Una herramienta de apoyo en la enseñanza de Teoría de Autómatas y Lenguajes Formales *

Román Sosa y Francisco de Sande
Dep. de Estadística, I.O. y Computación
ETS Ingeniería Informática
Universidad de La Laguna
38271 La Laguna
{alu1970,sande}@etsii.ull.es

Resumen

La herramienta *ToTalf* (Tutorial Online para Teoría de Autómatas y Lenguajes Formales) ha sido diseñada con la finalidad de apoyar la realización de prácticas en la asignatura de *Teoría de Autómatas y Lenguajes Formales*. Las prácticas de laboratorio constituyen una parte esencial de esta asignatura puesto que en los planes de estudios de la Universidad de La Laguna la mitad de sus créditos son de tipo práctico.

ToTalf permite al alumnado experimentar con los conceptos estudiados en las clases de teoría, a través de la ejecución de distintos algoritmos que se aplican sobre los diferentes tipos de modelos de computación que se estudian en las clases teóricas. La herramienta ha sido desarrollada en Java, y está disponible tanto en forma de aplicación como de applet, de modo que los alumnos pueden instalarla localmente en sus ordenadores o bien utilizarla a través de cualquier navegador. Dispone de un sistema de ayuda que facilita el uso de la interfaz de usuario y para cada uno de los diferentes modelos de computación contemplados se ofrecen al usuario ejemplos predefinidos que se pueden cargar directamente y que también ilustran el modo de operación de la herramienta.

*El proyecto ToTalf ha sido financiado por el Vicerrectorado de Calidad Docente y Nuevos Estudios de la Universidad de La Laguna en la II Convocatoria de Proyectos de Innovación Docente

Uno de los objetivos esenciales en el diseño de *ToTalf* ha sido que fuera una aplicación abierta de modo que en el futuro sea fácil incorporar nuevas funcionalidades, ejemplos, así como modificar o completar el sistema de ayuda.

1. Introducción

El proyecto *ToTalf* [6] es un sistema de ayuda para el aprendizaje de conceptos básicos en la materia de autómatas y lenguajes formales. Concretamente, ha sido desarrollado para los alumnos que cursan la asignatura de *Teoría de Autómatas y Lenguajes Formales* (TALF) [3] en la Universidad de La Laguna, aunque algunas de sus capacidades son también aprovechables en la asignatura de *Compiladores*. El desarrollo del proyecto tuvo una duración aproximada de seis meses y el primer prototipo fue implantado al final del curso académico 2003-2004.

ToTalf aborda el estudio de la asignatura mediante un enfoque práctico, dividiendo el temario de la asignatura en cuatro módulos principales.

1. Autómatas finitos y expresiones regulares
2. Gramáticas independientes del contexto
3. Autómatas a pila
4. Máquinas de Turing

En cada uno de estos módulos, el usuario puede introducir un ejemplo del concepto correspondiente, y aplicarle los distintos algoritmos

Actas del SImposio Nacional de Docencia en la Informática, SINDI2005 (AENUI), pp.181-188
ISBN: 84-9732-443-9

que se encuentran implementados, y que se estudian en la asignatura. De esta forma, el alumno puede aplicar directamente los conocimientos y algoritmos estudiados en las clases de teoría, en vez de tener que resolver los problemas usando lápiz y papel, lo que consume demasiado tiempo, es propenso a los errores, y sumamente tedioso.

En la actualidad es posible encontrar herramientas de características similares a las de *ToTalf*. Es el gran auge de Internet quien ha permitido la proliferación de programas de este tipo. Cada uno suele tener alguna propiedad interesante, pero la carencia de otras los hacen ser herramientas deficientes para nuestros propósitos. En general, la mayoría se centra en un único aspecto: la simulación de autómatas finitos. En el caso de las aplicaciones desarrolladas fuera de España, el hecho de que todos los menús así como el sistema de ayuda esté en inglés, desanima a algunos alumnos a utilizarlas. Revisemos a continuación las aplicaciones más relevantes:

- JFLAP [9] es un paquete de herramientas gráficas diseñado como asistente en la enseñanza de conceptos básicos en TALF. Se trata posiblemente de la herramienta más sofisticada y evolucionada de este tipo. Con respecto a *ToTalf*, la característica más notable es que dispone de una interfaz gráfica que permite al usuario introducir autómatas y visualizar la salida de los diferentes algoritmos.
- ABCEZ (Applet Based Computability Enlightenment Zone) [10] es otra de las herramientas basadas en applets con mayor impacto visual. Igual que JFLAP también posee una cuidada interfaz gráfica, pero en este caso centra su atención exclusivamente en la simulación de autómatas finitos. Una de las características más destacables de este programa es su sistema de ayudas.
- El applet diseñado por J. Calera [1] para la asignatura *Lenguajes, Gramáticas y Autómatas* en la Universidad de Alicante es uno de los pocos ejemplos disponibles de herramientas similares a *ToTalf* en el contexto español. Se centra también en algoritmos orientados a autómatas finitos, y al igual que en nuestro trabajo, en lugar de invertir esfuerzo en el diseño de una interfaz gráfica avanzada, se ha optado por implementar un amplio conjunto de algoritmos.

En las referencias pueden encontrarse otros trabajos [2, 5, 4], sobre simulación de autómatas finitos, y que no aportan grandes diferencias respecto a los ya comentados en los puntos anteriores.

2. Contexto y Objetivos

En la *Escuela Técnica Superior de Ingeniería Informática (ETSII)* de la Universidad de La Laguna se imparten los estudios correspondientes a las titulaciones de Ingeniero Técnico en Informática de Sistemas (ITIS) y de Gestión (ITIG), así como los de Ingeniero en Informática.

La asignatura de TALF figura como troncal en ITIS y como obligatoria en ITIG, y tiene asignados un total de nueve créditos, repartidos en partes iguales entre teoría y prácticas. La asignatura se imparte a lo largo del primer cuatrimestre del segundo curso.

TALF es una asignatura con unos contenidos eminentemente abstractos que se han desarrollado históricamente a partir de los problemas concretos a los que se ha ido enfrentando la comunidad científica en su intento de utilizar lenguajes de alto nivel en la programación de ordenadores, de traducir automáticamente o de matematizar el lenguaje humano para procesarlo automáticamente.

La repercusión más directa de los contenidos de TALF, quizás se produce en las asignaturas del plan de estudios relacionadas con compiladores: *Compiladores* en las ingenierías técnicas, así como *Procesadores de Lenguajes* en la Ingeniería en Informática. Aparte de esta repercusión directa, los contenidos de TALF también son fundamentales en muchos otros campos y asignaturas estudiados en la carrera.

2.1. Objetivos generales de TALF

Con esta asignatura se pretende transmitir al alumno los conceptos fundamentales de la *Teoría de Autómatas y Lenguajes Formales*, presentándole una visión global de la misma y profundizando principalmente en los aspectos más aplicados como son las herramientas básicas que se utilizan en el diseño de analizadores léxicos y sintácticos a partir de expresiones regulares y gramáticas independientes del contexto, respectivamente. Los objetivos principales de la asignatura son:

- Presentar los distintos métodos formales de representación de lenguajes, estudiando métodos generativos (gramáticas) y métodos por aceptación (autómatas).
- Conocer las herramientas que se utilizan para describir, reconocer y caracterizar dichos lenguajes.
- Desarrollar la capacidad de abstracción y análisis teórico en relación con la teoría de lenguajes.

2.2. El programa de la asignatura

El temario de teoría impartido en la asignatura es el siguiente:

1. Introducción al lenguaje C++.
2. Conceptos básicos
3. Lenguajes regulares y autómatas finitos.
4. Gramáticas. Lenguajes independientes del contexto. Autómatas a pila.
5. Máquinas de Turing. Lenguajes recursivos y recursivamente enumerables.
6. Resolubilidad.

Sin desdeñar los contenidos teóricos, indispensables para el desarrollo de cualquier técnica, consideramos que los estudios de Informática son eminentemente prácticos y es por ello que en las asignaturas que impartimos, las prácticas con ordenador constituyen un aspecto fundamental de las mismas como complemento insustituible a las clases de teoría. Semanalmente los alumnos presentan en una sesión de corrección de prácticas una de las prácticas cuyo enunciado se les ha propuesto con antelación. Los contenidos de las diferentes prácticas pueden variar de un año a otro, pero como es natural, siempre están íntimamente relacionadas con los temas que se estudian en teoría.

Los objetivos docentes a alcanzar a través de las prácticas de la asignatura los podemos resumir en:

1. La consolidación a través de las experiencias de laboratorio de los conceptos adquiridos en las clases teóricas.
2. Introducir a los estudiantes en un lenguaje de programación orientado a objetos, que es nuevo para ellos (durante el primer curso de la carrera los alumnos desarrollan sus prácticas usando Pascal).
3. Mejorar el dominio de la programación adquirido en el curso anterior en las asignaturas de *Metodología y Tecnología de la Programación I y II*.

2.3. Objetivos del Proyecto ToTalf

El proyecto *ToTalf* surge como una necesidad del profesor para mejorar la enseñanza de la asignatura. Para este fin, *ToTalf* implementa la totalidad de los algoritmos que se programan en las prácticas (hay que tener en cuenta que las prácticas de la asignatura cambian en cada curso), y muchos otros algoritmos interesantes estudiados en las clases de teoría, incluyendo algunos correspondientes a la asignatura de *Compiladores*. Revisamos a continuación los objetivos que perseguía *ToTalf* en su planteamiento inicial.

Como ya se comentó en la introducción, trata de ser una herramienta para ayudar al alumno en la comprensión de la teoría y los conceptos relacionados con la asignatura. El poder experimentar con dichos conceptos mediante la aplicación de algoritmos debe ser una actividad que facilite esa comprensión. Gracias a *ToTalf*, el alumno puede comprobar directamente si ha realizado de manera correcta un ejercicio teórico. Además, en el caso de problemas de mayor tamaño, el alumno invierte su tiempo en el análisis de los resultados de los algoritmos, en vez de utilizarlo en realizar trazas sobre papel.

ToTalf se diseña como asistente en la realización de prácticas. Los alumnos pueden experimentar con los algoritmos antes de acometer

las prácticas que deben realizar. Pero fundamentalmente, es un marco de comprobación. En general, los alumnos dedican mucho tiempo a la implementación de las prácticas, pero poco a la comprobación de los resultados. Se suelen conformar con comprobar su código con los ejemplos que se le han suministrado, lo cual a veces resulta insuficiente. Con *ToTalf* se pretende que el alumno tenga de una manera sencilla una forma de examinar los resultados que devuelve su programa.

También la herramienta es útil en la realización de prácticas de cierta envergadura, en las que haya que calcular ciertos resultados intermedios, los cuales no son objetivo de estudio de la práctica.

Por último podemos considerarla una herramienta interesante para aquellos alumnos curiosos que gusten de experimentar por su cuenta.

3. ToTalf: la herramienta

3.1. Implementación

ToTalf está programado en Java. La elección de Java como lenguaje de programación responde a la gran ventaja de su ejecución multiplataforma, que permite su uso en multitud de entornos, con la ventaja de poder ejecutarse de manera sencilla y directa en un navegador. Concretamente, *ToTalf* está programado en Java 1.4, y se proporciona tanto en versión applet como en versión aplicación.

Para la ejecución del applet (es decir, la ejecución a través del navegador) es necesario un navegador con soporte para la máquina virtual Java de Sun [7]. Por lo tanto, se necesita disponer del paquete J2RE (entorno de ejecución de Java) o el paquete J2SDK (entorno de desarrollo de Java), disponibles ambos en la sección de descargas.

Sin embargo, para un uso cotidiano, se recomienda la versión aplicación, que no está afectada por las restricciones de seguridad impuestas a los applets. Con la versión aplicación podremos cargar y guardar archivos, y copiar y pegar entre aplicaciones y el applet, o viceversa. Para la ejecución de la aplicación, es necesario, al igual que antes, J2RE o J2SDK, y también descargar el fichero `jar` del proyecto.

Pasemos a estudiar los principales requisitos que se impusieron a la aplicación a desarrollar. El primero es el de disponer de un sistema de ayuda fácilmente modificable para que el usuario pueda consultar cómo operar con la herramienta. Para cumplirlo se optó por realizar la ayuda de *ToTalf* en formato *HTML*. Se trata de una solución estándar ampliamente utilizada, ya que los ficheros de ayuda se puede crear y modificar fácilmente, más aún teniendo en cuenta la multitud de herramientas disponibles en este campo.

Pretendíamos también que la herramienta admitiese con facilidad la incorporación de nuevos algoritmos. En este sentido se puso un especial esfuerzo en el diseño de los objetos que representan los diferentes modelos de computación para permitir la escalabilidad del programa. Cada objeto dispone de abundantes métodos para modificar y acceder a la estructura interna del mismo, lo que simplifica la implementación de nuevos algoritmos.

También fue un requisito que la aplicación admitiese la incorporación de nuevos ejemplos predefinidos para cada modelo de computación. A través de la redefinición de una tabla es posible añadir nuevos ejemplos predefinidos. Cada ejemplo se codifica del mismo modo que se hace en la interfaz de usuario, lo cual facilita la modificación de dicha tabla.

El último de los requisitos del proyecto fue que se desarrollara sobre Linux, en un entorno de desarrollo de código abierto, ya sea en modo texto o gráfico. A la vez, el entorno de desarrollo debía ser de fácil uso, y permitir de forma sencilla la modificación del código fuera de él. Se optó por la herramienta de desarrollo JBuilder8 Personal [8] puesto que se ajusta a estos parámetros y está disponible gratuitamente para usos no comerciales.

3.2. Utilizando ToTalf

El tutorial se compone de cuatro grandes módulos que abarcan los diferentes aspectos abordados en la asignatura: automatas finitos y expresiones regulares, gramáticas independientes

del contexto, autómatas a pila y máquinas de Turing.

En cada módulo el usuario puede introducir una descripción textual del concepto correspondiente. Dicha descripción es igual a la que se utiliza en los enunciados de las prácticas, para así facilitar el uso del programa a los alumnos. Cada descripción puede guardarse en un fichero para su uso posterior, o también puede cargarse una descripción almacenada previamente. Se dispone también de la opción de cargar ejemplos predefinidos, representativos del concepto en cuestión.

A la descripción introducida, se le pueden aplicar los algoritmos implementados, que se corresponden con la totalidad de los algoritmos programados en las prácticas de la asignatura más algunos otros interesantes estudiados en clase.

Describiremos a continuación cada uno de los módulos que componen *ToTalf*.

- Autómatas finitos y expresiones regulares. En este módulo, se contemplan autómatas finitos deterministas (AFD) y no deterministas (AFN), al igual que un subconjunto de las expresiones regulares extendidas. Se encuentran disponibles los siguientes algoritmos:
  - Transformación de AFN a AFD.
  - Eliminación de ϵ-transiciones.
  - Minimización de un AFD.
  - Obtención de un AFN a partir de una expresión regular (Construcción de Thompson).
  - Obtención de una gramática regular a partir de un AFD.
  - Obtención de los AFNs que reconocen $L(M_1) \cup L(M_2)$ y $L(M_1)L(M_2)$ a partir de los autómatas M_1 y M_2.

  En las expresiones regulares, se reconocen los operadores básicos (alternativas, concatenaciones, asociaciones y cierre), junto a varias características de las expresiones regulares extendidas: conjuntos de caracteres (p. ej. `[abc]`, `[a-c]`), el cierre positivo, el operador de interrogación y algunas clases de conjuntos de caracteres (p. ej. `[:alpha:]` y `[:digit:]`).
- Gramáticas independientes del contexto (GIC). En este módulo, los algoritmos implementados son:
  - Cálculo de los conjuntos *first* y *follow*.
  - Algoritmo de Cocke–Younger–Kasami para el análisis sintáctico de una frase.
  - Cálculo de la forma Normal de Chomsky.
  - Simplificación de una GIC.
  - Eliminación de la recursividad por la izquierda de una GIC.
  - Transformación de una gramática regular lineal por la derecha a un AFN.
- Autómatas a pila deterministas. En este módulo el único algoritmo disponible es el de la simulación de un autómata a pila determinista.
- Máquinas de Turing. Sólo se ha implementado la simulación de una máquina de Turing con cinta infinita en ambos sentidos.

```
// gramática que genera expresiones
// regulares simples
ER -> ER + ALT
ER -> ALT
ALT -> ALT PIEZA
ALT -> PIEZA
PIEZA -> ATOMO
PIEZA -> ATOMO *
ATOMO -> t
ATOMO -> ( ER )
```

Figura 1: Descripción de una gramática

3.3. Descripciones

En cada uno de los módulos de *ToTalf*, la entrada a los diferentes algoritmos viene dada por una descripción del modelo de computación correspondiente. Las descripciones tratan de ser lo más general y flexible posible. Así, por ejemplo, el número de caracteres prohibidos intenta ser mínimo, se permite el uso de comentarios en las descripciones, o en el caso

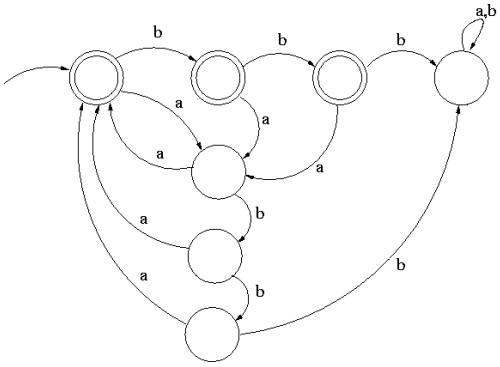

Figura 2: Autómata que reconoce cadenas con un n^{o} par de 'a' y que no contengan la subcadena 'bbb'

de las gramáticas, los símbolos pueden constar de más de un carácter.

Como ejemplo, la Figura 1 muestra el contenido de un fichero que describe una gramática que genera expresiones regulares.

3.4. Un ejemplo de uso con autómatas

Supongamos que el alumno desea resolver el siguiente problema: *Dado el alfabeto* $\Sigma = \{a, b\}$ *hallar un autómata finito con un número mínimo de estados que reconozca las cadenas de* Σ^* *que contengan un número par de símbolos 'a' y no contengan la subcadena 'bbb'.*

Cuando se les plantea este problema a los estudiantes, la mayoría obtienen directamente la solución, que se muestra en la Figura 2.

Sin embargo muchos otros alumnos tratan de obtener la solución siguiendo un proceso alternativo, en el que *ToTalf* es una herramienta muy útil a la hora de simplificar algunos de los cálculos que han de realizarse.

Para resolver este problema, puede resultar cómodo partir de dos autómatas que reconozcan los siguientes lenguajes:

$L_1 = \{w \in \Sigma^* | w$ *contiene un número par de símbolos 'a'*$\}$

$L_2 = \{w \in \Sigma^* | w$ *no contiene la subcadena 'bbb'*$\}$

Los AFD que reconocen L_1 y L_2 son los que aparecen en la Figura 3.

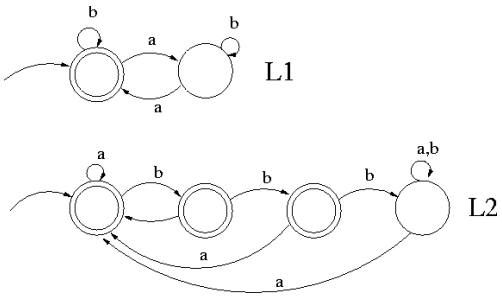

Figura 3: Autómatas que reconocen L_1 y L_2

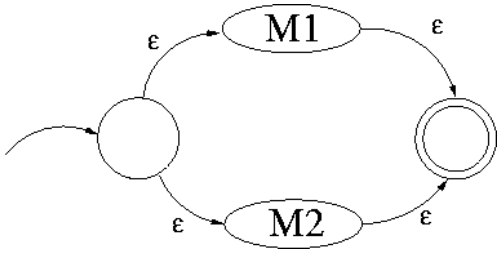

Figura 4: Autómata que reconoce $L(M_1) \cup L(M_2)$

El autómata que se nos pide puede obtenerse teniendo en cuenta la siguiente relación entre lenguajes, consecuencia de la aplicación de una de las leyes de Morgan:

$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$

Dado un AFD M que reconoce un lenguaje L, para obtener el AFD que reconoce $\overline{L}$ basta con cambiar en M los estados que son de aceptación por estados que no sean de aceptación y convertir en estados de aceptación aquellos que en M no lo eran.

Por otra parte, partiendo de las descripciones de los autómatas M_1 y M_2 que reconocen $\overline{L_1}$ y $\overline{L_2}$ respectivamente, y utilizando uno de los algoritmos implementados en *ToTalf*, podemos obtener el AFN que reconoce $\overline{L_1} \cup \overline{L_2}$. La Figura 4 muestra el fundamento teórico de esta construcción. Si aplicamos a este autómata el algoritmo de conversión de un AFN en AFD y al resultado le aplicamos la minimización del número de estados de un AFD, obtendremos el AFD que se muestra en la Figura 2, que es la solución del problema planteado.

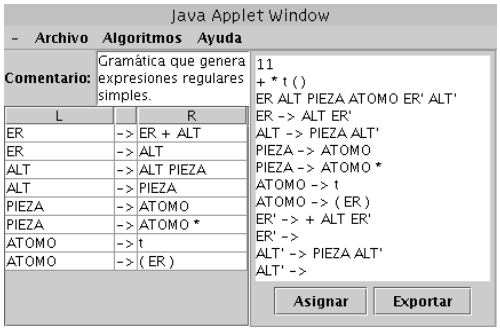

Figura 5: Gramática de expresiones regulares sin recursividad por la izquierda

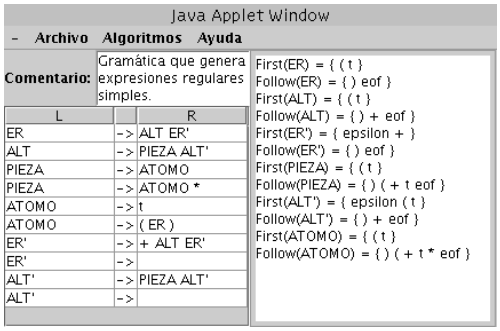

Figura 6: Conjuntos *First* y *Follow* de la gramática de expresiones regulares

3.5. Un ejemplo de uso con gramáticas

Supongamos otro caso real de trabajo: construir un analizador sintáctico descendente recursivo predictivo para la gramática representada en la Figura 1.

El primer paso consiste en introducir la gramática y aplicar el algoritmo de eliminación de la recursividad por la izquierda. Tras seleccionar el algoritmo adecuado en el menú *Algoritmos*, obtenemos el resultado en el marco de la derecha, tal y como se muestra en la Figura 5.

El siguiente paso sería comprobar que con esta gramática sin recursividad por la izquierda se puede construir un analizador descendente recursivo predictivo. Para ello, se debe cumplir que:
$\forall A \in V | A \rightarrow \alpha_i | \alpha_j$:
$First(\alpha_i) \cap First(\alpha_j) = \emptyset, \forall i \neq j$
Si $\alpha_i \Rightarrow^* \epsilon$, entonces
$First(\alpha_j) \cap Follow(A) = \emptyset, \forall j$
No existe actualmente un algoritmo implementado en *ToTalf* que realice esta comprobación, por lo que habría que hacerla a mano. No obstante, sí está implementado el cálculo de conjuntos *first* y *follow* de una gramática, por lo que la tarea es mucho más sencilla. Los conjuntos de *first* y *follow* se muestran en la Figura 6. Tras la correcta comprobación de que un analizador descendente recursivo predictivo puede funcionar con la gramática, el alumno puede empezar a codificar dicho analizador.

4. Conclusiones y trabajo futuro

La valoración que hacemos de los resultados del primer año de utilización de *ToTalf* en la docencia de *TALF* es absolutamente positiva. Las encuesta que hemos realizado entre el alumnado refleja que una mayoría de ellos han utilizado la herramienta y también mayoritariamente quienes la han utilizado están satisfechos con los resultados obtenidos. Para el próximo curso, una vez cubierta la etapa en la que la primera versión de *ToTalf* ha sido probada satisfactoriamente, planeamos utilizarla de forma más intensiva en el desarrollo de los contenidos de la asignatura.

Nos gustaría también destacar que *ToTalf* está siendo utilizado no sólo por los alumnos matriculados en *TALF* sino también por los alumnos de *Compiladores*, asignatura cuyos contenidos están muy cercanos a los desarrollados en *TALF*.

Por otra parte consideramos que el trabajo realizado hasta ahora debe ser considerado como un primer prototipo de la herramienta. Nos gustaría poder seguir perfeccionándola y sobre todo añadiéndole funcionalidades. La lista de posibles ampliaciones es tan amplia como los contenidos de las asignaturas de *TALF* y *Compiladores*, pero en concreto, tenemos ya en mente las siguientes extensiones:

- Obtención del autómata finito (AF) que reconoce el lenguaje complementario del reconocido por otro AF.

- Equivalencia de AFs.
- Simulación de Autómatas a pila no deterministas (APND).
- Obtención del APND asociado con una gramática independiente del contexto (GIC).
- Cálculo de la tabla LL(1) para una GIC.
- Obtención de la GIC que genera el lenguaje reconocido por APND.
- Simulación de máquinas de Turing con varias cintas.

Pero sin duda, una de los aspectos más interesantes a incluir es una interfaz gráfica para la definición de los autómatas, en vez del actual esquema de texto. La creación de autómatas sería más intuitiva, y el análisis de los autómatas resultado de los algoritmos, mucho más sencillo, lo que conseguiría un mejor entendimiento por parte del alumno.

Tal como ha sido concebido, el prototipo actual de *ToTalf* no es más que el núcleo básico de una herramienta que tiene toda la potencialidad de proseguir su desarrollo. En el caso de herramientas similares, lo único que podemos hacer es recomendar a nuestros alumnos su utilización, mientras que *ToTalf* es un prototipo sobre el que tenemos control total. También estamos considerando la posibilidad de liberar el código fuente de la aplicación, haciéndolo de dominio público, de forma que toda la comunidad académica de habla hispana pueda contribuir al desarrollo de la herramienta.

En resumen, consideramos satisfactorio el esfuerzo invertido en este proyecto de innovación docente. Pensamos que *ToTalf* es en estos momentos una de las herramientas de estas características más completas desarrollada en España, y que se ajusta plenamente a los objetivos con los que fue concebida.

Referencias

[1] J. Calera. *Applet Java para realizar diversas operaciones con autómatas finitos.* Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Alicante. Disponible electrónicamente en `http://www.dlsi.ua.es/asignaturas/lga/applet/Afapplet.html`.

[2] Nicolas Christin. DFApplet a deterministic finite automata applet simulator. Disponible electrónicamente en `http://www.cs.virginia.edu/~nc2y/dfa/`.

[3] Francisco de Sande. *Páginas web de Teoría de Autómatas y Lenguajes Formales.* Escuela Técnica Superior de Ingeniería Informática. `http://asignaturas.pcg.ull.es/etsii/talf/`.

[4] Thomas Dunn. Interactive animation of finite-state automata. Disponible electrónicamente en `http://www.ccs.neu.edu/home/tdunn/COM1350/honors/`.

[5] Dynalab. Finite State Automaton Applet. Disponible electrónicamente en `http://www.cs.montana.edu/~dynalab/fsa/fsa.html`.

[6] Román Sosa González. ToTalf tutorial online de teoría de autómatas y lenguajes formales. Disponible electrónicamente en `http://asignaturas.pcg.ull.es/etsii/talf/ToTALF/Applet_ToTalf.html`.

[7] Java. Página principal: `http://java.sun.com`.

[8] Jbuilder downloads. Disponible electrónicamente en `http://www.borland.com/products/downloads/download_jbuilder.html`.

[9] Susan H. Rodger. JFLAP 4.0b10 version. Disponible electrónicamente en `http://www.cs.duke.edu/~rodger/tools/jflap/index.html`.

[10] MindSpring Software. ABCEZ (applet based computability enlightenment zone). Disponible electrónicamente en `http://www1.cs.columbia.edu/~zeph/ComputabilityApplets/abcez.html`.