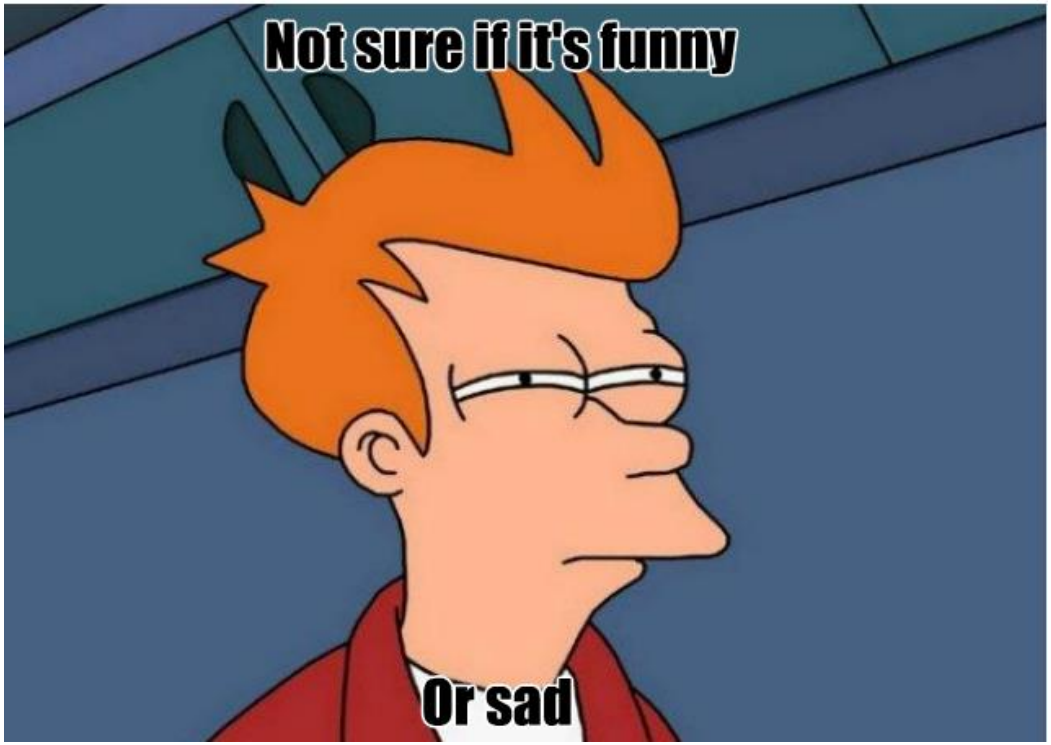
- ¿Existe un AFD o AFND que pueda reconocer strings de B?

Limitación de Lenguajes Regulares

- Para poder hacer el autómata se necesita recordar
  - Cuantos 0's
  - Cuantos 1's
- Pero un autómata no puede llevar un contador!

Limitación de Lenguajes Regulares

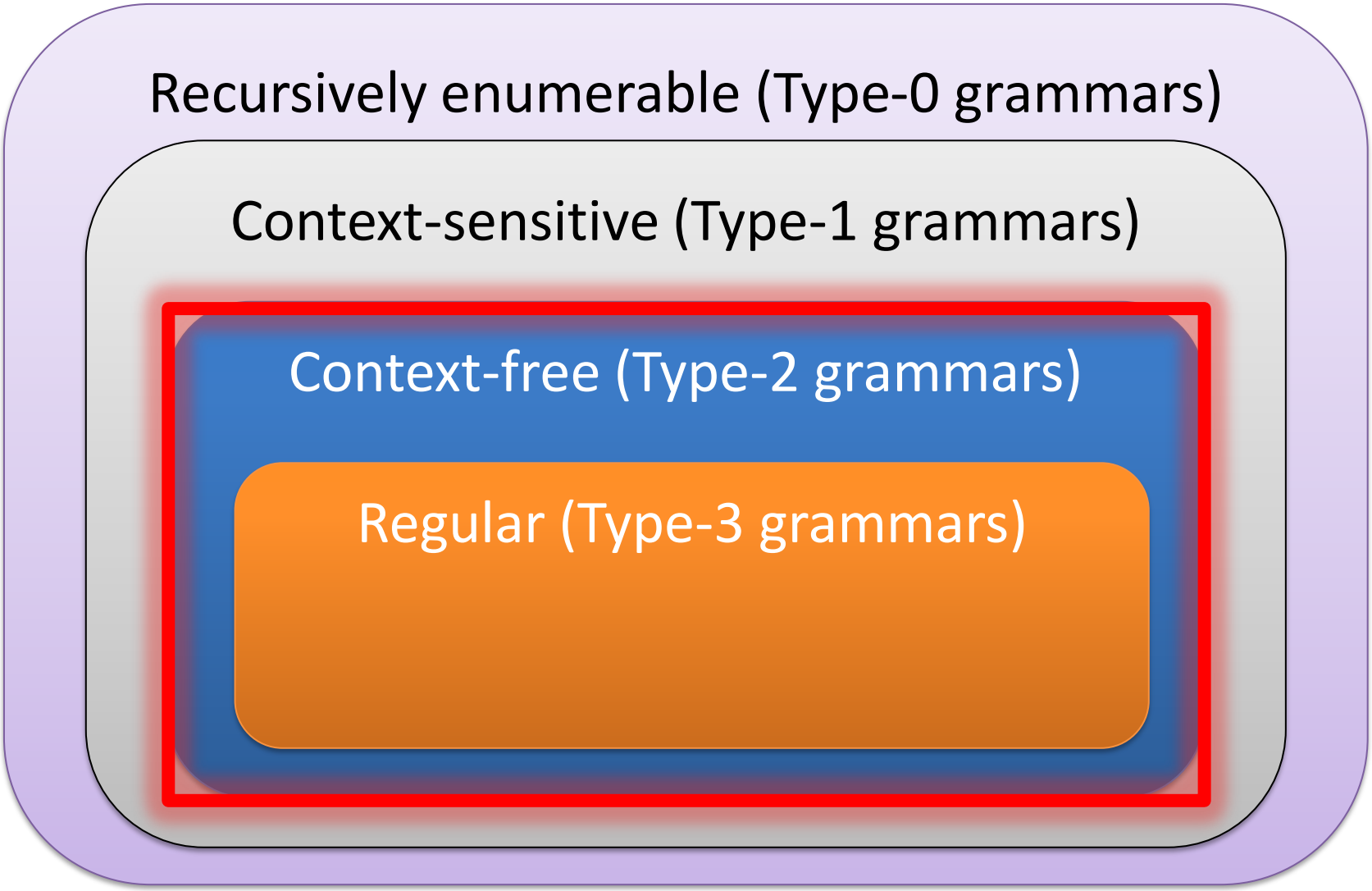
- La cantidad de 0’s y 1’s no está limitado
  - No es posible registrarlos con un número finito de estados.

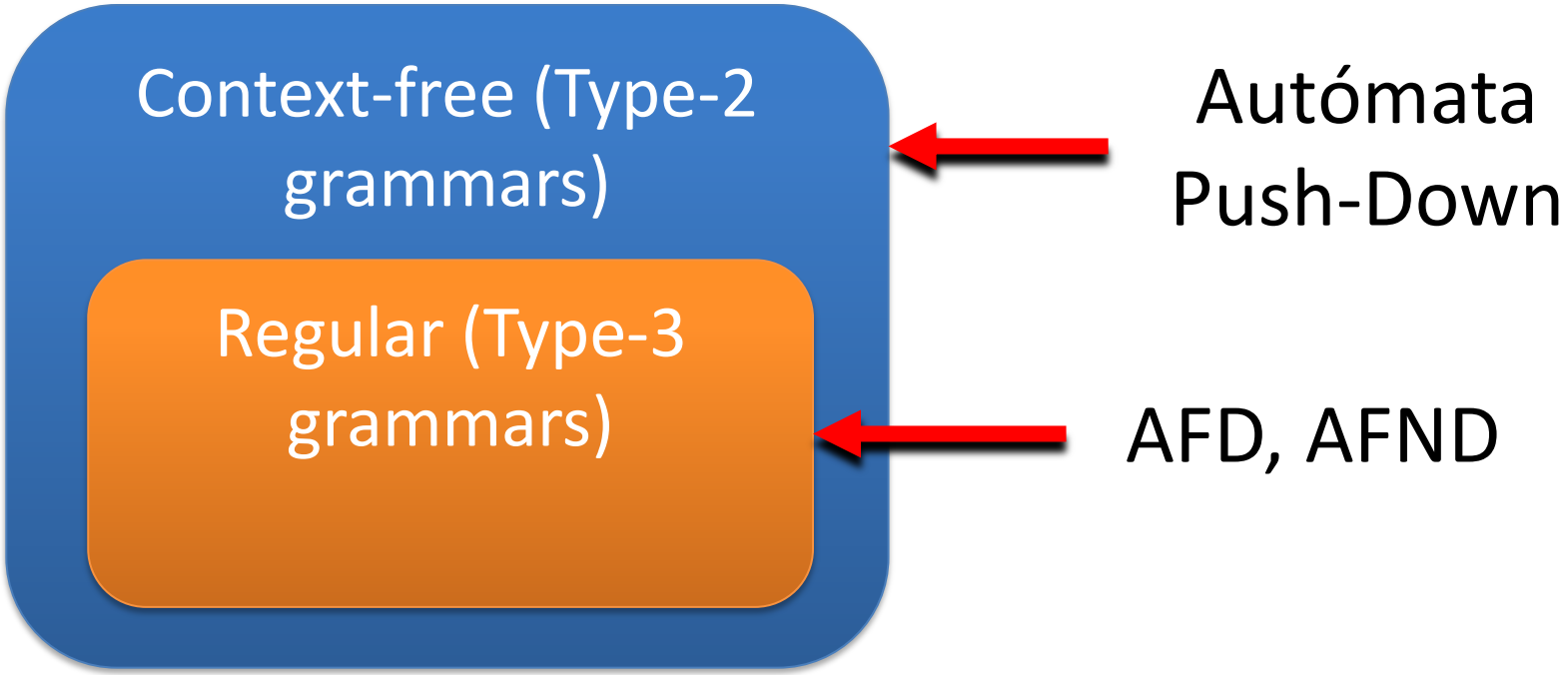


Limitación de Lenguajes Regulares

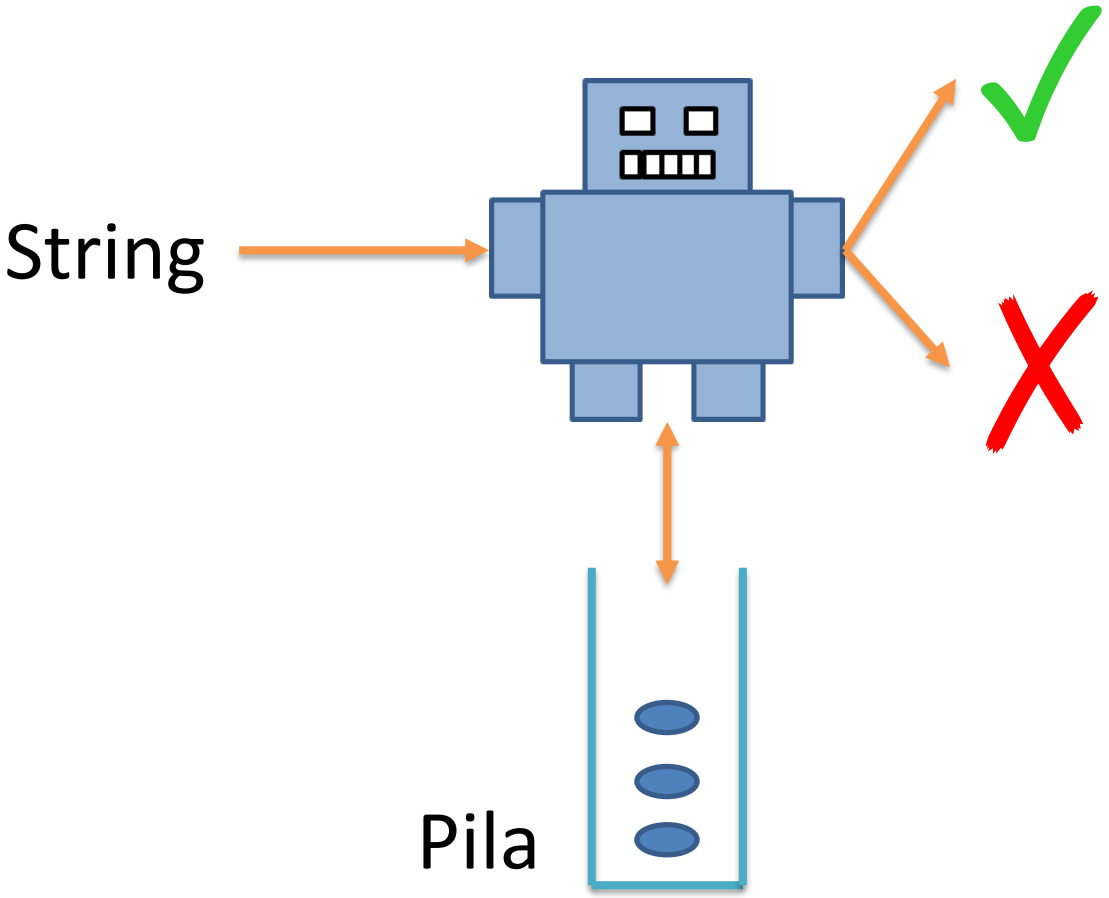
- Pumping lemma
  - Si L es un lenguaje regular, entonces existe una constante *n* tal que, si *w* es cualquier palabra de L tal que $|w| > n$, *w* puede ser dividido en tres partes *w=xyz* que satisfacen las siguientes condiciones:
    1. $\forall k \geq 0 \;\; xy^k z \, \epsilon \, L,$
    2. $|y| > 0$
    3. $|xy| \leq n$

Tipos de Gramáticas



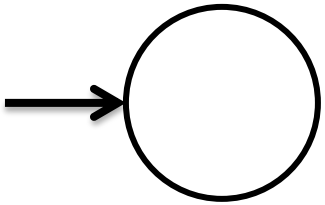

Context-free (Type-2 grammars)
Regular (Type-3 grammars)
Autómata Push-Down
AFD, AFND

Autómata Push-Down

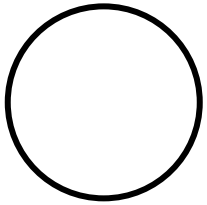


Notación

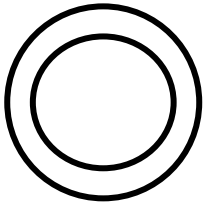
Estados



Estado inicial

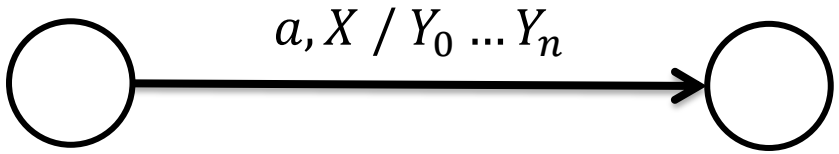


Estado



Estado final

Transiciones

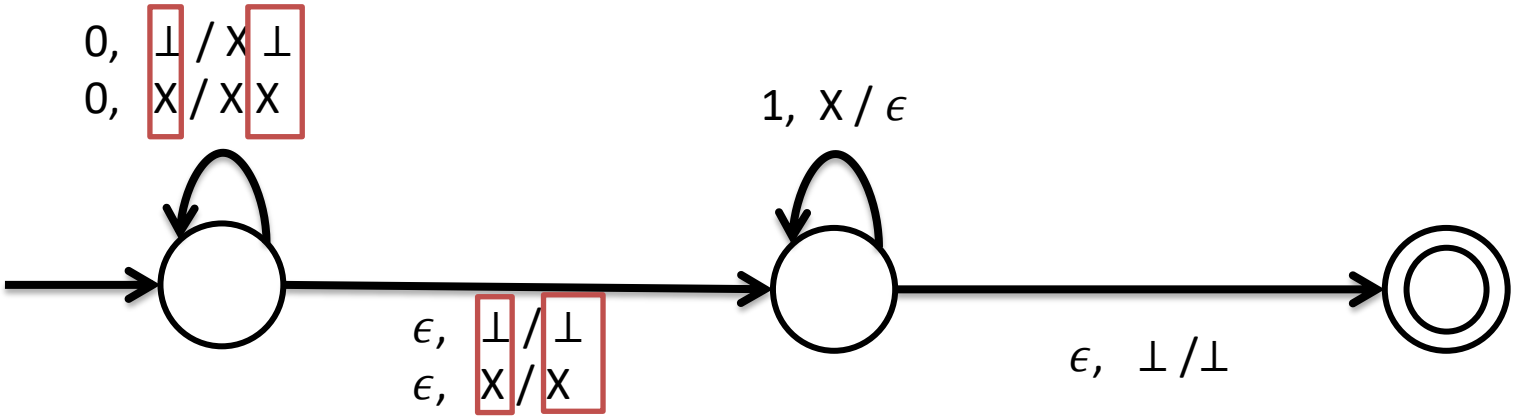


- a: Símbolo a leer del string de entrada, o épsilon
- X: Símbolo que **debe** estar en el tope de la pila para realizar la transición
- $Y_0 \ldots Y_n$: Cómo queda el **tope** de la pila **después** de la transición
  - X/X : La pila no cambia
  - X/ϵ: Se remueve el tope de la pila
  - X/Y : Se reemplaza X por Y en el tope de la pila
  - X/YX : Se inserta Y al tope de la pila
  - X/YZ : Se reemplaza X por Y y Z en el tope de la pila

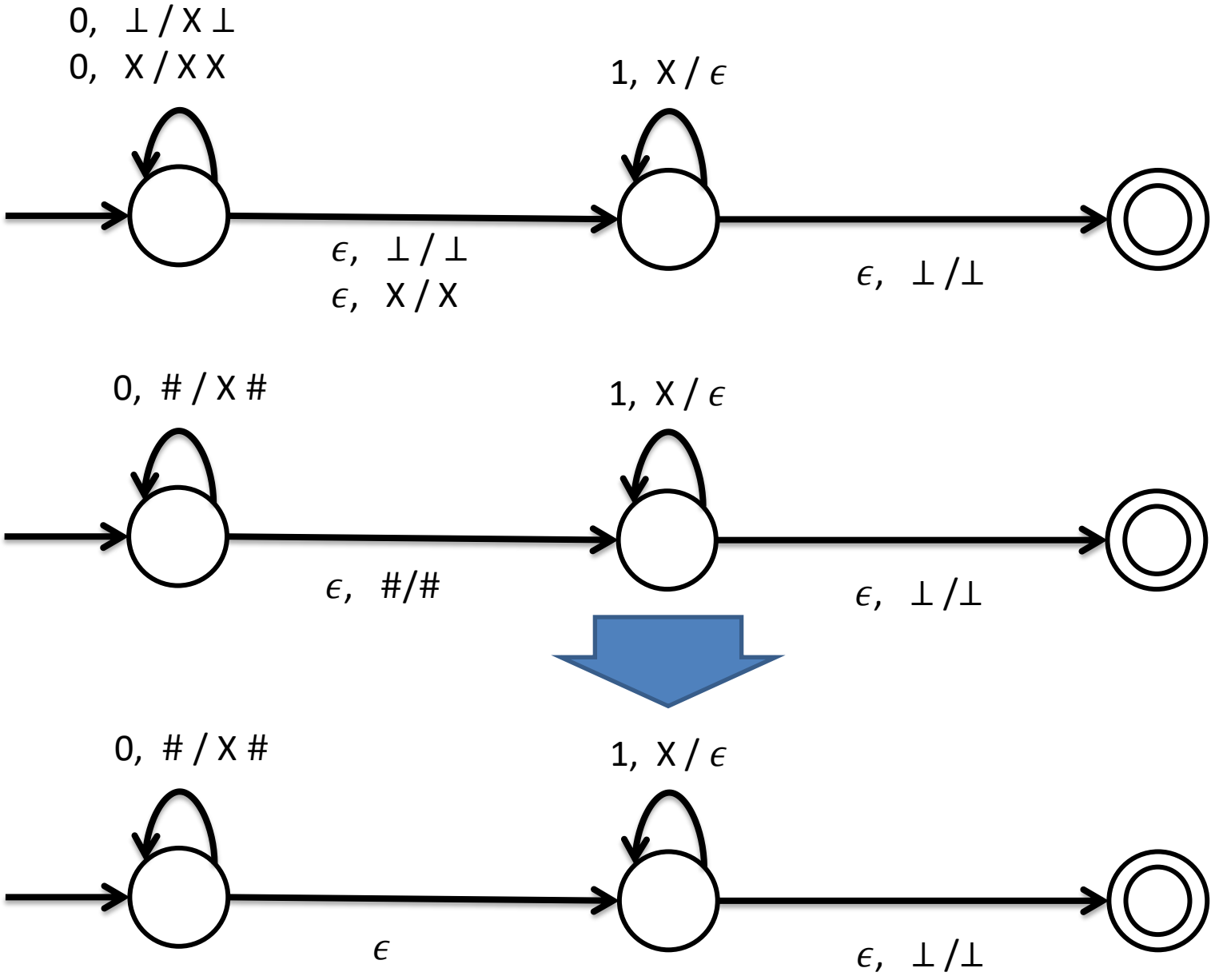
Ejemplo

- APD que reconoce el siguiente lenguaje

$$\{0^n 1^n | n \geq 0\}$$



Simplificando la notación



Definición Formal

- Un autómata push-down (APD) es la siguiente tupla:

$$A = (Q, \Sigma, \Gamma, \delta, q_0, \perp, F)$$

Q : Conjunto finito de *estados*

Σ: Alfabeto de *entrada*

Γ: Alfabeto de la *pila*

δ: $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$: *Función de transición*.

$q_0 \in Q$: *Estado inicial*

$\perp \in \Gamma$: Símbolo inicial, en la base de la pila

$F \subseteq Q$: Conjunto de *estados finales* o de aceptación

Construya un APD

- Que acepte strings que tengan la misma cantidad de ceros que de unos, en cualquier lugar del string ($\Sigma = \{0,1\}$)
- Que acepte palíndromos de longitud par ($\Sigma = \{a, \dots, z\}$)
- Que acepte strings que comiencen en 1 y tengan el doble de ceros que de unos

Construya APD

- Que determine si secuencias de if – else están correctamente escritas
  - Para cada if, debe existir un else
  - Asuma que la sentencia if se escribe *"I"*
  - Asuma que la sentencia else se escribe *"E"*
  - Cualquier cosa que no sea *"I"* o *"E"* corresponde a otras sentencias del lenguaje
    - $\Sigma = \{a, \dots, z, I, E\}$
- Que acepte el siguiente lenguaje

$$\{a^i b^j c^k | i = j \vee j = k\}$$