JLEFO: una herramienta para el aprendizaje de lenguajes y autómatas

Colaboración
Ofelia Gutiérrez Giraldi; Martha Martínez Moreno; Jose Carlos Xolio Xolio; Margarito Hernández García; Alexis Salazar Viveros, Tecnológico Nacional de México / Instituto Tecnológico de Veracruz.

RESUMEN: El grado de complejidad que presenta la teoría de Lenguajes Formales y Autómatas, generó el interés de los profesores del área de sistemas y computación de crear una herramienta que permitiera a los estudiantes plantear soluciones a problemas relacionados con el tema, enfocándose en estructuras con Autómatas Finitos y Expresiones Regulares y generar así un aprendizaje significativo.

A pesar de que existen otros softwares similares, a las necesidades que los estudiantes requieren.

En la implementación se utilizaron metodologías de desarrollo ágil y de programación: Extreme Programming y MVC respectivamente y como lenguaje de programación Java. El resultado fue una herramienta de aspecto minimalista, simple, eficiente y de gran utilidad a través de la visualización de datos importantes para el estudiante, aportando más información que sustenta su aprendizaje.

PALABRAS CLAVE: Autómatas Finitos, Expresiones Regulares, Extreme Programming, Java, Lenguajes Formales, MVC.

ABSTRACT: The theory of formal languages and automatons is part of the academic training of a computer systems engineer. In the subject of Languages and Automata I (specialized in this area) that is taught in the ITVer, the students present difficulties to raise the solution of problems related to the subject, so they thought about the development of software for the solution of structures with Finite automatons and regular expressions and thus generate meaningful learning.

Although there is other similar software, highlighting JFLAP, none adapts to the needs of the students.

In its implementation, agile development and programming methodologies were used: Extreme Programming and MVC respectively and as a Java programming language. The result was a minimalist, simple, efficient and very useful tool through the visualization of important data for the student, providing more information to support their learning.

KEYWORDS: Extreme Programming, Finite Automaton, Formal Languages, Java, MVC, Regular Expressions.

INTRODUCCIÓN

En el ITVER (Instituto Tecnológico de Veracruz) se imparte la materia de Lenguajes y Autómatas I como parte de la retícula de la carrera Ingeniería en Sistemas Computacionales. Dado el grado de dificultad de la misma, los estudiantes presentan problemas para plantear soluciones a través de la aplicación de estructuras matemáticas que resuelvan los ejercicios relacionados con autómatas finitos deterministas y expresiones regulares; los problemas principales son el rastreo de cadenas, diagramas incompletos y comprensión del flujo del diagrama.

Considerando los objetivos de la investigación se realizó un análisis del entorno en el contexto de la generación de lenguajes formales, se hizo un análisis de distintas herramientas existentes (se encontró que ninguno cubre las necesidades actuales de los estudiantes) y se diseñó el lenguaje específico para la interpretación de las estructuras; justificando el desarrollo de una herramienta propia para la materia. El software se realizó bajo el lenguaje Java y en la construcción del proyecto se utilizaron dos metodologías: desarrollo ágil Extreme Programming y desarrollo software MVC (Modelo Vista Controlador) aportando los resultados deseados y una estructura interna sólida para posteriores integraciones.

JLEFO cuenta con una interfaz atractiva e incluye los siguientes tres módulos:
- AFD (Autómatas Finitos Deterministas)
- AFND (Autómatas Finitos No Deterministas)
- e.r.'s (Expresiones Regulares)

Como resultado se obtuvo una herramienta funcional, que muestra los datos necesarios comprendidos en el margen de conocimientos impartidos en la materia; así, a través de pruebas de conocimientos sin la herramienta y con su utilización, se obtuvo un porcentaje incremental en los índices de aprovechamiento reflejándose en un aprendizaje significativo en los estudiantes.

MATERIAL Y MÉTODOS

Metodología

La metodología utilizada para el desarrollo y gestión del proyecto fue XP (Extreme Programming) considerada de desarrollo ágil.

XP es una de las llamadas metodologías ágiles de desarrollo de software más exitosas de los tiempos recientes. Está diseñada para entregar el software que los clientes necesitan en el momento en que lo requieren. XP alienta a los desarrolladores a responder a los requerimientos cambiantes de los clientes, aún en fases tardías del ciclo de vida del desarrollo. [1]

Esta metodología se compone por cuatro fases:
Planificación: Fase inicial de la metodología XP, donde se establece una comunicación continua entre el equipo de desarrollo y el cliente para obtener principalmente los requisitos del sistema. Además, permite establecer el alcance del proyecto y fechas de entrega del sistema, tomando en cuenta la prioridad y tiempo estimado para el desarrollo de cada historia de usuario. [2]

Diseño: Fase en la cual se crean estándares y patrones para realizar la codificación, de esta forma se obtiene un código más eficiente, con calidad y comprensión para todos los involucrados en el desarrollo.

Desarrollo: En esta fase se lleva a cabo toda la codificación del sistema siguiendo los estándares y patrones definidos en la fase dos, por lo cual los programadores ejecutan las Historias de Usuario mediante tareas de ingeniería hasta cumplir con cada una de ellas.

Pruebas: Última fase y piedra angular de la metodología XP, en ella se le da seguimiento al desarrollo del sistema mediante la implementación de pruebas unitarias, garantizando el menor número posible de errores al finalizar la construcción del proyecto.

Utilizar XP brindó al desarrollo la capacidad de minimizar los errores a través de la programación por pares, cumplir con las metas u objetivos del proyecto más relevantes, reducir los tiempos de programación, producir versiones funcionales y adaptación rápida a los cambios surgidos durante el desarrollo.

Como parte del desarrollo del software, se utilizó el patrón de diseño de software MVC (Modelo Vista Controlador) que es un estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control en tres componentes distintos. [3]

*El Modelo que contiene una representación de los datos que maneja el sistema, la lógica de negocio y los mecanismos de persistencia.

*La Vista o interfaz de usuario, que compone la información que se envía al cliente y los mecanismos de interacción con éste.

*El Controlador, que actúa como intermediario entre el Modelo y la Vista, gestionando el flujo de información entre ellos y las transformaciones para adaptar los datos a las necesidades de cada uno.

Utilizar el patrón MVC produjo una estructura sólida y orden lógico en el código, además de combinar su utilidad con el lenguaje de programación utilizado (Java).

Producción del ejecutable
De acuerdo a la metodología XP, en cada meta alcanzada se realizaron pruebas de funcionalidad, siendo necesaria la producción de un ejecutable para aplicarlas a la herramienta. Para lo anterior se utilizaron aplicaciones como Netbeans, Launch4j, Inno Setup Compiler.

Utilizando como entorno de desarrollo la aplicación de Netbeans se generó un archivo JAR, posteriormente a través de Launch4j se convirtió a un archivo EXE y se indicó la versión del JRE necesaria para su ejecución, una vez obtenido el archivo EXE con la ayuda de Inno Setup Compiler se realizó el instalador a través de un script para generar un ejecutable de la aplicación que se pudiera lanzar en el sistema operativo Windows.

Desarrollo

La primera parte del desarrollo consistió en diseñar las interfaces de la herramienta de una manera que el usuario no perdiera de vista ninguna de las opciones disponibles, bajo

el típico modelo de un programa con una barra de menú, barra de herramientas, menús de opciones y un área de canvas. Ese concepto de interfaz fue programado y añadido a un nuevo paquete del proyecto denominado "Vista" haciendo alusión a MVC.

Posteriormente se definen los modelos como lo dicta MVC, siendo los datos propios de la herramienta que se ponen a disposición del usuario, definidos por estados y transiciones propios de un diagrama AFD y AFND. En este punto se crea el paquete "Modelo" que almacén ó dichas clases.

Para completar el patrón de diseño MVC se definieron los controladores, encargados de todas las operaciones de la herramienta. Un controlador corresponde a cada una de las interfaces creadas, dando lugar a la creación del paquete "Control".

Análisis de cadenas

La función principal de la herramienta es el rastreo de las cadenas aceptadas por las estructuras definidas por el usuario; la teoría que sustenta este análisis es la Tabla de Transiciones la cual es representada por una matriz que describe las transiciones de los estados con los elementos del conjunto del alfabeto, incluyendo los que tienen como elemento de transición épsilon, además de indicar los estados finales o estados de aceptación del diagrama. La lectura de este tipo de tabla depende de la cadena de entrada y la función de transición dada por el diagrama, por ejemplo, si tenemos una cadena binaria "101" que corresponde al alfabeto ∑= {1,0} en un diagrama de transiciones con la siguiente tabla de transiciones. Tabla 1.

Tabla 1. Ejemplo de tabla de transiciones.

Q	∑ = 0	∑ = 1	F
q0	q1	q1	-
q1	q0	q1	F

Para saber si la cadena es aceptada o rechazada el recorrido comienza por el primer estado de transición "q0", se toma el primer carácter de la cadena "101" y a través de la función de transición se verifica a qué estado se debe realizar la transición con ese "1", para este ejemplo la transición es hacia "q1", ahora el nuevo estado para consumir otro elemento de la cadena es "q1"; este proceso es repetitivo hasta consumir toda la cadena y determinar si el estado donde finalizó la cadena es un estado final, en el ejemplo expuesto la cadena es aceptada por el estado final "q1". Este proceso se programó en una clase controladora del módulo AFD y es de suma importancia para los demás módulos.

Módulo AFD

Este módulo está orientado a la solución de Autómatas Finitos Deterministas, en este se puede crear desde cero un diagrama de transiciones y editarlo las veces que el usuario lo considere necesario. Una vez terminado el diagrama el usuario puede testear el diagrama; para este proceso se utilizó el método de análisis de cadenas antes descrito, además de complementarlo con un archivo de texto plano donde se almacenaron cadenas binarias de un caracter hasta nueve caracteres, estas sirven como cadenas de prueba que el usuario no se preocupa por generar manualmente. Como resultado de este proceso de muestran las cadenas que fueron aceptadas y rechazadas por el diagrama de transiciones.

Además, al obtener las cadenas aceptadas y rechazadas por el diagrama, el usuario puede seleccionar una cadena y visualizar en el diagrama como, carácter a carácter, el autómata consume cada uno de ellos. Para lograr esto se implementó el algoritmo Backtracking utilizado en el rastreo de AFND que se describe en la sección siguiente.

Este módulo brinda una opción de conversión del diagrama actual a una e.r. equivalente a través de la implementación del método algebraico de Brzozowski, el cual plantea que a través de la solución de un sistema de ecuaciones lineales obtenidas a partir del diagrama podemos obtener una ER.

Este método toma un único enfoque para generar una expresión regular. Plantea un sistema de ecuaciones (expresiones regulares) para cada estado en un AFD. Se resuelven las ecuaciones para Ra donde Ra es la expresión regular asociada con el estado inicial qn. Para cada estado qn en un AFD la ecuación para R1 es la unión de los términos. El sistema de ecuaciones toma la siguiente forma. Figura 1.

$$R_1 = a_1R_1 + a_2R_2 + \cdots$$
$$R_2 = a_1R_1 + a_2R_2 + a_3R_3 + \cdots$$
$$R_m = a_1R_1 + a_2R_2 + \cdots + \cdots \varepsilon$$

Figura 1: Formulación del sistema de ecuaciones del método algebraico de Brzozowski.z

ε es añadido si R_m es un estado final o de aceptación. [4]

El sistema puede ser resuelto por sustitución directa, excepto cuando la misma incógnita aparece a la derecha, izquierda o ambos lados de la ecuación. Esta situación ocurre cuando existe un ciclo hacia el mismo estado qn. El teorema de Arden es la llave para resolver esta situación. El teorema es el siguiente:

"Dada una ecuación de la forma X = AX + B, la ecuación tiene una única solución X = A*B".

Se utiliza el teorema para aislar Ra del lado izquierdo y así sucesivamente, sustituyendo Ra dentro de otra ecuación. Este proceso se repite hasta que se encuentra Rm sin incógnitas en ambos lados de la ecuación. [5] Este método se programó dentro del paquete "funciones" para aislar procesos específicos de funciones recurrentes o comunes.

Módulo AFND

Este módulo está dedicado a la solución y análisis de Autómatas Finitos No Deterministas. El funcionamiento de un AFND es muy similar al de un AFD. Sin embargo, mientras en un AFD sólo existe una posible acción para cada símbolo del alfabeto de entrada, en los AFND hay que considerar todos los casos en cada estado, ya que permiten cero, una o más transiciones de salida de un estado para el mismo símbolo del alfabeto de entrada.

Debido a que en programación tiene gran dificultad el manejo del indeterminismo de estas estructuras, para poder realizar el proceso de análisis de cadenas la herramienta internamente realiza una conversión a AFD; para dicha conversión se utiliza el método de construcción de tabla de subconjuntos que se observa en la Tabla 2. Considerando que, en la tabla de transiciones de un AFND, cada entrada es un conjunto de estados y que en la tabla de transiciones de un AFD, cada entrada es tan solo un estado, se comenzó por:

1. Construir la Tabla de Subconjuntos en base a la Tabla de Transiciones que consta de tres columnas, donde cada una está etiquetada la primera con Q que son los estados, las otras dos con un símbolo del alfabeto de entrada y cada fila se etiqueta con un conjunto de estados.
2. La primera fila se etiqueta con {q0}, estado inicial, y en cada entrada de la tabla [q0, Si] se almacena f (q0, Si) = {q0, q1, ..., qn} = P qué son los subconjuntos de estados.
3. Se etiqueta cada fila con cada uno de los conjuntos P que no tengan asociada una fila en la tabla (es decir, con cada uno de los conjuntos P que aparezcan por primera vez en la tabla) y se completa cada una de estas filas con el correspondiente f (P, Si).
4. El paso 3 se realiza hasta que no haya en la tabla conjuntos P sin filas asociadas.
5. Por último, se asocia a cada conjunto P que aparezca en la tabla un estado en el nuevo AFD y aquellos que tengan entre sus componentes algún estado final del AFND se considerarán estados finales en él AFD. [6]

Tabla 2. Tabla de subconjuntos

Q	∑ = 0	∑ = 1	F
{q0}	{q0, q1}	{q0}	-
{q0, q1}	{q0, q1}	{q0, q2}	-
{q0, q2}	{q0, q1}	{q0}	F

Al realizar este procedimiento se tiene la funcionalidad en la herramienta de poder obtener la conversión de un AFND a su equivalente AFD.

Una funcionalidad más con la que cuenta este módulo es la de poder visualizar el rastreo de cadenas sobre el diagrama; para lograr dicho propósito fue necesario la implementación del algoritmo Backtracking (o búsqueda con retroceso) que es una técnica de recursión intensiva para resolver problemas por etapas, que utiliza como árbol de decisiones la propia organización de la recursión. Cuando se "avanza" de etapa se realiza una llamada recursiva y cuando se "retrocede" lo que se hace es terminar el correspondiente proceso recursivo, con lo que efectivamente se vuelve al estado anterior por la pila de entornos creada en la recursión. Como el árbol de decisiones no es una estructura almacenada en memoria, se dice que el Backtracking utiliza un árbol implícito y que habitualmente se denomina árbol de expansión o espacio de búsqueda.

Se implementó para realizar el rastreo de una cadena sobre el diagrama del autómata construyendo posibles soluciones candidatas de manera sistemática desde el estado inicial hasta un estado de aceptación o no aceptación indicando una vez que se tienen todas las posibles soluciones se puede realizar el rastreo en el diagrama cambiando de color los elementos estados y transiciones de acuerdo a la simbología establecida como se muestra en la figura 2.

Figura 2. Simbología de rastreo.

Módulo e.r.'s

Este módulo se enfoca en la solución de expresiones regulares, es decir, el usuario puede ingresar la expresión regular que desee bajo el lenguaje definido de 0, 1 o ambos y decidir si quiere analizar la expresión o simplemente convertir la e.r.´s a un AFD equivalente. Si el usuario decide realizar la conversión se proporciona una opción de guardado para que posteriormente pueda abrir ese archivo en el módulo de AFD.

Para analizar qué cadenas acepta o rechaza la e.r. ingresada el módulo utiliza el método de análisis de cadenas pero primero es necesario convertir la e.r. a un AFD equivalente, debido a que no existe una manera directa de obtener una tabla de transiciones a partir de una expresión regular, por lo tanto para lograr el análisis

se realiza la conversión de la e.r. a un AFD equivalente internamente, es decir, se reemplaza el análisis de la expresión regular por el análisis del AFD que resulta de la conversión.

Para construir un AFD directamente a partir de una expresión regular existen varios métodos, unos más complejos que otros, pero para este caso se optó por el método de construcción de árboles sintácticos y método de subconjuntos respectivamente. Estos métodos requieren que la expresión regular sea extendida, es decir, que al final de la expresión se añade un símbolo que indique el final, el símbolo utilizado es "#". Un árbol sintáctico se constituye por nodos hojas que corresponden a los operandos (símbolos del alfabeto) y los nodos interiores corresponden a los operadores. A un nodo interior se le llama nodo-concat, nodo-o o modo-asterisco si se etiqueta mediante el operador de concatenación (punto), el operador de unión "|", o el operador "*", respectivamente. [7]

En la figura 3 se describe los elementos de un árbol sintáctico.

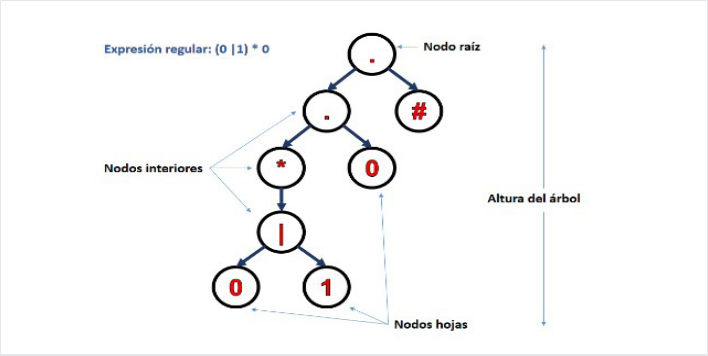

Figura 3. Elementos que conforman el árbol sintáctico.

Para llevar a cabo la conversión primero se construye el árbol sintáctico y después calcula las cuatro funciones: anulable, primera pos, última pos y siguiente pos, el cálculo de estas funciones se realiza de manera recursiva directa sobre la altura del árbol.

Después de que se realiza todo el cálculo de las funciones anteriores se procede a construir la tabla de transición resultante que es la que se utiliza para analizar qué cadenas acepta o rechaza la expresión regular.

RESULTADOS

El desarrollo de este proyecto dio como resultado una herramienta (JLEFO) orientada a la solución de autómatas finitos deterministas, autómatas finitos no deterministas y expresiones regulares, disponible para el sistema operativo Windows bajo el JRE 8.0.2 de Java, su interfaz es simple e intuitiva, fácil de usar.

Esta herramienta está dirigida a los estudiantes de la carrera de Ingeniería en Sistemas Computacionales (ITVER).

En la figura 4 se muestra la interfaz principal de la herramienta.

Figura 4. Interfaz principal de JLEFO.

La herramienta cuenta con tres módulos: AFD, AFND y e.r.'s.

En el módulo AFD el usuario puede diagramar un AFD y analizarlo, en el análisis se muestra la tabla de transición correspondiente al AFD diagramado, así como las cadenas aceptadas y rechazadas, además, puede visualizar como es consumida o rechazada las cadenas gráficamente por último puede convertir su AFD a una e.r. equivalente. En la figura 5 se muestra la interfaz del módulo.

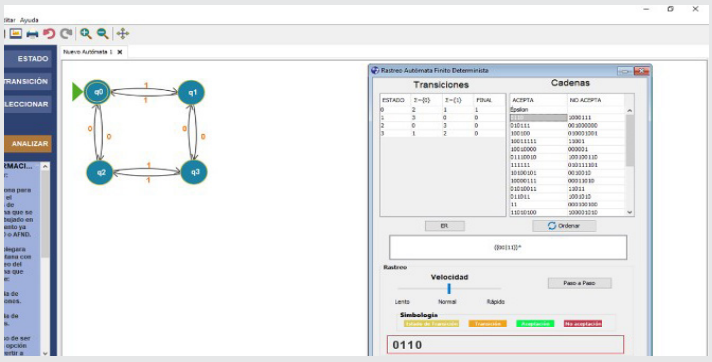

Figura 5. Interfaz del módulo AFD.

En el módulo AFND es similar al primer módulo descrito, a diferencia que en la venta de resultados nos indica el tipo de autómata y la conversión que se puede realizar es la de un AFND a un AFD equivalente, lo anterior se puede observar en la figura 6.

En el módulo e.r.'s el usuario puede crear una expresión regular y analizarla, para esta estructura solo se muestran las cadenas aceptadas y rechazadas, por último, el usuario puede convertir la expresión regular a un AFD equivalente, la interfaz de este módulo se observa en la figura 7. Funciones:

- Construcción de diagramas de tipo AFD
- Construcción de diagramas de tipo AFND
- Construcción de e.r.´s
- Conversión de AFD - e.r.´s
- Conversión de AFND - AFD
- Conversión de e.r.´s - AFD

Rastreo de cadenas automático para diagra mas AFD y AFND
Rastreo de cadenas visual para diagramas AFD y AFND
Rastreo de cadenas automático para e.r.´s

Una vez aplicadas las encuestas en los momentos diferentes (recabar información de impacto para cada uno de los modelos que se incluyen en la herramienta), realizadas las gráficas y tablas, se obtuvieron los siguientes resultados relevantes: a) un significativo 85% consideraron mejor herramienta a JLEFO que otras utilizadas para el mismo fin; b) más del 75% de los encuestados consideraron un apoyo invaluable la facilidad del rastreo de cadenas (que no tienen otras herramientas similares); c) casi el 90% de los encuestados manifestaron estar satisfechos con la utilización de JLEFO; d) el 92% dijeron que la herramientas les permitió una mejor comprensión de los modelos matemáticos planteados.

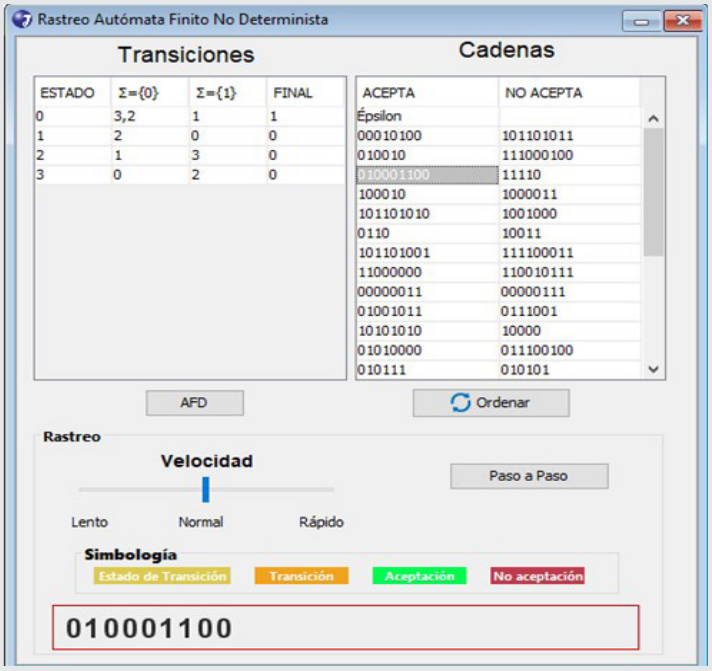

Figura 6. Interfaz de rastreo del módulo AFND.

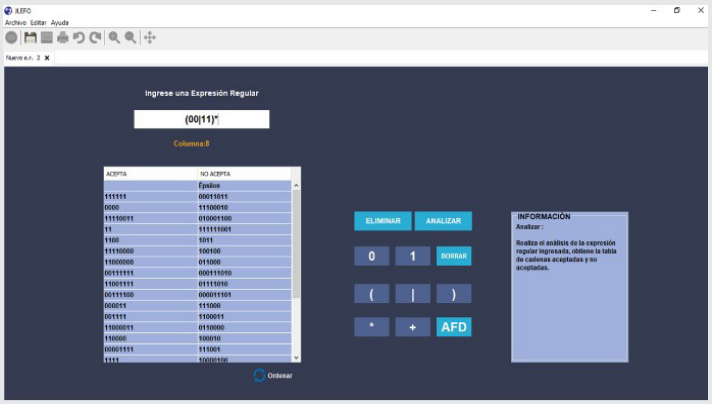

Figura 7. Interfaz del módulo e.r.'s.

CONCLUSIONES

Para recabar información del impacto de JLEFO en la materia Lenguajes y Autómatas I del periodo enero - junio 2019, especialmente en relación con las estructuras que se plantean en la herramienta, se proporcionó el ejecutable de la aplicación a 39 estudiantes, con el fin de que lo utilizarán en la resolución de ejercicios, además de familiarizarse con la herramienta, el formato de la representación del resultado de análisis de las estructuras y con las demás funcionalidades.

Durante el curso una vez vistas las unidades correspondientes a autómatas finitos y expresiones regulares se aplicó un examen ordinario tradicional con resolución de ejercicios de AFD y e.r.´s en papel. Posteriormente se calificaron obteniendo así un 18% de aprobados, lo cual reflejó un índice muy bajo de aprobación. Paso siguiente se procedió a preparar una serie de ejercicios que conformaron un nuevo examen en el cual se evaluó el mismo tipo de estructuras AFD y e.r´s, pero esta vez se resolvieron utilizando la herramienta JLEFO. Los resultados obtenidos fueron favorables, el índice de aprobación fue de un 77%. Comparando los resultados se obtuvo la siguiente información que se muestra en la figura 8.

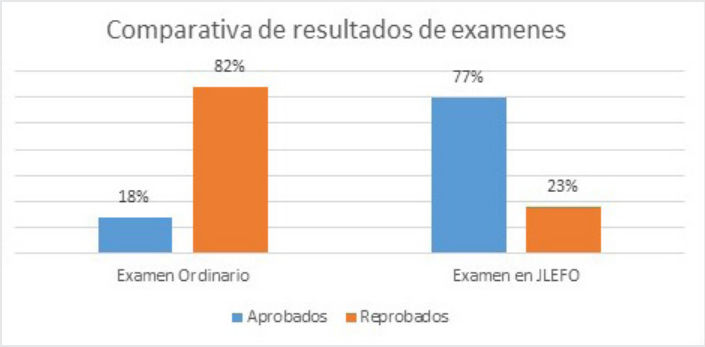

Figura 8: Comparativa de resultados de evaluaciones con y sin la herramienta JLEFO.

En base a todo lo anterior se concluye que el desarrollo de la aplicación JLEFO resultó un éxito en cuanto a funcionalidad y objetivos que se buscaban cumplir. Las ventajas de la herramienta y que se logran a través de la interfaz más agradable visualmente y fácil de entender, le permite al estudiante realizar el rastreo automático de cadenas, presentando la tabla de transiciones y la de cadenas, teniendo una reducción del tiempo que le toma realizarlo manualmente, además de tener un rango más amplio de cadenas y de longitud variable permitiendo que la interpretación del lenguaje que aceptaba su estructura fuera más fácil que hacerlo con sus rastreos manuales. Para los gustos visuales que tienen los estudiantes se proporcionó un rastreo de determinada cadena en el módulo AFD/AFND, esto también ayudó a comprender cómo funciona un diagrama de transiciones al leer una cadena y también para los que no es de su agrado solo observar la tabla de cadenas.

Finalmente, uno de los puntos fuertes y de gran interés fueron las conversiones entre estructuras, siguiendo el siguiente orden:
- AFD - e.r.
- AFND - AFD
- e.r. - AFD

A través de la repetición de esta acción de conversión permitió al alumno entender cómo generar e.r´s simplificadas y eficientes.

Por el lado de construcción del diagrama, muchas veces los estudiantes no completan su diagrama y aunque no es explícito que la aplicación refleje a través de un cuadro de diálogo el tipo de estructura que se haya dibujado, al generar un análisis en la ventana de rastreo el título indica el tipo de estructura al que se le realiza, dado que AFD y AFND pueden ser diagramados en el mismo espacio de trabajo, pero este título ayudó a remediar esta habitual equivocación.

Por último, esta herramienta fue registrada ante INDAUTOR (Instituto Nacional del Derecho de Autor) por la propiedad intelectual con número de registro 03-201 9-0621 09365000-01.

BIBLIOGRAFÍA

[1] Extreme Programming: A gentle introduction. (n.d.). Recuperado el 6 de septiembre de 2019 de http://www.extremeprogramming.org/

[2] Meléndez, S., et. al (2015). Metodología ágil de desarrollo de software programación extrema. Universidad Nacional Autónoma de Nicaragua, Managua.

[3] Modelo Vista controlador (MVC). (n.d.). Recuperado 6 de septiembre 2019 de https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html

[4] Chhabra, T. (2012). Conversion of deterministic finite automata to regular expression. Thapar University, Patiala.

[5] Neumann, C. (2005). Converting Deterministic Finite Automata to Regular Expressions. Recuperado de https://liacs.leidenuniv.nl/~bonsanguemm/FI2/DFA_to_RE.pdf

[6] Ruiz L., Edgar, Raffo L., Eduardo, (2003). Conversión de un AFN a un AFD. Industrial Data. ISSN 1560-9146.

[7] Aho, Alfred V. (2008). Compiladores principios, técnicas y herramientas (2da ed.). México, PEARSON EDUCACIÓN.