ACTIVIDAD LABORATORIO NO.2
AUTÓMATA A PILA CON JFLAP

PRESENTADO POR:
ALEJANDRO DE MENDOZA TOVAR

PRESENTADO AL PROFESOR:
ING ROGERIO ORLANDO BELTRAN CASTRO
PROFESOR

FUNDACIÓN UNIVERSITARIA INTERNACIONAL DE LA RIOJA
FACULTAD DE INGENIERÍA – INGENIERIA INFORMÁTICA
BOGOTÁ D.C.
22 DE NOVIEMBRE
2025

TABLA DE CONTENIDO

RESUMEN

Para el desarrollo del Laboratorio: Autómatas a Pila con JFLAP, se implementaron dos ejercicios prácticos orientados al diseño, construcción y validación de autómatas a pila (PDA - Pushdown Automata), con el fin de comprender su funcionamiento en el reconocimiento de lenguajes libres de contexto que no pueden ser procesados mediante autómatas finitos.

El objetivo principal del laboratorio fue construir dos autómatas a pila capaces de reconocer los lenguajes $L_1 = \{X^nY^{2n} : n > 0\}$ y $L_2 = \{X^{2n}Y^n : n \geq 0\}$, así como analizar su comportamiento mediante la prueba de cadenas aceptadas y rechazadas en el entorno de simulación JFLAP. Estos lenguajes presentan dependencias entre símbolos que requieren el uso de una estructura de datos tipo pila para su reconocimiento, característica fundamental que distingue a los autómatas a pila de los autómatas finitos.

Durante el proceso, se construyeron ambos autómatas utilizando JFLAP en su modalidad "Pushdown Automaton - Multiple Character Input", herramienta que permitió observar de manera visual y dinámica tanto las transiciones de estado como las operaciones de apilamiento (push) y desapilamiento (pop) ante distintas cadenas de entrada. Para el primer autómata (X^nY^{2n}), se implementó una estrategia que apila un símbolo por cada X leída y desapila ese símbolo por cada dos Y's consecutivas, requiriendo cinco estados y un mecanismo de conteo de pares de Y's. Para el segundo autómata ($X^{2n}Y^n$), se diseñó un enfoque que acepta la cadena vacía y apila un símbolo por cada par de X's leídas, desapilando luego un símbolo por cada Y, lo cual garantiza la relación 2:1 entre X's e Y's.

Se elaboró un conjunto exhaustivo de pruebas para cada autómata, verificando experimentalmente su comportamiento con cadenas válidas como XYY, XXYYYY, XXXYYYYYY para el primer lenguaje, y ε (cadena vacía), XXY, XXXXYY para el segundo lenguaje. Asimismo, se validó el rechazo correcto de cadenas inválidas como XY, XYYY y XXY para el primer caso, y X, XY, XXXYY para el segundo. Este análisis permitió comprobar la validez del diseño de ambos autómatas y afianzar la relación entre teoría y práctica dentro del campo de los lenguajes formales y la teoría de la computación.

El laboratorio favoreció el entendimiento de conceptos fundamentales como estados, transiciones con operaciones de pila, símbolos de entrada, símbolos de pila (stack alphabet), símbolo inicial de pila (Z), aceptación por estado final, y el uso de transiciones épsilon (λ) para validación final. Se reforzó la capacidad para diseñar estrategias de apilamiento y desapilamiento que garanticen el balance correcto entre diferentes símbolos de entrada, competencia esencial en el análisis de lenguajes libres de contexto.

Durante el desarrollo se presentaron desafíos técnicos importantes relacionados con la configuración correcta de JFLAP, específicamente en la selección del tipo de autómata apropiado ("Multiple Character Input" vs "Single Character Input"), el formato exacto de las transiciones, y la comprensión de las restricciones en las operaciones de pila. Estos obstáculos, una vez superados, fortalecieron significativamente la comprensión práctica del funcionamiento de los autómatas a pila.

Finalmente, se concluye que el uso de JFLAP constituye una herramienta pedagógica de gran utilidad para el aprendizaje de los fundamentos de la teoría de autómatas a pila y lenguajes libres de contexto. El ejercicio permitió no solo afianzar conocimientos sobre los mecanismos de reconocimiento de cadenas con dependencias estructurales, sino también desarrollar competencias analíticas en el diseño de autómatas que utilizan memoria auxiliar (pila) para procesar lenguajes más complejos que los regulares, habilidades esenciales en el ámbito de la ingeniería informática, el diseño de compiladores y el análisis sintáctico.

1. INTRODUCCIÓN

El presente laboratorio tiene como finalidad profundizar en la comprensión de los fundamentos teóricos y prácticos de los autómatas a pila y los lenguajes libres de contexto, empleando el programa JFLAP como entorno de simulación y análisis. A través de esta herramienta, se busca fortalecer la capacidad del estudiante para modelar, representar y analizar formalmente lenguajes que requieren memoria auxiliar mediante autómatas a pila, superando las limitaciones de los autómatas finitos en el reconocimiento de dependencias estructurales.

En este contexto, JFLAP se presenta como un entorno didáctico y visual que permite experimentar directamente con la construcción de autómatas a pila (PDA - Pushdown Automata), observando en tiempo real las operaciones de apilamiento y desapilamiento que caracterizan a estos modelos computacionales. Estas representaciones constituyen elementos fundamentales en la teoría de la computación, ya que permiten comprender cómo las máquinas pueden reconocer y procesar cadenas con dependencias anidadas o balanceadas de acuerdo con reglas bien definidas, capacidad esencial para el análisis sintáctico en compiladores.

El desarrollo del laboratorio se centra específicamente en el diseño e implementación de dos autómatas a pila que reconocen lenguajes con diferentes relaciones de dependencia entre símbolos, a partir de los cuales se derivan tres tareas principales:

- Diseñar un autómata a pila que reconozca el lenguaje $L_1 = \{X^nY^{2n} : n > 0\}$, donde cada X debe estar seguida por exactamente el doble de Y's.
- Construir un autómata a pila que reconozca el lenguaje $L_2 = \{X^{2n}Y^n : n \geq 0\}$, donde debe haber exactamente el doble de X's que de Y's, incluyendo la cadena vacía.
- Elaborar conjuntos exhaustivos de cadenas aceptadas y rechazadas para cada autómata, validando los resultados mediante la simulación paso a paso en JFLAP.

Este enfoque combina la aplicación de los principios teóricos de la teoría de autómatas y lenguajes libres de contexto con el uso de herramientas computacionales, permitiendo comprobar experimentalmente cómo la incorporación de una estructura de datos tipo pila expande significativamente el poder computacional respecto a los autómatas finitos. De esta manera, el estudiante no solo observa el comportamiento del autómata y su pila, sino que también desarrolla la capacidad de diseñar estrategias de reconocimiento basadas en balance y conteo —una habilidad clave en el diseño de analizadores sintácticos y sistemas de validación de estructuras anidadas.

El laboratorio permite, además, comprender la importancia de los autómatas a pila como modelo intermedio entre los autómatas finitos (que reconocen lenguajes regulares) y las máquinas de Turing (que reconocen lenguajes recursivamente enumerables). Se evidencia cómo la capacidad de memoria ilimitada pero restringida (LIFO - Last In, First Out) de la pila permite reconocer lenguajes libres de contexto que son esenciales en el análisis de lenguajes de programación, expresiones matemáticas balanceadas y estructuras XML/HTML.

La experimentación con cadenas aceptadas y rechazadas refuerza la comprensión de conceptos críticos como las operaciones push (apilar), pop (desapilar), el símbolo inicial de pila (Z), las transiciones épsilon, y la diferencia entre aceptación por estado final versus aceptación por pila vacía. El proceso iterativo de diseño, prueba y refinamiento desarrollado durante el laboratorio fortalece el pensamiento algorítmico y la capacidad de depuración sistemática.

Asimismo, este ejercicio representa una oportunidad formativa para el fortalecimiento de competencias esenciales en la ingeniería informática y la computación teórica, como el razonamiento lógico sobre estructuras de datos abstractas, el diseño de máquinas de estado con memoria, y la aplicación de conceptos matemáticos en la resolución de problemas computacionales complejos. Estas habilidades resultan fundamentales para comprender el funcionamiento interno de los analizadores sintácticos (parsers), la validación de paréntesis balanceados, el procesamiento de lenguajes de programación y la construcción de compiladores, pilares del desarrollo de software avanzado.

Durante el desarrollo del laboratorio, se enfrentaron diversos desafíos técnicos relacionados con la configuración apropiada de JFLAP, particularmente la distinción crítica entre "Single Character Input" y "Multiple Character Input" en autómatas a pila, el formato exacto de especificación de transiciones (read, pop, push), y el diseño de estrategias eficientes de conteo mediante estados intermedios. La superación de estos obstáculos mediante experimentación sistemática y análisis de errores constituye un aprendizaje valioso sobre la brecha entre conocimiento teórico y aplicación práctica.

En síntesis, el Laboratorio: Autómatas a Pila con JFLAP no se limita a la simple simulación de autómatas, sino que constituye un ejercicio integral que articula teoría, práctica y análisis formal, permitiendo al estudiante afianzar su dominio sobre los fundamentos del reconocimiento de lenguajes libres de contexto. Este trabajo representa un paso fundamental en la jerarquía de Chomsky, sirviendo de puente entre los lenguajes regulares estudiados previamente y los lenguajes más complejos que

serán abordados posteriormente, fortaleciendo la comprensión estructural de la computación y la capacidad de aplicar modelos formales con memoria auxiliar en entornos reales de análisis sintáctico y compilación.

2. DESARROLLO DEL LABORATORIO

Este laboratorio tiene como objetivo comprender y aplicar los conceptos de autómatas a pila (PDA - Pushdown Automata) y lenguajes libres de contexto mediante el uso de la herramienta JFLAP (Java Formal Languages and Automata Package).

A partir de dos especificaciones de lenguajes formales proporcionadas en la actividad, se realizaron las siguientes tareas:

- Diseño y construcción de autómatas a pila en JFLAP
- Implementación de estrategias de apilamiento y desapilamiento
- Configuración de transiciones con operaciones de pila
- Prueba y validación exhaustiva de cadenas aceptadas y rechazadas
- Análisis del comportamiento de la pila durante la ejecución

Este trabajo permite establecer la relación práctica entre los lenguajes libres de contexto y los autómatas a pila como mecanismo de reconocimiento, demostrando cómo la incorporación de memoria auxiliar (pila) expande el poder computacional más allá de los autómatas finitos.

2.1 METODOLOGÍA

2.1.1 Instalación y Configuración de JFLAP

Para el desarrollo de este laboratorio se utilizó JFLAP 7.1, una herramienta educativa especializada en el estudio de autómatas y lenguajes formales.

2.1.2 Proceso de instalación:

2.1.2.1 Descarga de JFLAP:

- Se accedió al sitio oficial: https://www.jflap.org/jflaptmp/
- Se descargó el archivo JFLAP7.1.jar

2.1.2.2 Instalación de Java:

- Requisito previo: Java Runtime Environment (JRE)
- Se instaló Java desde: https://www.java.com/es/download/
- Versión utilizada: Java(TM) Platform SE binary

2.1.2.3 Ejecución de JFLAP:

- Se ejecutó el archivo JFLAP7.1.jar
- El sistema solicitó seleccionar la aplicación para abrir archivos .jar
- Se seleccionó: Java(TM) Platform SE binary
- JFLAP se ejecutó correctamente

2.1.3 Selección del Tipo de Autómata

2.1.3.1 Desafío crítico identificado:

Durante el desarrollo del laboratorio se identificó que JFLAP ofrece dos variantes de autómatas a pila:

- **Pushdown Automaton - Single Character Input:** Permite apilar únicamente UN carácter por transición

- **Pushdown Automaton - Multiple Character Input:** Permite apilar MÚLTIPLES caracteres por transición

2.1.3.2 Decisión técnica:

Después de pruebas iniciales con "Single Character Input" que resultaron infructuosas debido a la limitación de apilar solo un símbolo a la vez, se determinó que era necesario utilizar "Pushdown Automaton - Multiple Character Input" para implementar correctamente las estrategias de reconocimiento requeridas por los lenguajes especificados.

Esta variante permite operaciones como apilar "aZ" o "aa" en una sola transición, lo cual es fundamental para los diseños propuestos.

2.2 Construcción De Los Autómatas A Pila

2.2.1 Autómata 1: Lenguaje $L^1 = \{X^n Y^{2n} : n > 0\}$

2.2.1.1 Especificación del lenguaje:

Este lenguaje requiere que por cada símbolo X en la cadena, haya exactamente el doble de símbolos Y. La condición n > 0 indica que no se acepta la cadena vacía.

2.2.1.2 Ejemplos válidos:

- XYY (1 X, 2 Y's)
- XXYYYY (2 X's, 4 Y's)
- XXXYYYYYY (3 X's, 6 Y's)

2.2.1.2.1 Estrategia de reconocimiento:

Para reconocer este lenguaje, se implementó la siguiente estrategia:

- Por cada X leída, apilar UN símbolo 'a'
- Por cada DOS Y's leídas, desapilar UN símbolo 'a'
- Al final, verificar que la pila solo contenga el símbolo de fondo Z

2.2.1.3 Diseño del autómata:

2.2.1.3.1 Componentes:

- **Estados:** q0, q1, q2, q3, q4 (5 estados)
- **Estado inicial:** q0
- **Estado final:** q4
- **Alfabeto de entrada:** Σ = {X, Y}
- **Alfabeto de pila:** Γ = {Z, a}
- **Símbolo inicial de pila:** Z

2.2.1.3.2 Proceso de construcción en JFLAP:

2.2.1.3.2.1 Creación de estados:

- Se seleccionó "Pushdown Automaton - Multiple Character Input"
- Con la herramienta "State Creator" se crearon los 5 estados
- Se distribuyeron horizontalmente para facilitar la visualización del flujo

2.2.1.3.2.2 Configuración de estados especiales:

- Estado inicial: Se marcó q0 como inicial mediante clic derecho → "Initial"
- Estado final: Se marcó q4 como final mediante clic derecho → "Final"

2.2.1.3.2.3 Creación de transiciones:

- Se utilizó la herramienta "Transition Creator"
- Para cada transición se hizo clic en el estado origen y se arrastró hasta el estado destino
- Se ingresaron los tres componentes en los campos correspondientes: Read, Pop, Push

2.2.1.3.3 Formato de transiciones en JFLAP:

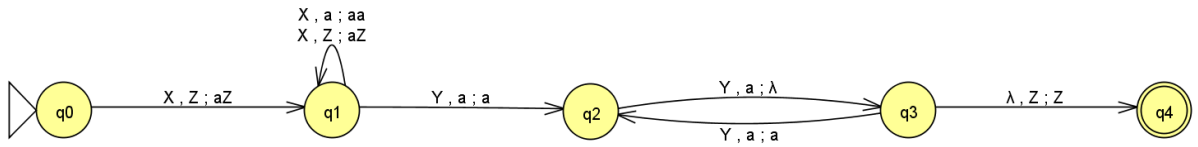
JFLAP con "Multiple Character Input" presenta tres campos separados para cada transición:

- **Campo 1 (Read from tape):** Símbolo a leer de la entrada
- **Campo 2 (Pop from stack):** Símbolo a desapilar
- **Campo 3 (Push to stack):** Símbolo(s) a apilar

2.2.1.4 Transiciones implementadas:

Origen	Destino	Read	Pop	Push	Descripción
q0	q1	X	Z	aZ	Lee primera X, apila 'a' sobre Z
q1	q1	X	Z	aZ	Lee más X's con pila solo Z
q1	q1	X	a	aa	Lee más X's cuando ya hay 'a's
q1	q2	Y	a	a	Lee primera Y, mantiene 'a'
q2	q3	Y	a	(vacío)	Lee segunda Y, desapila 'a'
q3	q2	Y	a	a	Si hay más Y's, reinicia conteo
q3	q4	(vacío)	Z	Z	Transición épsilon al estado final

2.2.1.5 Diagrama del autómata:

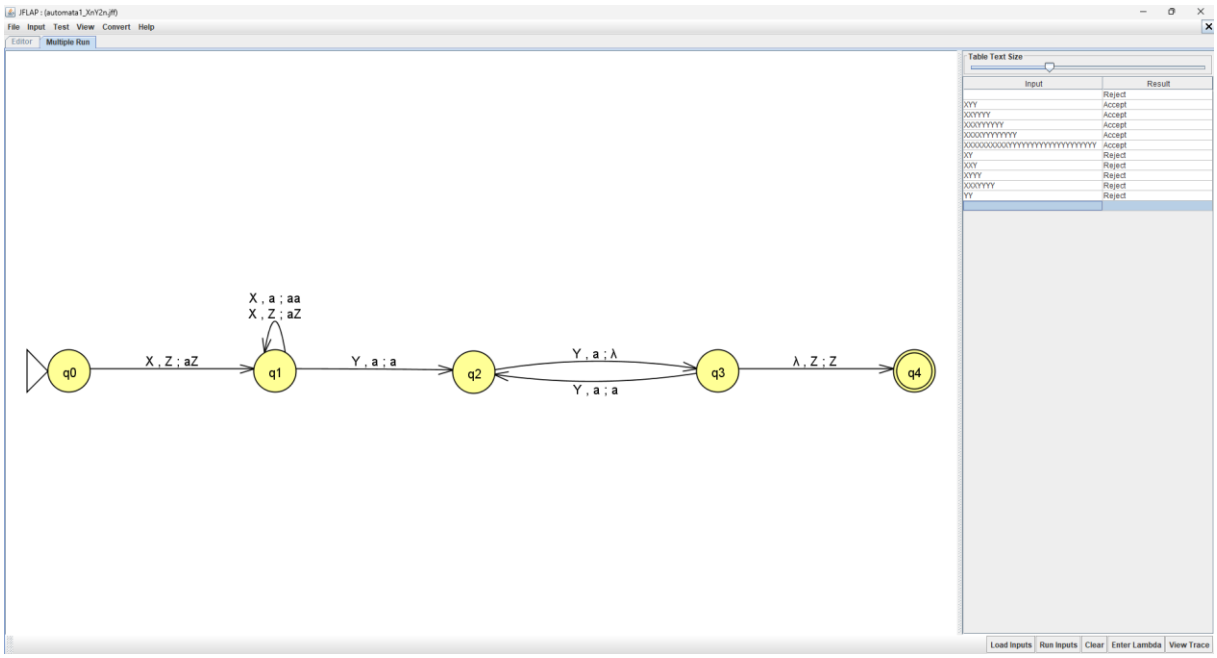


2.2.1.6 Desafíos enfrentados:

1. **Formato de transiciones:** Inicialmente se intentó usar el formato con comas y punto y coma (X,Z;aZ) pero JFLAP requiere campos separados
2. **Espacios en las especificaciones:** Se descubrió que NO deben incluirse espacios en los valores de los campos
3. **Estrategia de conteo:** Se requirió diseñar un mecanismo con estados intermedios (q2, q3) para contar Y's de dos en dos

2.2.1.7 Validación del autómata:

El autómata fue probado extensivamente usando la función "Multiple Run" de JFLAP. A continuación la imagen con los resultados:



2.2.2 Autómata 2: Lenguaje $L^2 = \{X^{2n}Y^n : n \geq 0\}$

2.2.2.1 Especificación del lenguaje:

Este lenguaje requiere que haya exactamente el doble de símbolos X que de símbolos Y. La condición n ≥ 0 indica que SÍ se acepta la cadena vacía.

2.2.2.2 Ejemplos válidos:

- ε (cadena vacía)
- XXY (2 X's, 1 Y)
- XXXXYY (4 X's, 2 Y's)
- XXXXXXYYY (6 X's, 3 Y's)

2.2.2.3 Estrategia de reconocimiento:

Para reconocer este lenguaje, se implementó la siguiente estrategia:

- Aceptar la cadena vacía mediante transición épsilon desde q0
- Por cada PAR de X's leídas, apilar UN símbolo 'a'
- Por cada Y leída, desapilar UN símbolo 'a'
- Al final, verificar que la pila solo contenga el símbolo de fondo Z

2.2.2.4 Diseño del autómata:

2.2.2.4.1 Componentes:

- **Estados:** q0, q1, q2, q3, q4 (5 estados)
- **Estado inicial:** q0
- **Estado final:** q4
- **Alfabeto de entrada:** Σ = {X, Y}
- **Alfabeto de pila:** Γ = {Z, a}
- **Símbolo inicial de pila:** Z

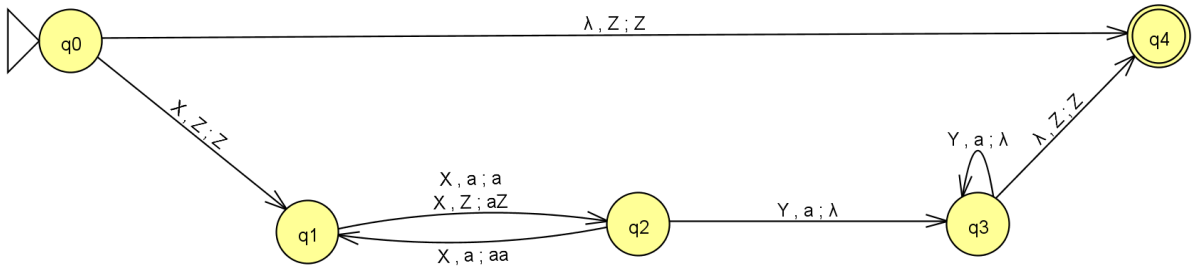
2.2.2.4.2 Proceso de construcción en JFLAP:

Similar al autómata anterior, se utilizó "Pushdown Automaton - Multiple Character Input" y se crearon 5 estados con las configuraciones apropiadas.

2.2.2.5 Transiciones implementadas:

Origen	Destino	Read	Pop	Push	Descripción
q0	q4	(vacío)	Z	Z	Acepta cadena vacía
q0	q1	X	Z	Z	Lee primera X (impar) sin apilar
q1	q2	X	Z	aZ	Lee segunda X (par), apila 'a'
q1	q2	X	a	a	Lee X impar cuando hay 'a's
q2	q1	X	Z	aZ	Lee X par con solo Z
q2	q1	X	a	aa	Lee X par y apila otra 'a'
q2	q3	Y	a	(vacío)	Comienza a leer Y's, desapila
q3	q3	Y	a	(vacío)	Sigue leyendo Y's
q3	q4	(vacío)	Z	Z	Transición épsilon al estado final

2.2.2.6 Diagrama del autómata:

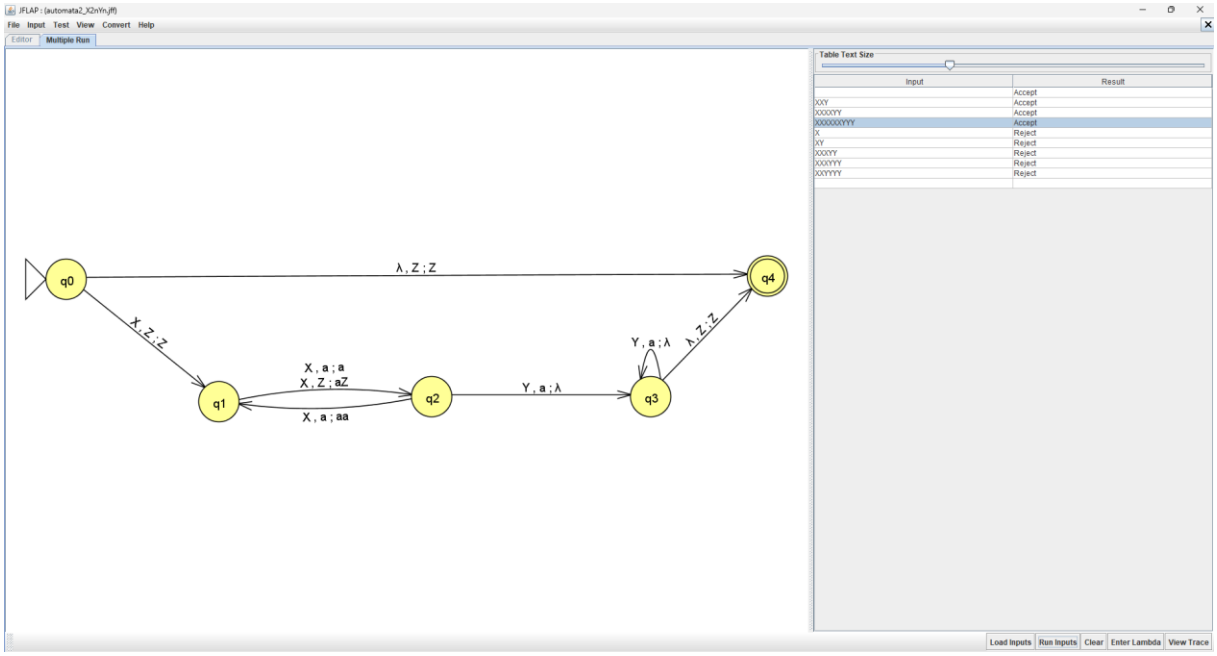


2.2.2.7 Características especiales:

1. **Aceptación de cadena vacía:** La transición directa q0 → q4 con λ,Z;Z permite aceptar ε
2. **Conteo de pares:** El mecanismo q1 ↔ q2 alterna entre X's impares y pares, apilando solo en las pares
3. **Simplicidad en desapilamiento:** A diferencia del autómata 1, aquí cada Y desapila directamente una 'a'

2.2.2.8 Validación del autómata:

El autómata fue probado extensivamente usando la función "Multiple Run" de JFLAP. A continuación la imagen con los resultados:



2.2.3 VALIDACIÓN Y PRUEBAS

2.2.3.1 Metodología de Pruebas

Para ambos autómatas se utilizaron dos métodos de validación en JFLAP:

1. **Multiple Run:** Permite probar múltiples cadenas simultáneamente
2. **Step by State:** Permite ejecutar paso a paso observando el comportamiento de la pila

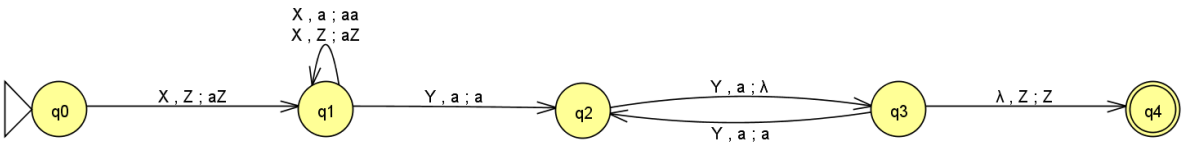
2.2.3.1.1 Proceso de prueba:

- Se ingresaron conjuntos de cadenas válidas e inválidas
- JFLAP mostró visualmente cuáles eran aceptadas (verde) y cuáles rechazadas (rojo)
- Se verificó el comportamiento de la pila mediante ejecución paso a paso

2.2.4 RESULTADOS

2.2.4.1 Autómata 1: $L^1 = \{X^n Y^{2n} : n > 0\}$

2.2.4.1.1 Imagen del autómata construido:



2.2.4.1.2 Pruebas de Aceptación y Rechazo:

2.2.4.1.2.1 Cadenas ACEPTADAS:

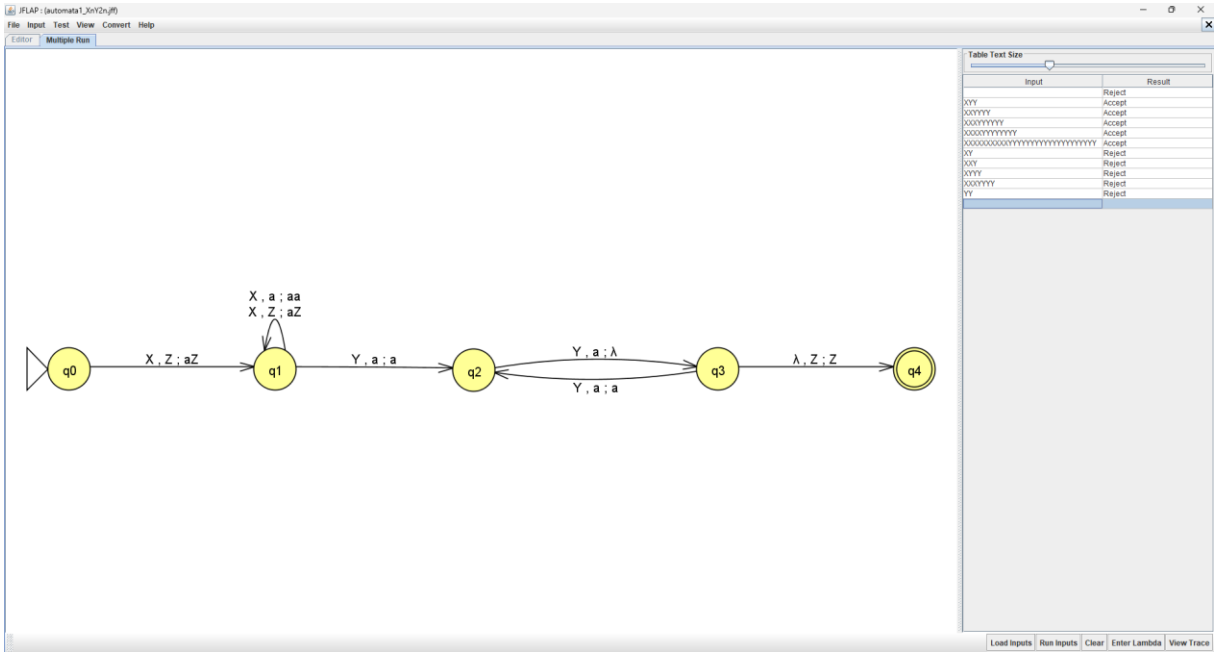
#	Cadena	Análisis de Pila	Estado Final
1	XYY	Z → aZ → a → Z	q4 (final)
2	XXYYYY	Z → aZ → aaZ → aa → a → Z	q4 (final)
3	XXXYYYYYY	Z → aZ → aaZ → aaaZ → aaa → aa → a → Z	q4 (final)
4	XXXXYYYYYYYY	Apila 4 a's, desapila 4 a's (de 2 en 2)	q4 (final)
5	XXXXXXXXXXYYYYYYYYYYYYYYYYYYYY	Apila 10 a's, desapila 10 a's	q4 (final)

2.2.4.1.2.2 Cadenas RECHAZADAS:

#	Cadena	Razón de Rechazo	Estado Final
1	XY	Solo 1 Y, necesita 2	No alcanza q4
2	XXY	Solo 1 Y, necesita 4	No alcanza q4
3	XYYY	3 Y's no es múltiplo de 2	No alcanza q4
4	XXXYYYY	4 Y's para 3 X's (debería ser 6)	No alcanza q4
5	YY	Comienza con Y sin X	Falla desde q0

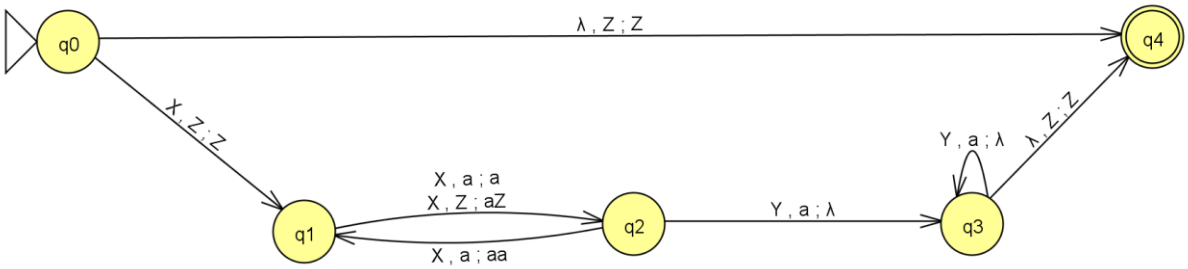
2.2.4.1.2.3 Imagen del Desarrollo de Pruebas:

A continuación la imagen completa con las pruebas desarrolladas en JFLAP:



2.1.1.1 Autómata 2: $L^2 = \{X^{2n}Y^n : n \geq 0\}$

2.1.1.1.1 Imagen del autómata construido:



2.1.1.1.2 Pruebas de Aceptación y Rechazo:

2.1.1.1.2.1 Cadenas ACEPTADAS:

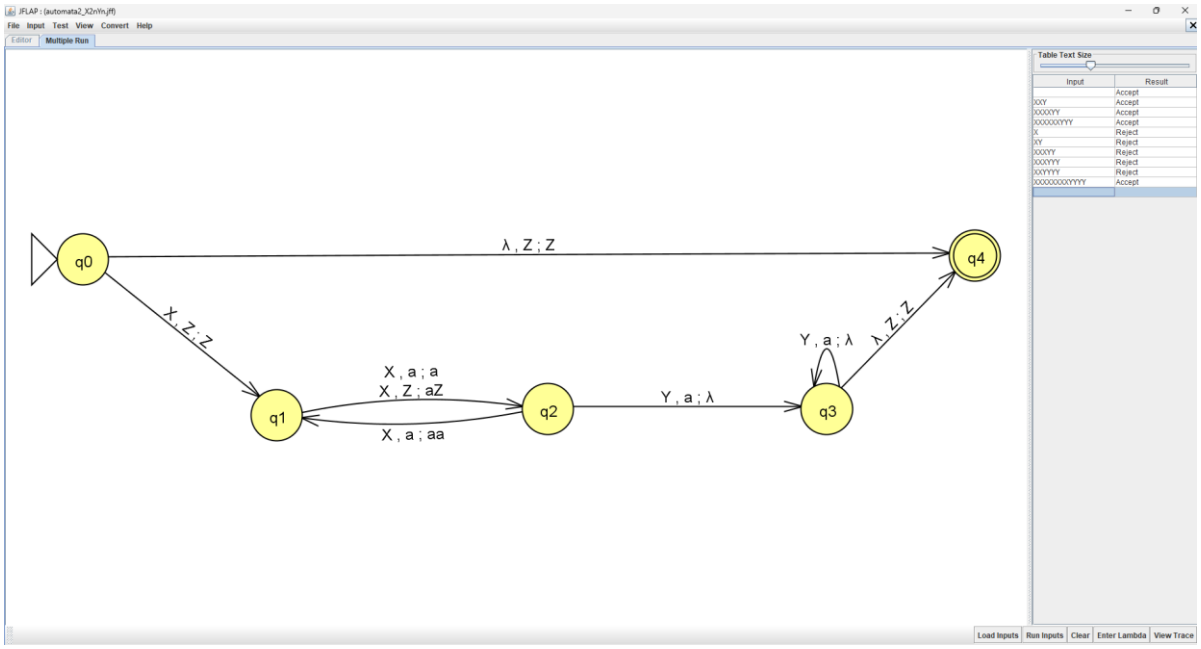
#	Cadena	Análisis de Pila	Estado Final
1	ε	Transición directa q0→q4	q4 (final)
2	XXY	Z → Z → aZ → Z	q4 (final)
3	XXXXYY	Apila 2 a's (4X ÷ 2), desapila 2 a's	q4 (final)
4	XXXXXXYYY	Apila 3 a's (6X ÷ 2), desapila 3 a's	q4 (final)
5	XXXXXXXYYYY	Apila 4 a's (8X ÷ 2), desapila 4 a's	q4 (final)

2.1.1.1.2.2 Cadenas RECHAZADAS:

#	Cadena	Razón de Rechazo	Estado Final
1	X	X impar sin su par	No alcanza q4
2	XY	Solo 1 X, necesita 2	No alcanza q4
3	XXXYY	3 X's (impar) para 2 Y's	No alcanza q4
4	XXXYYY	3 X's para 3 Y's (debería ser 1.5Y)	No alcanza q4
5	XXYYYY	2 X's para 4 Y's (debería ser 1Y)	No alcanza q4

2.1.1.1.2.3 Imagen del Desarrollo de Pruebas:

A continuación la imagen completa con las pruebas desarrolladas en JFLAP:



2.2.5 ANÁLISIS TÉCNICO

2.2.5.1 Comparación de Estrategias

2.2.5.1.1 Autómata 1 (X^n Y^(2n)):

- Estrategia: Apilar 1 símbolo por X, desapilar 1 símbolo por cada 2 Y's
- Complejidad: Requiere estados intermedios para contar pares de Y's
- Estados necesarios: 5 (q0, q1, q2, q3, q4)

2.2.5.1.2 Autómata 2 (X^(2n) Y^n):

- Estrategia: Apilar 1 símbolo por cada 2 X's, desapilar 1 símbolo por Y
- Complejidad: Requiere alternancia entre X impares y pares
- Estados necesarios: 5 (q0, q1, q2, q3, q4)

2.2.6 LECCIONES APRENDIDAS

2.2.6.1 Desafíos técnicos superados:

- **Selección del tipo de autómata:** La diferencia crítica entre "Single" y "Multiple Character Input"
- **Formato de transiciones:** Comprender que JFLAP usa campos separados, no notación con comas y punto y coma
- **Diseño de estrategias de conteo:** Implementar mecanismos de estados intermedios para contar símbolos en grupos

2.2.6.2 Conceptos reforzados:

- Operaciones de pila: push (apilar) y pop (desapilar)
- Transiciones épsilon (λ) para validación final
- Símbolo de fondo de pila (Z) como marcador de pila vacía
- Diferencia entre aceptación por estado final vs. pila vacía

2. CONCLUSIONES

A lo largo del desarrollo de este laboratorio, logré obtener un aprendizaje integral sobre el diseño, implementación y validación de autómatas a pila como mecanismo de reconocimiento de lenguajes libres de contexto, empleando el entorno JFLAP como herramienta de simulación. En primer lugar, el trabajo práctico con autómatas a pila me permitió comprender de forma visual y estructurada el funcionamiento de las operaciones de pila (push y pop), la importancia de la memoria auxiliar en el reconocimiento de patrones, y cómo la combinación de estados, transiciones y estructura de datos tipo pila contribuye al reconocimiento de cadenas con dependencias estructurales complejas.

El proceso de diseñar estrategias diferenciadas para cada lenguaje (X^n Y^(2n) y X^(2n) Y^n) reforzó mi comprensión sobre la necesidad de adaptar el enfoque de apilamiento y desapilamiento según las características específicas del lenguaje a reconocer. Pude evidenciar cómo lenguajes aparentemente similares requieren mecanismos de conteo y validación completamente diferentes, lo que amplía la perspectiva sobre la importancia del análisis previo al diseño de autómatas y la necesidad de comprender profundamente la estructura del lenguaje objetivo.

Asimismo, el ejercicio de construir mecanismos de conteo mediante estados intermedios (como el conteo de pares de Y's en el primer autómata, o la alternancia entre X's impares y pares en el segundo) me permitió desarrollar habilidades de diseño algorítmico aplicadas a autómatas. Esta experiencia fortaleció mi capacidad para descomponer problemas complejos en estados y transiciones más simples, promoviendo el razonamiento lógico estructurado y la precisión en la especificación de operaciones de pila.

El enfrentamiento y superación de desafíos técnicos durante el desarrollo constituyó un aprendizaje particularmente valioso. La identificación del problema con "Single Character Input" versus "Multiple Character Input", la comprensión del formato exacto de especificación de transiciones en JFLAP, y la depuración sistemática de errores me enseñaron la importancia de la paciencia, la experimentación controlada y la lectura cuidadosa de documentación técnica. Estos obstáculos, lejos de ser frustraciones, se convirtieron en oportunidades de aprendizaje profundo sobre la brecha entre conocimiento teórico y aplicación práctica.

Por otro lado, la validación exhaustiva mediante conjuntos de pruebas cuidadosamente diseñados me permitió aplicar de manera práctica los conceptos teóricos, observando en tiempo real cómo las operaciones de pila determinaban la aceptación o rechazo de cadenas. Esta interacción fortaleció mi capacidad para trazar la ejecución de autómatas, visualizar el estado de la pila en cada paso, y comprender cómo la verificación del símbolo de fondo (Z) garantiza el balance correcto entre símbolos de entrada.

La documentación detallada y análisis de los resultados me ayudaron a valorar la importancia de la claridad y el rigor en la presentación de soluciones formales complejas. La correcta especificación de transiciones, el uso coherente de notaciones para operaciones de pila, y la verificación mediante simulación paso a paso facilitaron la comprensión y trazabilidad del proceso, elementos esenciales en entornos académicos y profesionales donde la precisión formal es indispensable, especialmente en el diseño de compiladores y analizadores sintácticos.

Este laboratorio me permitió comprender la posición de los autómatas a pila dentro de la jerarquía de Chomsky, situándose entre los autómatas finitos (lenguajes regulares) y las máquinas de Turing (lenguajes recursivamente enumerables). Pude apreciar cómo la incorporación de memoria auxiliar tipo pila permite reconocer lenguajes con estructuras anidadas, balanceo de paréntesis, y dependencias contextuales que son fundamentales en el análisis sintáctico de lenguajes de programación.

Finalmente, este laboratorio no solo representó una actividad técnica, sino también una experiencia formativa que consolidó mis competencias en teoría de autómatas a pila, lenguajes libres de contexto y diseño de mecanismos de reconocimiento con memoria auxiliar. La práctica con JFLAP me permitió integrar la teoría con la experimentación, entendiendo cómo los fundamentos matemáticos de la computación sustentan el diseño de parsers, analizadores sintácticos y compiladores modernos.

La experiencia de trabajar con dos autómatas diferentes para lenguajes con relaciones inversas (n:2n vs 2n:n) me enseñó que pequeñas variaciones en la especificación del lenguaje pueden requerir cambios significativos en la estrategia de reconocimiento, reforzando la necesidad de análisis cuidadoso y diseño reflexivo antes de la implementación.

En conclusión, el Laboratorio: Autómatas a Pila con JFLAP me permitió fortalecer mis habilidades analíticas, mi razonamiento formal sobre estructuras de datos abstractas, mi capacidad de diseño algorítmico, y mi comprensión sobre los mecanismos que definen el reconocimiento de lenguajes libres de contexto. Este aprendizaje constituye un avance fundamental desde los lenguajes regulares hacia lenguajes más complejos, preparándome para abordar temas más avanzados como gramáticas libres de contexto, análisis sintáctico descendente y ascendente, y eventualmente máquinas de Turing, pilares esenciales en la formación del ingeniero informático y en el desarrollo de compiladores, intérpretes y sistemas de procesamiento de lenguajes precisos y eficientes.

3. BIBLIOGRAFÍA

Respetado profesor Ing. Rogerio Orlando Beltrán Castro, a continuación se presenta la bibliografía y los recursos de apoyo utilizados para el desarrollo del Laboratorio: Autómatas a Pila con JFLAP:

- Clases virtuales dirigidas por el profesor Ing. Rogerio Orlando Beltrán Castro.
- Material de estudio sobre autómatas a pila y lenguajes libres de contexto proporcionado en el aula virtual.
- Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2006). *Introduction to Automata Theory, Languages, and Computation* (3.ª ed.). Pearson Education. [Capítulos sobre Pushdown Automata y Context-Free Languages]

- Sipser, M. (2013). *Introduction to the Theory of Computation* (3.ª ed.). Cengage Learning. [Capítulo 2: Context-Free Languages y Pushdown Automata]
- Linz, P. (2011). *An Introduction to Formal Languages and Automata* (5.ª ed.). Jones & Bartlett Learning. [Capítulos sobre autómatas a pila y gramáticas libres de contexto]
- JFLAP Official Website. (s.f.). *JFLAP - Java Formal Languages and Automata Package*. Recuperado de: https://www.jflap.org/
- JFLAP Documentation. (s.f.). *Pushdown Automata in JFLAP*. Recuperado de: https://www.jflap.org/tutorial/
- Rodger, S. H., & Finley, T. W. (2006). *JFLAP: An Interactive Formal Languages and Automata Package*. Jones & Bartlett Publishers.
- Oracle Corporation. (2023). *Java Platform, Standard Edition Documentation*. Recuperado de: https://www.java.com/es/

Agradecimientos especiales

Al profesor Ing. Rogerio Orlando Beltrán Castro por su guía, paciencia y dedicación durante el desarrollo de este laboratorio, especialmente en la resolución de desafíos técnicos relacionados con la configuración de JFLAP y el diseño de estrategias de reconocimiento para autómatas a pila. Sus enseñanzas han sido fundamentales para comprender los conceptos de lenguajes libres de contexto y su aplicación práctica.

¡Que Dios lo bendiga, profesor!