

5. Autómatas con Pila

Copyright (c) 2025 Adrián Quiroga Linares Lectura y referencia permitidas; reutilización y plagio prohibidos

Concepto Intuitivo: Imagina un Autómata Finito (AFN) que lleva una mochila llena de platos.

- Puede leer la entrada.
- Puede **mirar** el plato de arriba de la pila.
- Dependiendo de lo que ve y lee, puede **quitar platos (pop)** o **poner platos nuevos (push)**.

Esta "mochila" (memoria LIFO) le permite contar y comparar, algo que un autómata finito simple no puede hacer (ej: saber si hay el mismo número de 'a' que de 'b').

5.1 Definición

Un autómata con pila (AP) es un AFN con transiciones \mathcal{E} y con una pila en la que se puede almacenar una cadena de símbolos de pila. El AP puede recordar una cantidad infinita de información. Reconoce Lenguajes Independientes del Contexto.

Un AP se define matemáticamente así:

$$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

Diferencias clave con el autómata finito:

1. Γ (**Alfabeto de Pila**): Símbolos que podemos guardar en la memoria. Puede ser diferente al de entrada (Σ).
2. Z_0 (**Fondo de Pila**): El símbolo que está en la pila antes de empezar nada. Nos avisa de que "la pila está vacía".
3. δ (Función de Transición): La "ley" del movimiento.

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow P(Q \times \Gamma^*)$$

- **Input:** Estado actual + Símbolo entrada (o ε) + **Símbolo en la CIMA de la pila**.
- **Output:** Nuevo estado + **Cadena para reemplazar la cima**.

5.2 Mecánica de Transición (Cómo leer los arcos)

Esta es la parte vital para los ejercicios prácticos. En los diagramas verás arcos con la etiqueta:

$$a, A \rightarrow \gamma$$

Esto se lee: "**Leo, Saco → Meto**".

1. **Leo (a)**: Leo el símbolo 'a' de la cinta de entrada.
2. **Saco (A)**: Compruebo si A está en la **cima** de la pila y lo extraigo (pop).
3. **Meto (γ)**: Escribo la cadena γ en la cima de la pila (push).

Casos Prácticos de "Meto" (γ):

- **Apilar (Push)**: $a, Z_0 \rightarrow AZ_0$
 - *Explicación*: Saco Z_0 y meto A seguido de Z_0 . Efecto neto: A queda encima de Z_0 .
- **Desapilar (Pop)**: $a, A \rightarrow \epsilon$
 - *Explicación*: Saco A y meto... nada (ϵ). Efecto neto: Borro A .
- **Cambiar (Swap)**: $a, A \rightarrow B$
 - *Explicación*: Saco A y meto B .
- **No tocar (Peep)**: $a, A \rightarrow A$
 - *Explicación*: Saco A y vuelvo a meter A . La pila se queda igual.

5.3 Tipos de Aceptación

Un AP puede decir "OK" de dos formas. En los ejercicios te especificarán cuál usar.

Aceptación por Estado Final (F)

- **Condición**: La entrada se ha terminado ($w = \epsilon$) **Y** el autómata está en un estado $q \in F$.
- **La pila**: No importa lo que tenga dentro (puede estar llena de basura).
- **Uso**: Es lo más parecido a los autómatas normales.

Aceptación por Pila Vacía (\emptyset)

- **Condición**: La entrada se ha terminado ($w = \epsilon$) **Y** la pila está totalmente vacía (ni siquiera queda Z_0).
- **El estado**: No importa en qué estado termine.
- **Uso**: Muy común en análisis sintáctico (compiladores).

Conversión: Todo lenguaje aceptado por pila vacía puede ser aceptado por estado final y viceversa. Son equivalentes en poder.

5.4 Protips

Patrón 1: "El Acumulador" (Sumar cosas)

Cuándo usarlo: Ecuaciones tipo $k > i + j$ o $k = i + j$.

Lógica: Tienes dos variables que "suman" y una que "resta".

- **Fase 1 (Entrada 'a')**: Apilas 'a'.
- **Fase 2 (Entrada 'b')**: Sigues apilando (pero ojo, como tienes la restricción, apila 'b' o sigue apilando 'a' si te dejan. Aquí dice "alfabeto de pila = alfabeto entrada", así que apila 'b').
- **Fase 3 (Entrada 'c')**: Desapilas todo. Primero las 'b' y luego las 'a'.
- **Resultado**: Si al acabar de leer 'c' la pila se vacía, eran iguales. Si sobra pila, $i + j$ era mayor. Si falta pila, k era mayor.

Patrón 2: "La Deuda" (El orden da igual / Sopa de letras)

Cuándo usarlo: Ejercicios donde el orden es libre ($N(a) = N(b)$ entrando en cualquier orden) o ecuaciones complejas ($i + k = j + m$).

Lógica: La pila representa el Balance.

- Define dos bandos: **Positivos** (los que suman) y **Negativos** (los que restan).
- **Regla de Oro**:
 - Si leo un símbolo y la pila tiene al del **bando contrario**: **DESAPILO** (se cancelan, como materia y antimateria).
 - Si leo un símbolo y la pila tiene al de **mi bando** o está vacía (Z_0): **APILO** (aumento mi deuda).

Patrón 3: "El 2x1" (Proporciones)

Cuándo usarlo: $N(a) = 2N(b)$ o $3k = i$.

Lógica:

- Opción A (Apilar doble): Por cada 'a' que lees, metes **dos** 'a' en la pila. Luego las 'b' borran de una en una.
- Opción B (Desapilar doble): Metes las 'a' normales. Cuando llegan las 'b', cada 'b' elimina **dos** 'a' de la pila (necesitas un estado intermedio auxiliar para hacer el "doble pop").
- Hay que tener una forma de marcar números negativos también

Patrón 4: "El Multiverso" (La Unión / Ó)

Cuándo usarlo: "Cadenas que cumplen X ó cumplen Y ".

Lógica: El estado inicial (q_0) no lee nada. Lanza dos transiciones λ (o ϵ) hacia dos caminos distintos.

- Camino 1: Resuelve el problema X .
- Camino 2: Resuelve el problema Y .
- El autómata "adivina" qué camino tomar.

Patrón 5: "Álgebra Simple" (Reorganizar ecuaciones)

Cuándo usarlo: Ecuaciones con restas como $k = i - j$ o $k - i < j$.

Truco: Los autómatas odian restar, pero aman sumar. Pasa todo a positivo.

- Si te dan $k = i - j \rightarrow$ Transfórmalo en $i = k + j$.
 - Significa: Las 'a' (i) deben ser iguales a la suma de 'b' (j) + 'c' (k).
 - Estrategia: Apila las 'a'. Las 'b' borran 'a'. Las 'c' borran las 'a' que queden.

5.5 Lema del Bombeo para Lenguajes Independientes del Contexto

Sea L un lenguaje independiente del contexto. Entonces existe una constante n (llamada cte. de bombeo) tal que cualquier cadena z perteneciente a L puede descomponerse en 5 partes, $z = uvwxy$, cumpliendo:

1. $|vwx| \leq n$: La "ventana" donde ocurre el bombeo tiene un tamaño limitado.
2. $vx \neq \epsilon$: Al menos una de las dos partes que se bombean (v o x) debe contener algo (no pueden estar ambas vacías).
3. Para todo $k \geq 0$, la cadena $uv^kwx^ky \in L$: Si repetimos v y x el mismo número de veces (k), la cadena resultante sigue perteneciendo al lenguaje.

Si no cumple el lema, no estamos ante un Lenguaje Independiente del Contexto.

Ejemplo:

$$L = \{a^i b^j c^k \mid k = i/j; i, j, k \geq 1\}$$

Y nos dan como componentes (llaman a $y z$):

- $u = a^{2n-2}$
- $v = a^2$
- $w = \lambda$ (nada)
- $x = b$
- $z = b^{n-1}c^2$

Si usamos $k = 0$ en la expresión uv^kwx^kz , nos queda uwz por lo que tendríamos $a^{2n}b^n c^2$. Además mirando el lenguaje sabemos que se debe cumplir que $N(c) = \frac{N(a)}{N(b)}$. Por lo que podemos sustituir y comprobar si se cumple o no:

$$2 = \frac{2n - 2}{n - 1}$$

Por lo que vemos que cumple el lema (también siguen el orden de abc), no podemos demostrar que no sea un LIC. Y así seguiríamos probando con los k que nos digan.

Aunque a simple vista ya se puede afirmar que no lo será porque los *LIC* no tienen la capacidad de realizar multiplicaciones, solo sumar y contar.