



## UNIVERSIDAD NACIONAL MAYOR DE SAN MARCOS

Facultad de Ingeniería de Sistemas e Informática

# Evaluador y Validador Postfijo con Autómata de Pila (AP)

## Proyecto Final

**Curso:** Teoría de la Computación

**Docente:** Víctor Hugo Bustamante Olivera

**Integrantes:** Diego Sotelo

Alexis Gonzales

Paolo Villavicencio

Álvaro Salazar

**Fecha:** 14 de noviembre de 2025

# Índice

<b>1. Resumen</b>	<b>2</b>
<b>2. Introducción</b>	<b>3</b>
2.1. Descripción del Problema . . . . .	3
2.2. Planteamiento del Proyecto . . . . .	3
2.3. Modelo y Método . . . . .	3
<b>3. Objetivos</b>	<b>4</b>
3.1. Objetivo general . . . . .	4
<b>4. Marco teórico</b>	<b>5</b>
4.1. ¿Qué es un autómata de pila (AP)? . . . . .	5
4.2. Estado de Transiciones de un AP . . . . .	5
4.3. ¿Qué es la notación postfija (RPN) y por qué es adecuada para un AP? . . . . .	5
4.4. Gramática libre de contexto . . . . .	5
<b>5. Desarrollo del programa</b>	<b>6</b>
5.1. Formalización del autómata . . . . .	6
5.2. Criterio de aceptación . . . . .	7
5.3. Representación del proceso . . . . .	7
5.4. Resumen operativo . . . . .	8
5.5. Análisis del funcionamiento . . . . .	8
<b>6. Resultados</b>	<b>9</b>
<b>7. Conclusiones</b>	<b>9</b>

## 1. Resumen

## 2. Introducción

### 2.1. Descripción del Problema

Evaluar expresiones en notación infija es un poco complicado porque hay que pensar en qué operación va primero y en cómo se agrupan los paréntesis. En cambio, la Notación Postfija (RPN) es más directa: primero van los números y después el operador. Esto permite ir resolviendo todo paso a paso usando una pila. El proceso es simple, cada vez que aparece un número, lo metemos en la pila; y cuando aparece un operador, sacamos los dos números de arriba, hacemos la operación y volvemos a guardar el resultado.

El proyecto trata justamente de eso: dado un conjunto de elementos escritos en RPN (números enteros o decimales y los operadores  $+$ ,  $-$ ,  $*$ ,  $/$ ), calcular cuánto vale la expresión y, al mismo tiempo, verificar si está bien escrita. La expresión será válida si al final queda exactamente un valor en la pila y si no pasan errores, como que falten números para operar o intentar dividir entre cero.

### 2.2. Planteamiento del Proyecto

Se implementará un simulador de Autómata de Pila (AP). La idea es que el programa reciba una lista de tokens ya identificados y vaya usando la pila de la forma básica: cuando llega un número, lo ponemos en la pila (push); cuando aparece un operador que necesita dos valores, sacamos los dos de arriba (pop), hacemos la operación y guardamos el resultado otra vez.

El AP aceptará la expresión siempre que esté bien escrita en notación postfija y que, al terminar de leer todos los tokens, la pila quede con un único valor. Si ocurre algo extraño, como que falten números para operar o que queden valores de más en la pila, entonces la expresión se rechaza. Además, el sistema debe ser capaz de detectar errores como intentar dividir entre cero.

### 2.3. Modelo y Método

**Modelo.** Para este proyecto usamos un Autómata de Pila (AP) descrito de forma general como una séptupla, pero en palabras simples es solo un sistema con estados, una entrada (los números y operadores), y una pila donde vamos guardando valores. El estado inicial es donde empieza todo, y el estado de aceptación es donde el AP determina que la expresión está correcta. Mediante el proceso la pila usa operaciones básicas de *push* y *pop* para recorrer toda la expresión.

**Método.** El algoritmo recorre los tokens una sola vez. Cuando aparece un número, simplemente lo metemos a la pila (*push*). Cuando aparece un operador como  $+$ ,  $-$ ,  $*$  o  $/$ , el programa revisa si hay por lo menos dos valores en la pila; si los hay, saca ambos (*pop*), hace la operación y vuelve a meter el resultado. Si falta algún valor para operar, se rechaza la expresión. Al terminar de leer todos los tokens, solo se acepta si queda exactamente un valor en la pila. Además, si el operador es una división y el divisor es cero, se detiene todo y se reporta un error. El proceso completo toma tiempo lineal respecto al tamaño de la expresión y usa una cantidad de memoria proporcional a lo que llegue a crecer la pila.

## 3. Objetivos

### 3.1. Objetivo general

Desarrollaremos un programa que sea capaz de evaluar y validar expresiones que estén en notación postfija, nos basaremos en un autómata de pila que sea capaz de calcular el resultado solo si la expresión es válida, en caso de que no, la expresión sería rechazada, además colocaremos los pasos y detalles del proyecto en nuestra página de github en el README.md

### Objetivos específicos

## 4. Marco teórico

4.1. ¿Qué es un autómata de pila (AP)?

4.2. Estado de Transiciones de un AP

4.3. ¿Qué es la notación postfija (RPN) y por qué es adecuada para un AP?

4.4. Gramática libre de contexto

## 5. Desarrollo del programa

En el desarrollo del programa se explica cómo el modelo implementado logra evaluar expresiones en notación postfija usando un autómata de pila. Básicamente, se muestra cómo la definición formal del autómata se convierte en un proceso que realmente funciona, siguiendo la lógica de cada transición. Así se puede ver, de manera sencilla, cómo cada regla teórica termina siendo una acción concreta sobre la pila.

### 5.1. Formalización del autómata

El autómata de pila diseñado se expresa mediante la séptupla:

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

donde:

$Q = \{q_1\}$	(conjunto de estados)
$\Sigma = \{V, +, -, *, /\}$	(alfabeto de entrada)
$\Gamma = \{X, Z_0\}$	(alfabeto de pila)
$q_0 = q_1$	(estado inicial)
$Z_0$ símbolo de fondo de pila	
$F = \emptyset$	(aceptación por pila vacía)

El funcionamiento completo está definido por la función de transición:

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

que básicamente indica qué apila o qué desapila según el símbolo leído y lo que haya en la pila:

- $\delta(q_1, \varepsilon, Z_0) = (q_1, Z_0)$  Inicialización.
- $\delta(q_1, V, Z_0) = (q_1, XZ_0)$  Primer operando apilado.
- $\delta(q_1, V, X) = (q_1, XX)$  Apila operandos sucesivos.
- $\delta(q_1, +, XX) = (q_1, X)$  Suma: desapila dos, apila uno.
- $\delta(q_1, -, XX) = (q_1, X)$  Resta: desapila dos, apila uno.
- $\delta(q_1, *, XX) = (q_1, X)$  Multiplica: desapila dos, apila uno.
- $\delta(q_1, /, XX) = (q_1, X)$  Divide: desapila dos, apila uno.
- $\delta(q_1, \varepsilon, XZ_0) = (q_1, \varepsilon)$  Pila vacía: aceptación.

## 5.2. Criterio de aceptación

El autómata acepta una expresión siempre y cuando, al terminar de leer toda la entrada, la pila quede completamente vacía (es decir, ya no quede ni el símbolo  $Z_0$ ):

$$\delta^*(q_1, w, Z_0) = (q_1, \varepsilon), \quad w \in L(M)$$

Esto asegura que:

- Todos los operadores encontraron sus operandos.
- No quedaron operandos sin usar al finalizar.

## 5.3. Representación del proceso

Para mostrar cómo funciona todo el mecanismo, el proceso de evaluación se describe como una secuencia de configuraciones del autómata:

$$(q, \text{entrada}, \text{pila}) \Rightarrow^* (q', \varepsilon, \varepsilon)$$

Por ejemplo, para evaluar la expresión:

$$3 \ 4 \ + \ 2 \ *$$

el autómata avanza así:

$$\begin{aligned} (q_1, 3 \ 4 \ + \ 2 \ *, Z_0) &\Rightarrow (q_1, 4 \ + \ 2 \ *, XZ_0) \Rightarrow (q_1, + \ 2 \ *, XXZ_0) \\ &\Rightarrow (q_1, 2 \ *, XZ_0) \Rightarrow (q_1, *, XXZ_0) \Rightarrow (q_1, \varepsilon, \varepsilon) \end{aligned}$$

que muestra claramente cómo la pila va creciendo con operandos y reduciéndose cada vez que aparece un operador.

## 5.4. Resumen operativo

El comportamiento del autómata se resume de forma sencilla así:

- **Si se lee un operando:** se apila un símbolo  $X$ .
- **Si se lee un operador:** se desapilan dos  $X$  y se vuelve a apilar uno.
- **Si se llega al final con pila  $XZ_0$ :** la cadena se acepta.

Cuadro 1: Resumen equivalente de las transiciones del autómata

#	Entrada	Cima de pila	Acción	Nuevo contenido
1	$\varepsilon$	$Z_0$	Inicializa la pila	$Z_0$
2	$V$	$Z_0$	Guarda primer operando	$XZ_0$
3	$V$	$X$	Guarda nuevo operando	$XX$
4	$+$	$XX$	Suma (pop 2, push 1)	$X$
5	$-$	$XX$	Resta (pop 2, push 1)	$X$
6	$*$	$XX$	Multiplicación (pop 2, push 1)	$X$
7	$/$	$XX$	División (pop 2, push 1)	$X$
8	$\varepsilon$	$XZ_0$	Vacía la pila	$\varepsilon$

**Interpretación:** El autómata reproduce la lógica típica de las expresiones postfijas: cada operando hace crecer la pila y cada operador binario la reduce. Cuando la pila queda vacía al final, significa que la expresión estaba correctamente formada y pudo evaluarse sin problemas.

## 5.5. Análisis del funcionamiento

Finalmente, los resultados muestran que el programa se comporta justo como se esperaba del autómata de pila que se diseñó. Cada transición  $\delta$  termina convirtiéndose directamente en una acción de apilar o desapilar, y el criterio de aceptación funciona tal como fue pensado: al final, la pila tiene que quedar vacía para asegurar que la expresión se procesó correctamente.

## 6. Resultados

## 7. Conclusiones