Autómatas de Pila II (*pushdown automata*)

Alan Reyes-Figueroa
Teoría de la Computación (Aula 18) 09.octubre.2023

Construcción de PDA
Ejemplos

Ejemplo 1

Construir un autómata de pila para el lenguaje:

$L = \{w \in \{0,1\}^*: w = w^R\}$ (palíndromos)

Ej: 1000001, 001 100, 00100, ep, 1, 0, 010, …

- Símbolos de entrada (inputs): 0, 1, ϵ
- Símbolos de pila (stack): Z_0, 0, 1, ϵ
- Estados: q = start, p = parsing, f = final
- Estado final: f

Ejemplo 1

- Transiciones:
  - $\delta(q, 0, Z_0) = \{(q, 0Z_0)\}$

    $\delta(q, 1, Z_0) = \{(q, 1Z_0)\}$

    $\delta(q, 0, 0) = \{(q, 00)\}$

    $\delta(q, 1, 0) = \{(q, 10)\}$

    $\delta(q, 0, 1) = \{(q, 01)\}$

    $\delta(q, 1, 1) = \{(q, 11)\}$

    Para agregar un símbolo imput a la pila.

Ejemplo 1

- $\delta(q, \epsilon, Z_0) = \{(p, Z_0)\}$
  $\delta(q, 0, Z_0) = \{(p, Z_0)\}$
  $\delta(q, 1, Z_0) = \{(p, Z_0)\}$
  $\delta(q, \epsilon, 0) = \{(p, 0)\}$
  $\delta(q, 0, 0) = \{(p, 0)\}$
  $\delta(q, 1, 0) = \{(p, 0)\}$
  $\delta(q, \epsilon, 1) = \{(p, 1)\}$
  $\delta(q, 0, 1) = \{(p, 1)\}$
  $\delta(q, 1, 1) = \{(p, 1)\}$

  Lee un símbolo input y pasa al estado p, sin alterar la pila.

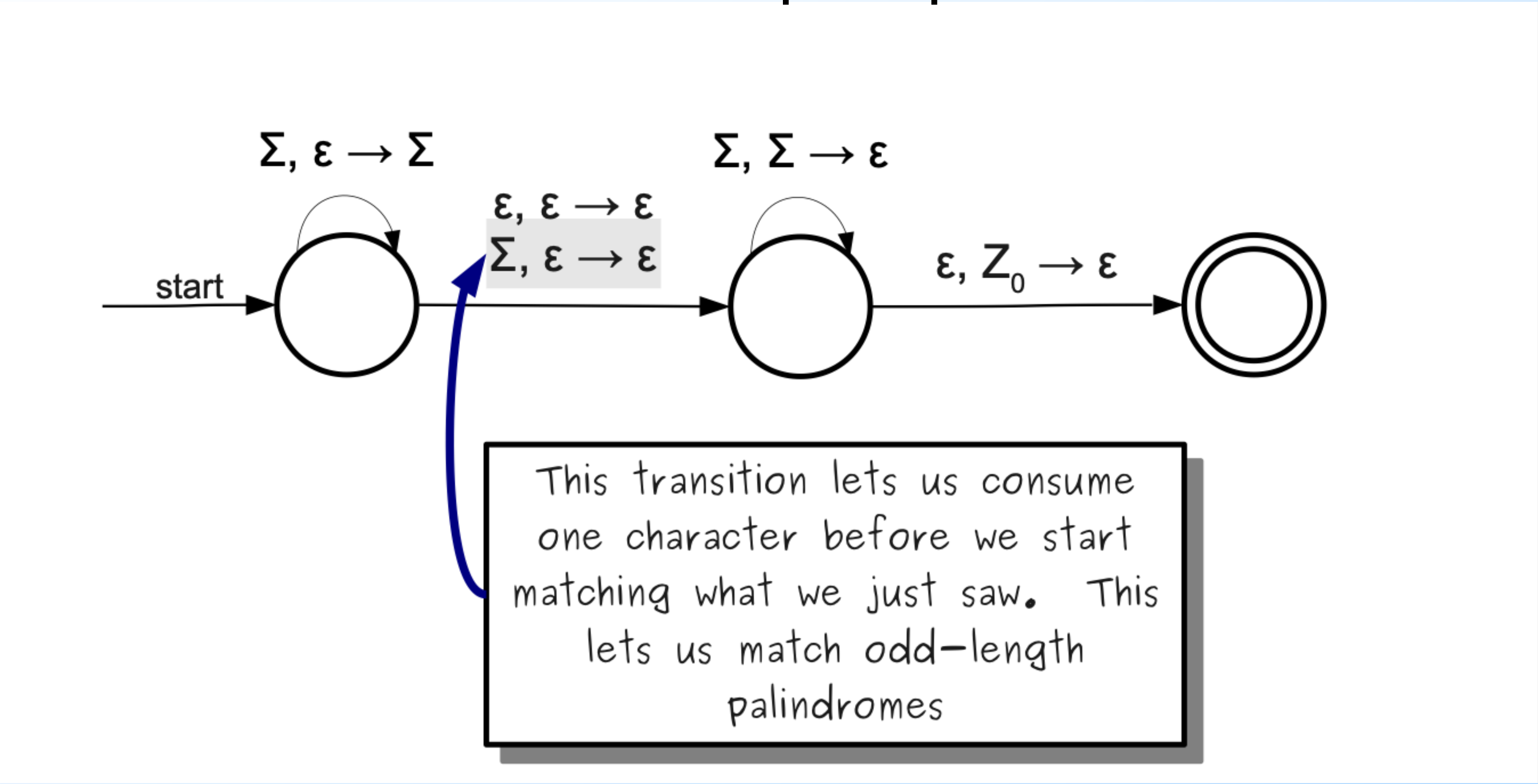
Ejemplo 1

- $\delta(p, 0, 0) = \{(p, \epsilon)\}$
  $\delta(p, 1, 1) = \{(p, \epsilon)\}$
  Borra un símbolo de la pila, siempre que coincida con el símbolo imput leído.

- $\delta(p, \epsilon, Z_0) = \{(f, Z_0)\}$
  Al terminar la lectura, pasa al estado de aceptación f sólo si la pila está vacía.

Ejemplo 1

Autómata de Pila para palíndromos



Ejemplo 2

Construir un autómata de pila para el lenguaje:

$L = \{1^n \geq 1^m : m, n \in \mathbb{N} \ \wedge \ n \geq m\}$

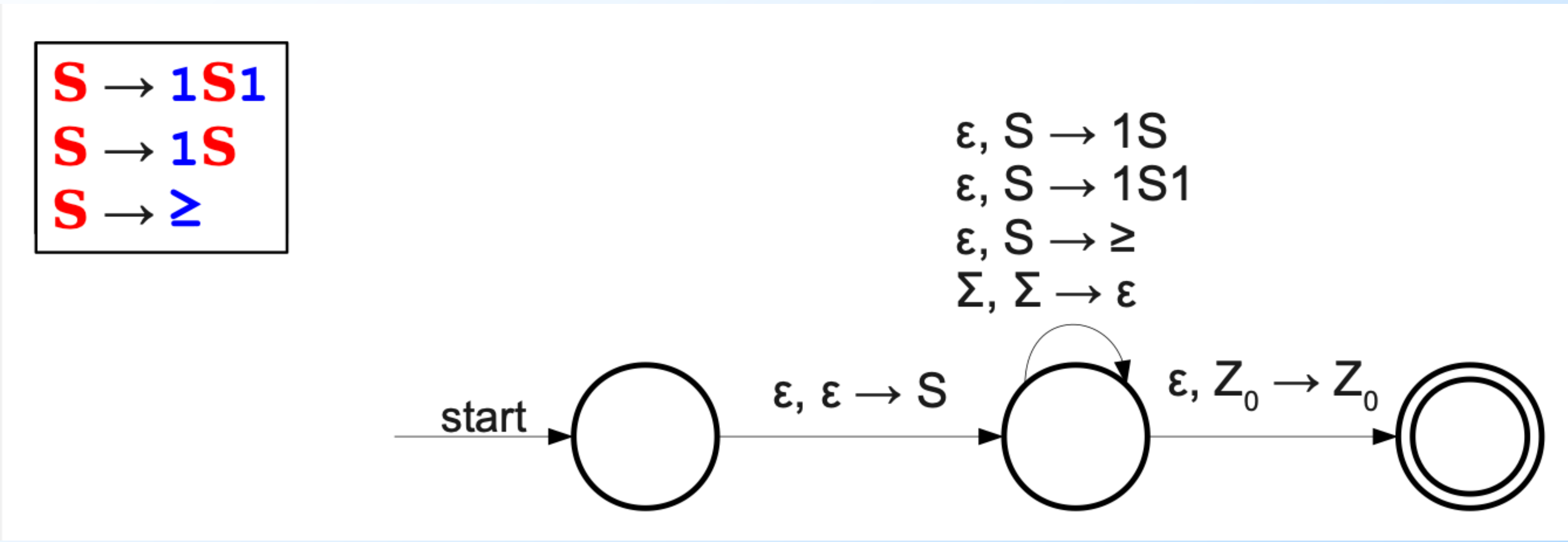
Tenemos la gramática:

$S \ \rightarrow \ 1S1 \ | \ 1S \ | \ \geq$

- Símbolos de entrada (inputs): $1, \geq, \epsilon$
- Símbolos de pila (stack): $Z_0, S, 1, \geq, \epsilon$
- Estados: q = start, p = parsing, f = final
- Estado final: f

Ejemplo 2

Autómata de Pila



Ejemplo 2

- $\delta(p, \epsilon, Z_0) = \{(f, Z_0)\}$

  Al terminar la lectura, pasa al estado de aceptación f sólo si la pila está vacía.

Ejemplo 2

- Transiciones:
  - $\delta(q, \epsilon, Z_0) = \{(p, SZ_0)\}$

    Cambia al estado de “parsing”.

  - $\delta(p, \epsilon, S) = \{(p, 1S1)\}$

    $\delta(p, \epsilon, S) = \{(p, 1S)\}$

    $\delta(p, \epsilon, S) = \{(p, \geq)\}$

    Imitan las reglas de la gramática.

  - $\delta(p, S, S) = \{(p, \epsilon)\}$

    $\delta(p, 1, 1) = \{(p, \epsilon)\}$

    $\delta(p, \geq, \geq) = \{(p, \epsilon)\}$

    Hacen el borrado por cada input leído.

Ejemplo 3

Construir un autómata de pila para la gramática de expresiones aritméticas válidas:

$$\Sigma = \{ \text{int}, +, *, (,) \}$$

Y el lenguaje

$$\text{ARITH} = \{w \in \Sigma^* \mid w \text{ es una exp. aritmética legal}\}$$

Tenemos la gramática:

$$S \rightarrow SS \mid (S) \mid S + S \mid S * S \mid E$$

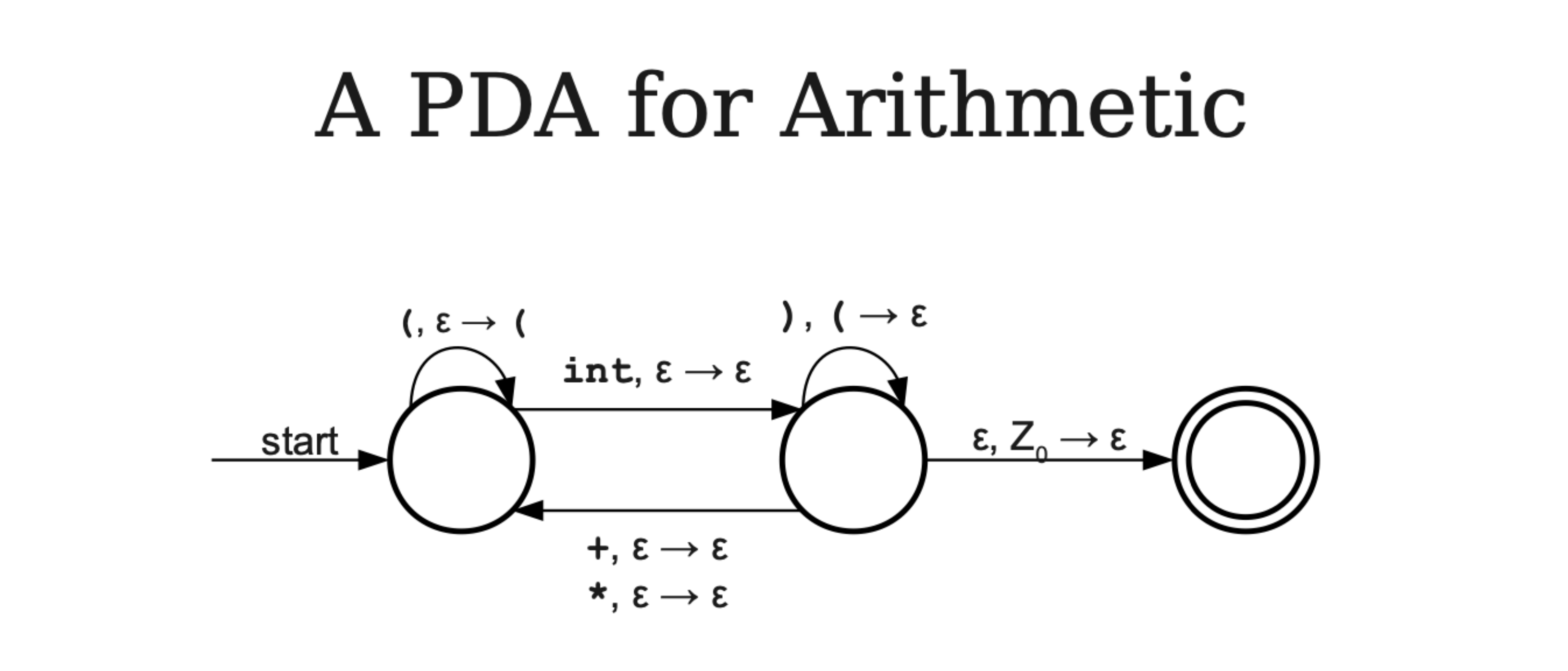
$$E \rightarrow E + E \mid E * E$$

$$E \rightarrow 0 \mid 1 \mid 2 \mid \ldots \mid 9$$

Ejemplo 3

- Símbolos de entrada (inputs): S, E, (,), +, *, int, ϵ
- Símbolos de pila (stack): Z_0, S, E, (,), +, *, int, ϵ
- Estados: ???
- Estado final: ???

Ejemplo 3



¿Por qué los PDA son importantes?

Why PDAs Matter

- Recall: A language is context-free iff there is some CFG that generates it.
- **Important, non-obvious theorem**: A language is context-free iff there is some PDA that recognizes it.
- Need to prove two directions:
  - If L is context-free, then there is a PDA for it.
  - If there is a PDA for L, then L is context-free.
- Part (1) is absolutely beautiful and we'll see it in a second.
- Part (2) is brilliant, but a bit too involved for lecture (you should read this in Sipser).

From CFGs to PDAs

- ***Theorem:*** If G is a CFG for a language L, then there exists a PDA for L as well.
- **Idea:** Build a PDA that simulates expanding out the CFG from the start symbol to some particular string.
- Stack holds the part of the string we haven't matched yet.

From CFGs to PDAs

- Make three states: **start**, **parsing**, and **accepting**.
- There is a transition $\varepsilon, \varepsilon \to$ **S** from **start** to **parsing**.
  - Corresponds to starting off with the start symbol S.
- There is a transition ε, **A** $\to$ $\boldsymbol{\omega}$ from **parsing** to itself for each production **A** $\to$ $\boldsymbol{\omega}$.
  - Corresponds to predicting which production to use.
- There is a transition $\Sigma, \Sigma \to \varepsilon$ from **parsing** to itself.
  - Corresponds to matching a character of the input.
- There is a transition $\varepsilon, Z_0 \to Z_0$ from **parsing** to **accepting**.
  - Corresponds to completely matching the input.

From CFGs to PDAs

- The PDA constructed this way is called a **predict/match parser**.
- Each step either **predicts** which production to use or **matches** some symbol of the input.

From PDAs to CFGs

- The other direction of the proof (converting a PDA to a CFG) is much harder.
- Intuitively, create a CFG representing paths between states in the PDA.
- Lots of tricky details, but a marvelous proof.
  - It's just too large to fit into the margins of this slide.
- Read Sipser for more details.