



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: 11 de noviembre de 2025

Índice

1. Resumen	2
2. Introducción	3
2.1. Descripción del Problema	3
2.2. Planteamiento del Proyecto	3
2.3. Modelo y Método	3
3. Objetivos	4
3.1. Objetivo general	4
4. Marco teórico	5
4.1. Qué es un autómata de pila (AP)	5
4.2. Estado de Transiciones de un AP	6
4.3. Qué es la notación postfija (RPN) y por qué es adecuada para un AP	8
4.4. Gramática libre de contexto	9
4.4.1. Definición Formal	9
4.4.2. Reglas de Producción	9
4.4.3. Derivaciones	10
4.4.4. Lenguaje Generado	10
5. Desarrollo del programa	11
5.1. Formalización del autómata	11
5.2. Criterio de aceptación	12
5.3. Representación del proceso	12
5.4. Resumen operativo	12
5.5. Análisis del funcionamiento	13
6. Resultados	13
7. Conclusiones	14
8. Referencias bibliográficas	15

1. Resumen

Este proyecto presenta el diseño e implementación de un evaluador y validador de expresiones aritméticas en notación postfixa (RPN) utilizando un Autómata de Pila (AP) como modelo formal. Se parte de la definición teórica del AP y de las gramáticas libres de contexto para modelar el proceso de evaluación mediante operaciones de *push* y *pop* sobre la pila, estableciendo criterios formales de aceptación y rechazo de cadenas. El sistema procesa una secuencia de tokens (operando numéricos y operadores binarios $+, -, *, /$), calcula el resultado cuando la expresión es sintácticamente válida y detecta errores como operandos insuficientes o división entre cero. Finalmente, se documenta la formalización del autómata, su comportamiento de ejecución y se discute cómo este enfoque ilustra la relación entre teoría de autómatas y evaluación de expresiones aritméticas en contextos computacionales.

Palabras clave: Autómata de pila, Notación postfixa, RPN, Evaluación de expresiones, Teoría de la computación.

Abstract

This project presents the design and implementation of an evaluator and validator for arithmetic expressions in postfix notation (RPN) using a Pushdown Automaton (PDA) as the formal model. The work starts from the theoretical definition of PDAs and context-free grammars to model the evaluation process through *push* and *pop* operations on the stack, establishing formal acceptance and rejection criteria for input strings. The system processes a sequence of tokens (numeric operands and binary operators $+, -, *, /$), computes the result when the expression is syntactically valid, and detects errors such as insufficient operands or division by zero. Finally, the formalization of the automaton, its runtime behavior, and how this approach illustrates the relationship between automata theory and the evaluation of arithmetic expressions in computational settings are discussed.

Keywords: Pushdown automaton, Postfix notation, RPN, Expression evaluation, Theory of computation.

2. Introducción

2.1. Descripción del Problema

La evaluación y validación de expresiones aritméticas en notación infija exige tratar precedencias y paréntesis. En cambio, la Notación Postfija (RPN) coloca el operador después de sus operandos y permite procesar la expresión de manera lineal con una pila: se apilan operandos y, al aparecer un operador binario, se desapilan los dos valores superiores, se aplica la operación y se vuelve a apilar el resultado. El problema que aborda este proyecto es: dada una lista de tokens en RPN (enteros/decimales y operadores $+, -, *, /$), calcular el resultado numérico y, simultáneamente, decidir si la cadena es sintácticamente válida. La validez se determina por el tamaño final de la pila (exactamente un elemento) y por la ausencia de errores como operandos insuficientes o división entre cero.

2.2. Planteamiento del Proyecto

Se implementará un simulador de Autómata de Pila (AP) que lea una lista de tokens ya clasificados y ejecute las acciones de pila estándar: *push* para operandos; ante un operador binario, dos *pop*, cómputo y *push* del resultado. La aceptación ocurre si, tras consumir todos los tokens, la pila queda con un único valor; en caso contrario se rechaza. Además, el sistema debe manejar explícitamente el error aritmético por división entre cero. La documentación incluirá el diseño del AP, reglas de transición, criterios de aceptación/rechazo y una batería mínima de pruebas en un `README.md`.

2.3. Modelo y Método

Modelo. Usaremos el AP como séptupla $AP = \langle Q, \Sigma, \Gamma, q_0, Z_0, A, \delta \rangle$, donde Q es el conjunto de estados; Σ el alfabeto de entrada (números enteros/decimales y operadores); Γ el alfabeto de pila (valores numéricos y Z_0); q_0 el estado inicial; A el conjunto de estados de aceptación; y δ la función de transición con operaciones *push/pop*. **Método.** El algoritmo recorre la cadena una sola vez: (i) si el token es un número x , se hace $push(x)$; (ii) si el token es un operador $op \in \{+, -, *, /\}$, se comprueba que haya al menos dos elementos en la pila; de ser así se realizan dos *pop*, se computa $r = o_1 op o_2$ y se hace *push*(r); si falta algún operando, se rechaza; (iii) al final de la entrada, se acepta si y sólo si la pila contiene exactamente un elemento; (iv) si op es división y el divisor es cero, se reporta error aritmético. La complejidad es $O(n)$ en tiempo y $O(k)$ en espacio (altura máxima de la pila).

3. Objetivos

3.1. Objetivo general

Desarrollar un evaluador y validador de expresiones en notación postfija basado en Autómata de Pila, capaz de calcular el resultado cuando la expresión es válida y de rechazar formalmente entradas inválidas (operando insuficientes, tamaño final de pila distinto de uno y división entre cero); además, documentar el diseño, la implementación y la evidencia experimental en el `README.md`.

Objetivos específicos

- Formalizar, a partir de la teoría de autómatas de pila y gramáticas libres de contexto, el modelo que representa la evaluación de expresiones aritméticas en notación postfija (RPN).
- Diseñar e implementar un autómata de pila capaz de validar sintácticamente expresiones postfijas con operandos numéricos y operadores binarios ($+, -, *, /$), así como de calcular su resultado.
- Detectar y reportar errores frecuentes en las expresiones de entrada (operandos insuficientes, símbolos no válidos, división entre cero, entre otros) dentro del proceso de evaluación.
- Desarrollar un módulo computacional interactivo que integre la lógica del autómata de pila y permita al usuario ingresar expresiones y observar su procesamiento paso a paso.
- Definir y aplicar un conjunto de casos de prueba que permitan verificar la corrección y el comportamiento del evaluador/validador, documentando los resultados obtenidos.

4. Marco teórico

4.1. Qué es un autómata de pila (AP)

Un autómata de pila es un modelo matemático de una máquina con un estado finito y una memoria auxiliar en forma de pila (stack), que opera siguiendo una lógica de "último en entrar, primero en salir"(LIFO). Se utiliza para reconocer lenguajes libres de contexto. A diferencia de los autómatas finitos, estos pueden procesar y aceptar cadenas más complejas al tener la capacidad de almacenar y recuperar símbolos de su pila.

Definición de un Autómata de pila

$$\text{APD} = \langle E, A, P, \delta, e_0, Z_0, F \rangle$$

E : Conjunto finito de estados,

A : Alfabeto o conjunto finito de símbolos de la cinta de entrada,

P : Alfabeto o conjunto finito de símbolos de la pila. $P \cap A = \emptyset$

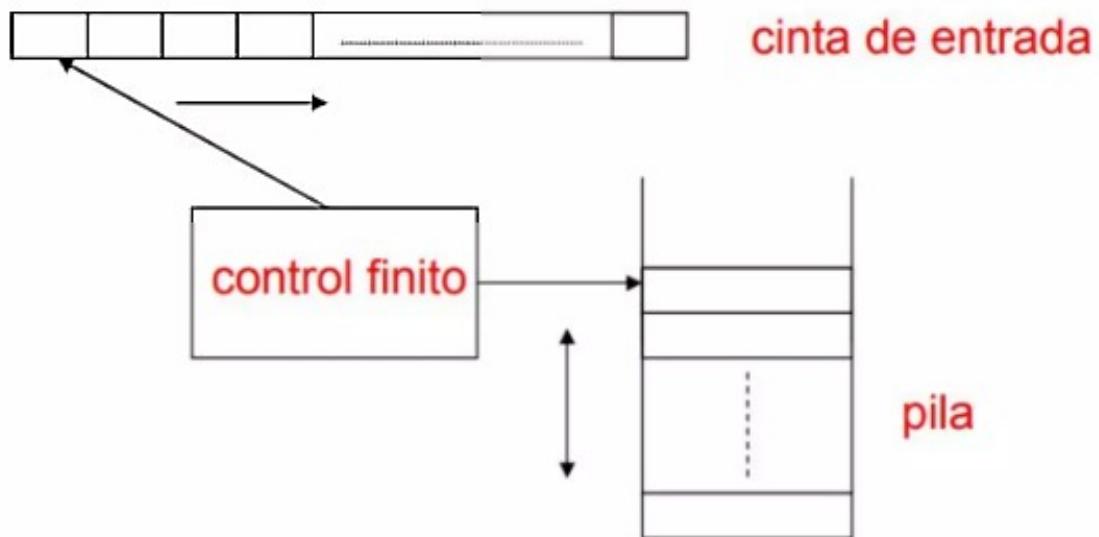
δ : función de transición de estados

e_0 : Estado inicial $e_0 \in E$.

Z_0 : Símbolo distinguido $Z_0 \in P$

F : Conjunto de estados finales o estados de aceptación. $F \subseteq E$.

Representación gráfica:



¿Cómo una palabra es aceptada?

Para que una palabra de entrada sea aceptada en un AP se deben cumplir todas las condiciones siguientes:

1. La palabra de entrada se debe haber agotado (consumido totalmente).
2. El AP se debe encontrar en un estado final.
3. La pila debe estar vacía.

4.2. Estado de Transiciones de un AP

En un autómata de pila, la *función de transición* se define como

$$\delta : E \times (A \cup \{\varepsilon\}) \times P \rightarrow E \times P^*,$$

donde:

- E es el conjunto de estados.
- A es el alfabeto de entrada.
- P es el alfabeto de la pila.
- P^* es el conjunto de todas las cadenas (posiblemente vacías) de símbolos de pila.
- ε representa la transición sin consumir símbolo de entrada.

Una transición se suele escribir como

$$\delta(q, a, X) = (p, \gamma),$$

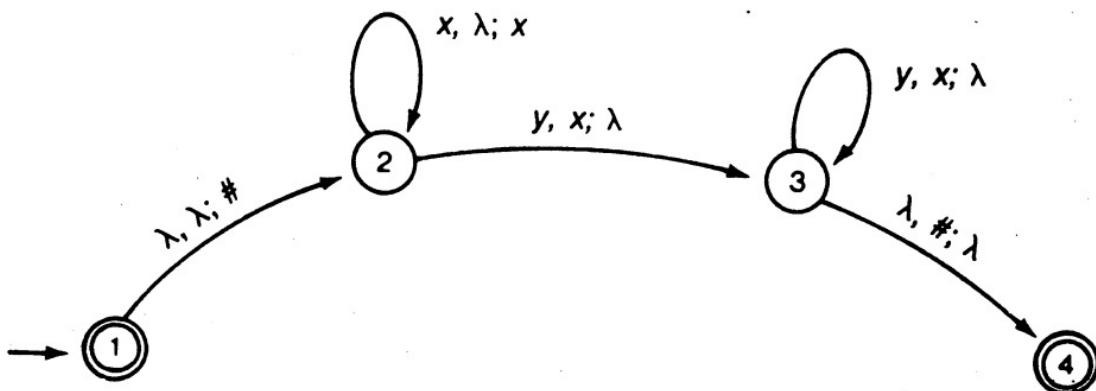
y se interpreta así:

- El autómata está en el estado q .
- Lee el símbolo a de la cinta de entrada (o ε si no consume nada).
- Observa el símbolo X en la cima de la pila.
- Cambia al estado p y reemplaza X por la cadena γ en la pila (es decir, desapila X y apila los símbolos de γ ; si $\gamma = \varepsilon$, simplemente desapila X).

De manera general, una transición puede:

- **Apilar** símbolos (cuando γ contiene más símbolos que X).
- **Desapilar** símbolos (cuando $\gamma = \varepsilon$).
- **Reemplazar** el símbolo de la cima por otros (cuando γ tiene uno o varios símbolos distintos de X).
- **No consumir entrada** (cuando $a = \varepsilon$), modificando solo el estado y/o la pila.

Ejemplo de diagrama de transiciones:



Estado actual	Símbolo entrada	Símbolo pila	Nuevo estado	Operación en la pila
1	λ	λ	2	apilar $\#$
2	x	λ	2	apilar x
2	y	x	3	desapilar x
3	y	x	3	desapilar x
3	λ	$\#$	4	desapilar $\#$

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

Una expresión aritmética en notación postfija (o notación polaca inversa) es una secuencia formada por símbolos de dos tipos diferentes: operadores (para simplificar, consideraremos únicamente los operadores aritméticos binarios +, -, * y /) y operandos (para simplificar pensaremos en identificadores de una sola letra). Cada operador se escribe detrás de sus operandos.

Por ejemplo, a la expresión siguiente, escrita en la notación habitual (infija):

$$a*b/c$$

Le corresponde la siguiente expresión en notación postfija:

$$ab*c/$$

Si en la expresión infija aparecen paréntesis, estos cambian la correspondiente expresión postfija sólo si los paréntesis alteran el orden de prioridad de los operadores:

$a/b+c*d-e*f$, traducido a notación postfija es: $ab/cd*+ef*-$

$a/(b+c)*(d-e)*f$, se traduce en cambio por: $abc+/de-*f*$

Tres ventajas importantes de la notación postfija frente a la convencional infija son las siguientes:

- En notación postfija nunca son necesarios los paréntesis.
- En notación postfija no es necesario definir prioridades entre operadores.
- Una expresión postfija puede evaluarse de forma muy sencilla.

¿Porqué es adecuada para un AP?

La razón por la cual la notación postfija es adecuada para un Autómata de Pila es gracias a que su estructura elimina la necesidad de manejar paréntesis y prioridades complejas, y se basa directamente en operaciones de pila (push y pop). Esto permite que el autómata modele de forma sencilla la verificación de expresiones postfijas válidas, utilizando únicamente su memoria tipo pila para controlar la forma y la consistencia de la expresión analizada.

4.4. Gramática libre de contexto

4.4.1. Definición Formal

Una **Gramática Libre de Contexto** (GLC), o G , se define formalmente como una 4-tupla:

$$G = (V, \Sigma, P, S)$$

Donde:

- V es un conjunto finito de **variables** (o símbolos no terminales).
- Σ es un conjunto finito de **símbolos terminales** (el alfabeto).
- Se requiere que V y Σ sean disjuntos: $V \cap \Sigma = \emptyset$.
- P es un conjunto finito de **reglas de producción** (o producciones).
- S es el **símbolo inicial** (o axioma), y debe pertenecer al conjunto de variables: $S \in V$.

4.4.2. Reglas de Producción

El término "libre de contexto" se refiere a la forma de las reglas en P . Cada regla debe tener la siguiente estructura:

$$A \rightarrow \alpha$$

Donde:

- A debe ser una **única variable**, es decir, $A \in V$.
- α es una cadena de cero o más símbolos, donde cada símbolo puede ser una variable o un terminal. Formalmente, $\alpha \in (V \cup \Sigma)^*$.

El asterisco (*) denota la **clausura de Kleene**, lo que significa cero o más "símbolos". Esto incluye la cadena vacía (denotada como ϵ o λ). Una regla de la forma $A \rightarrow \epsilon$ es una **producción épsilon** (o producción vacía).

La restricción clave es que el lado izquierdo de la regla (A) es *siempre* una única variable, sin importar el contexto (los símbolos que la rodean).

4.4.3. Derivaciones

Las reglas de producción se utilizan para generar cadenas mediante un proceso llamado **derivación**.

- **Derivación en un paso (\Rightarrow)**

Sean β y γ cadenas cualesquiera en $(V \cup \Sigma)^*$. Si existe una regla $A \rightarrow \alpha \in P$, entonces decimos que la cadena $\beta A \gamma$ deriva en un paso a la cadena $\beta \alpha \gamma$.

Esto se nota como:

$$\beta A \gamma \Rightarrow \beta \alpha \gamma$$

- **Derivación en cero o más pasos (\Rightarrow^*)**

La notación \Rightarrow^* representa la **clausura reflexiva y transitiva** de la derivación en un paso (\Rightarrow).

- **Reflexiva:** Para cualquier cadena α , $\alpha \Rightarrow^* \alpha$ (una cadena siempre deriva a sí misma en cero pasos).
- **Transitiva:** Si $\alpha \Rightarrow \beta$ y $\beta \Rightarrow^* \gamma$, entonces $\alpha \Rightarrow^* \gamma$.

Básicamente, $\alpha \Rightarrow^* \beta$ significa que β puede obtenerse a partir de α aplicando cero o más reglas de producción.

4.4.4. Lenguaje Generado

El **lenguaje generado** por una gramática G , denotado como $L(G)$, es el conjunto de todas las cadenas de *terminales* que pueden ser derivadas a partir del símbolo inicial S .

Formalmente:

$$L(G) = \{w \in \Sigma^* \mid S \Rightarrow^* w\}$$

- $w \in \Sigma^*$ significa que w es una cadena que consiste *exclusivamente* en símbolos terminales.
- $S \Rightarrow^* w$ significa que w se puede derivar desde S en cero o más pasos.

Cualquier lenguaje que pueda ser generado por una Gramática Libre de Contexto se denomina **Lenguaje Libre de Contexto (LLC)**.

5. Desarrollo del programa

En este capítulo se presenta el desarrollo conceptual del modelo de evaluación postfija, basado en el autómata de pila definido en el marco teórico. Se describe la traducción de la definición formal a una estructura operativa, conservando la semántica matemática de cada transición δ .

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 comportamiento del autómata está determinado por la función de transición $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow Q \times \Gamma^*$, que define las acciones de apilado y desapilado en función del símbolo de entrada y del contenido de 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 si, tras consumir completamente la cadena de entrada, la pila queda vacía (es decir, solo se ha eliminado el símbolo de fondo Z_0):

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

Esto implica que:

- Cada operador ha encontrado suficientes operandos en la pila.
- No quedan operandos sin combinar al finalizar la lectura.

5.3. Representación del proceso

De manera general, el proceso de evaluación de una expresión postfija puede representarse como una secuencia de configuraciones del autómata:

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

Ejemplo para la expresión:

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

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

5.4. Resumen operativo

El comportamiento del autómata puede resumirse en las siguientes reglas:

- **Lectura de operando:** se apila un símbolo X .
- **Lectura de operador:** se desapilan dos símbolos X y se apila uno nuevo.
- **Lectura vacía con pila XZ_0 :** se acepta la cadena.

Cuadro 1: Resumen equivalente de las transiciones del autómata

#	Entrada	Cima de pila	Acción	Nuevo contenido
1	ε	Z_0	Inicializa pila	Z_0
2	V	Z_0	Apila primer operando	XZ_0
3	V	X	Apila 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	ε	XZ_0	Vacía pila	ε

Interpretación: El autómata de pila simula exactamente la lógica de evaluación de una expresión postfija: cada operando incrementa la altura de la pila y cada operador binario reduce su tamaño en uno. El estado final de aceptación representa el momento en que la pila se vacía, confirmando que la expresión fue correctamente balanceada y evaluada.

5.5. Análisis del funcionamiento

Los resultados muestran que el programa reproduce fielmente la semántica del autómata de pila definido teóricamente. Cada transición δ se traduce directamente en una operación de apilado o desapilado, y las condiciones de aceptación coinciden con las definidas formalmente: una única celda en la pila al finalizar la lectura.

6. Resultados

La implementación del autómata de pila permitió evaluar y validar correctamente un conjunto representativo de expresiones aritméticas en notación postfija, incluyendo casos simples y expresiones de mayor longitud. En las pruebas realizadas, el sistema mostró consistencia entre los resultados obtenidos y los valores esperados calculados manualmente o con herramientas de referencia, lo que respalda la corrección del modelo formal y de la lógica de evaluación. Además, la detección de errores como operandos insuficientes, símbolos no válidos o intentos de división entre cero se ejecutó de forma adecuada, generando mensajes claros para el usuario.

Desde el punto de vista didáctico, la visualización paso a paso del contenido de la pila y de las transiciones de estado favoreció la comprensión del funcionamiento interno del autómata de pila. Esto permitió evidenciar cómo las operaciones de *push* y *pop* se relacionan directamente con la estructura de las expresiones postfijas, reforzando el vínculo entre la teoría de la computación y su aplicación práctica. No obstante, el sistema se centró en un subconjunto limitado de operadores y tipos de datos, lo que abre la posibilidad de extender el modelo a escenarios más complejos.

7. Conclusiones

Se concluye que el uso de un autómata de pila como base para la evaluación de expresiones aritméticas en notación postfija es una estrategia válida y efectiva, tanto desde el punto de vista teórico como práctico. La implementación desarrollada demuestra que los conceptos de gramáticas libres de contexto y autómatas de pila pueden materializarse en un sistema funcional capaz de validar expresiones, calcular resultados y gestionar errores de manera controlada.

Asimismo, la integración de una interfaz interactiva y de mecanismos de visualización del proceso de evaluación contribuyó significativamente al objetivo académico del proyecto, al facilitar la comprensión de los conceptos de la asignatura de Teoría de la Computación. Finalmente, se reconoce que el trabajo puede ampliarse incorporando nuevos operadores, soporte para diferentes tipos de datos y optimizaciones en la representación visual, lo que permitiría explorar aplicaciones aún más cercanas a entornos reales de programación.

8. Referencias bibliográficas

- Autómatas de pila. (2008). *Universidad Nacional del Centro de la Provincia de Buenos Aires*. <https://users.exa.unicen.edu.ar/catedras/ccomp1/Apunte4.pdf>
- Borbón Alpízar, A. (2006). ¿Cómo evaluar expresiones matemáticas en el computador? *Revista Digital: Matemática, Educación e Internet*, 7(2), 1–23. <https://www.redalyc.org/pdf/6079/607972904002.pdf>
- Coordinación de Ciencias Computacionales, INAOE. (s.f.). Gramáticas libres de contexto y lenguajes *Apuntes del Curso Propedéutico: Teoría de Autómatas y Lenguajes Formales* https://posgrados.inaoep.mx/archivos/PosCsComputacionales/Curso_Propedeutico/Automatas/05_Automatas_GramaticasLibresContextoLenguajes/CAPTUL1.PDF
- Ejemplos de aplicación de pilas. (s. f.). *Universidad de Zaragoza*. https://webdiis.unizar.es/asignaturas/EDA/varios/pilas_colas/ejemplos_aplicacion_pilas.pdf
- Gutiérrez Giraldi, O., & Martínez Moreno, M. (2025). Algoritmo de conversión de una gramática libre de contexto a un autómata de pila. *XIKUA Boletín Científico de la Escuela Superior de Tlahuelilpan*, 13(25), 31–38. <https://doi.org/10.29057/xikua.v13i25.13821>
- Juárez Fuentes, J. (s. f.). Autómata de pila. *Universidad Tecnológica de la Mixteca*. https://www.utm.mx/~jjf/tc/3.1%20AUTOMATA_DE_PILA.pdf
- Rincón, L. (s. f.). Autómatas de pila y lenguajes independientes del contexto. *Alfaomega*. https://libroweb.alfaomega.com.mx/book/685/free/ovas_statics/cap9/Automatas%20de%20Pila%20y%20Lenguajes%20independientes%20del%20contexto.%20Rincon,%20Luis.pdf