Enseñanza asistida de teoría de autómatas y lenguajes formales mediante el uso de THOTH

César García Osorio, Andrés Arnáiz Moreno, Álvar Arnáiz González
Departamento de Ingeniería Civil
Universidad de Burgos
Avda. Cantabria s/n, 09006, Burgos
cgosorio@ubu.es, arnaizmoreno@hotmail.com, aagcaballeo@hotmail.com

Resumen

La teoría de autómatas y lenguajes formales suele ser percibida como árida y aburrida por los estudiantes. El uso de numerosos ejemplos es indispensable para facilitar la compresión de los conceptos. El problema es que estos suelen ser tediosos de escribir a mano o excesivamente simples. Thoth es una herramienta que pretende asistir en la docencia de los cursos de autómatas y lenguajes formales proporcionando al alumno una forma de interaccionar con conceptos y algoritmos. En su desarrollo se ha priorizado dos aspectos, la facilidad de utilización, y la reutilización de su código como applets que puedan integrarse en la web. En esta su primera versión se centra en los autómatas finitos, las expresiones regulares y las gramáticas independientes del contexto.

1. Introducción

Uno de los principales problemas de la enseñanza de las asignaturas de teoría de autómatas y lenguajes formales es la percepción que tienen los alumnos de la misma como demasiado compleja y teórica. Esta percepción se puede aliviar con la abundante resolución de ejercicios y el uso de ejemplos. Desafortunadamente estos son o muy simples o largos y tediosos de hacer a mano. Además los alumnos están acostumbrados a usar el ordenador en otras asignaturas. Aquí presentamos Thoth, una herramienta de apoyo a la docencia de teoría de autómatas y lenguajes formales. Thoth permite la simulación de los dispositivos de esta teoría y la ejecución paso a paso de sus algoritmos. Inicialmente estaba pensada para trabajar exclusivamente con autómatas finitos y expresiones y gramáticas regulares, pero en su versión actual también incluye algoritmos para gramáticas independientes del contexto, tales como la obtención de la forma normal de Chomksy, la generación de tablas de análisis LL(1) y el algoritmo de Cocke-Younger-Kasami. Existen otras herramientas que cubren un espectro más amplio de dispositivos y algoritmos, la ventaja de Thoth frente a éstas es que esta en castellano y que presenta una muy cuidada interfaz gráfica. La edición de un autómatas finito se ha simplificado enormemente, es posible dibujar y ubicar todos sus elementos, estados y transiciones, con simples operaciones de ratón sin necesidad de seleccionar previamente en un menú o botón el elemento a dibujar. Además, muestra mucha más información en pantalla sobre la estructura del autómata.

Antes de dar más detalles sobre la funcionalidad que Thoth proporciona, la sección que sigue da un rápido repaso a los principales conceptos de la teoría de lenguajes y autómatas. A continuación enumeramos y describiremos algunas otras herramientas que como Thoth asisten en la enseñanza de dichos conceptos. La siguiente sección detalla los algoritmos y operaciones con los que podemos interaccionar en Thoth. El artículo finaliza con una sección de conclusiones y de trabajo futuro.

2. Conceptos de la teoría de autómatas y lenguajes formales

Los tres conceptos fundamentales con los que se trabaja en THOTH son: gramática independiente del contexto, expresión regular y autómata finito. En esta sección explicamos brevemente en que consisten.

Las *gramáticas formales* permiten describir lenguajes. Desde un punto de vista formal, los *lenguajes* no son más que un conjunto de cadenas construidas utilizando un conjunto de símbolos de un alfabeto. Las gramáticas dan las reglas de cómo formar las cadenas que constituyen el

lenguaje. Formalmente una gramática esta constituida por los siguientes elementos, el alfabeto sobre el que construye las cadenas, un conjunto de símbolos auxiliares denominados *símbolos no terminales* (en contraste los elementos del alfabeto se llaman *símbolos terminales*), un conjunto de reglas de reescritura, llamadas *producciones* (indican como transformar unas cadenas en otras) y un símbolo especial dentro del conjunto de símbolos no terminales denominado *axioma* de la gramática, que inicia el proceso generativo. Así el lenguaje que genera una gramática esta constituido por las cadenas de símbolos del alfabeto que podemos derivar a partir del axioma aplicando las transformaciones definidas en las reglas o producciones. Cualquier subcadena que aparezca en la parte izquierda de una regla se puede sustituir por su parte derecha. Una gramática con las siguientes reglas { $S \rightarrow aSB$ (1), $S \rightarrow a$ (2), $aB \rightarrow a$ (3), $aB \rightarrow Ba$ (4), $B \rightarrow b$ (5)} (*S* es el axioma) permite generar todas las cadenas de aes y bes con un número de aes mayor que número de bes. Por ejemplo, la cadena *baaa*, se podría derivar del axioma con la secuencia $\underline{S} \Rightarrow^1$ a$\underline{S}$B $\Rightarrow^1$ *a*a$\underline{S}$B*B* $\Rightarrow^2$ *aa*a$\underline{B}$*B* $\Rightarrow^4$ *aa*B$\underline{a}$a*B* $\Rightarrow^4$ $\underline{B}$a*aaB* $\Rightarrow^3$ b*aa*$\underline{aB}$ $\Rightarrow^5$ *baa*a (en cada paso se muestra subrayada la subcadena substituida, en regular la nueva cadena y el superíndice índica la regla aplicada). La anterior secuencia de cadenas se denomina *derivación*, cada una de las cadenas obtenidas *forma sentencial*, y la cadena finalmente obtenida, formada exclusivamente por símbolos terminales, *palabra* o *sentencia*.

Dependiendo de la estructura de las producciones se establece una jerarquía de gramáticas y lenguajes. En THOTH sólo se utilizan las *gramáticas independientes del contexto*, los lenguajes que generan (o describen) estas gramáticas se denominan *lenguajes independientes del contexto*. En las gramáticas, independientes del contexto, las partes izquierdas de las reglas sólo pueden tener un símbolo no terminal. Un caso especial de gramáticas independientes del contexto son las *gramáticas regulares*. En estas gramáticas además se restringe a que en la parte izquierda sólo pueda aparecer un símbolo terminal, o dos símbolos el primero de ellos terminal y el segundo no terminal. Las gramáticas regulares permiten generar los llamados *lenguajes regulares*.

Una forma alternativa de describir los lenguajes regulares son las *expresiones regulares*. Los lenguajes regulares se caracterizan porque se pueden definir aplicando las operaciones de unión, concatenación y cierre a lenguajes que sean regulares. Las operaciones de unión y concatenación no son más que la unión y producto cartesiano de conjuntos. El cierre no es más que una concatenación repetida. Las expresiones regulares definen una sintaxis para describir estas operaciones. La unión se representa con el operador '|' (algunos autores usan el carácter '+' para representar este operador), la concatenación no tiene un operados explícito (algunos autores utilizan el carácter '·'), y el cierre con un operador sufijo '*'. De este modo la expresión regular $(b|c)(aa)^*a$ representa el lenguaje de las cadenas que comienzan por *b* o *c* y van seguidas de un número impar de aes.

En teoría de lenguajes formales se suele hacer una división entre dispositivos reconocedores de lenguajes y dispositivos descriptores o generadores. Las gramáticas y expresiones regulares pertenecen a esta segunda categoría. En la primera tenemos las máquinas de Turing, los autómatas de pila y los autómatas finitos. Mientras que las máquinas de Turing pueden reconocer cualquier lenguaje que pueda ser descrito por una gramática, los autómatas de pila sólo pueden reconocer los lenguajes independientes del contexto, y los autómatas finitos sólo los lenguajes regulares. En la versión actual de THOTH no se implementan ni las máquinas de Turing ni los autómatas de pila.

Los *autómatas finitos* constan de un conjunto de *estados*, en cada momento, sólo uno de ellos puede ser el estado actual, y de unas *transiciones* etiquetadas que nos indican como cambiar el estado actual dependiendo de cual sea el símbolo en curso. El autómata se puede representar como un grafo, cuyos nodos representan estados y cuyos arcos representan transiciones. Si partiendo del estado inicial, y siguiendo los arcos etiquetados con los símbolos que constituyen una cadena, logramos llegar a un estado etiquetado como final, la cadena se considera reconocida por el autómata. El conjunto de todas las cadenas que el autómata reconoce constituye el *lenguaje reconocido por el autómata*.

Esta sección ha dado una descripción necesariamente somera de conceptos, los libros de

Kelley [1], Brookshear [2] y Alfonseca [3] son excelentes para obtener descripciones más amplias.

3. Otras herramientas

En esta sección presentamos otras herramientas que pueden ser usadas para facilitar el aprendizaje de la teoría de autómatas y lenguajes formales. El principal inconveniente de la mayoría de ellas es que no se encuentras disponible en castellano.

- *Grail+* (http://www.csd.uwo.ca/Research/grail/ index.html): se trata de una colección de programas escritos en C++ para manipular autómata finitos y expresiones y lenguajes regulares. Estos programas se pueden utilizar como filtros en la línea de comandos o como una libraría cuya funcionalidad se puede incluir en otros programas C++. Por tanto carece de una interfaz gráfica de usuario propiamente dicha. En la actualidad es un proyecto abandonado, la última actualización de su página web se remonta al 2002.
- *Simple Java regular expression parser and recognized* (http://www.oursland.net/projects/regexp/): Se trata de un analizador de expresiones regulares en modo texto. Permite conocer si una cadena pertenece al lenguaje definido por una expresión regular. Además muestra por pantalla el autómata construido y el autómata mínimo resultante. No está orientado al aprendizaje de la materia. Al ser toda la información en modo texto no resulta demasiado atractivo de cara al usuario final.
- *The finite state machine explorer* (http://www.belgarath.org/java/fsme.html): Se trata de una aplicación que permite la simulación de un autómata finito. Disponible una versión de la aplicación, con capacidad para cargar y guardar en ficheros, y un applet para poder utilizarle en el navegador. Ofrece una pobre interfaz de usuario y dibujar el diagrama de transiciones para un autómata se hace muy pesado, pero es capaz de reconocer palabras iluminando los estados actuales en cada momento. Permite mover los estados y redibujarlos. No muestra apenas información por pantalla de las operaciones que realiza, está desarrollado para mostrar una solución al problema no para mostrar como se consigue. Es muy específico y no presenta grandes funcionalidades. Resulta muy lento y pesado el uso de su entorno gráfico, demasiados botones con funciones que pueden agruparse dentro del mismo botón.
- *Visual automaton simulator* (http://www.cs.usfca.edu/~jbovet/vas.html): Es un simulador de autómatas finitos y máquinas de Turing en modo gráfico. Permite guardar y salvar en disco los elementos creados. Es la primera herramienta encontrada con una interfaz de usuario algo compleja, muestra la construcción de las máquinas paso a paso. Permite validar cadenas tanto en autómatas finitos como máquinas de Turing. No implementa algoritmos importantes a la hora de trabajar con los autómatas, solo permite validar cadenas y eliminar el determinismo. Además para el reconocimiento de palabras es necesario especificar el alfabeto del autómata en lugar de calcularlo automáticamente. La introducción de nuevos estados (algo muy habitual) resulta muy costosa, esto hace que la aplicación resulte incómoda al usuario final. Los archivos guardados en disco resultan muy extensos, ocupando espacio innecesario para almacenar autómatas muy sencillos. Presenta un lienzo poco apto para autómatas complejos, debido a su escaso tamaño.
- *Java formal language and automata processor* (*JFLAP*) (http://www.jflap.org/): Se trata de una herramienta muy completa y bien documentada [4]. La cualidad más destacable de esta aplicación es la gran cantidad de operaciones que se pueden llevar a cabo. Puede trabajar con distinta tipos de máquinas: autómatas finitos, autómatas de pila y máquinas de Turing. En su última versión permite utilizar máquinas de Turing como elementos constructivos de otras máquinas de Turing, lo que ayuda mucho en el diseño de máquinas de Turing complejas. También es capaz de trabajar con gramáticas y expresiones regulares. Su interfaz gráfica es fácil e intuitiva y está bastante lograda, aunque puede ser mejorada considerablemente (sólo se pueden usar líneas rectas para las transiciones, necesitamos cambiar de herramienta de dibujo para completar el diseño del autómata, no es posible mover

grupos de estados). Al ser demasiado extenso no profundiza en todos los aspectos, sobre todo en gramáticas y expresiones regulares, y hay algoritmos que no implementa. No implementa el *algoritmo de Cocke-Younger-Kasami* (CYK) [5,6] ni el *método de las derivadas* [7].

- *Simulations in Automata Theory* (http://www.cs.iitm.ernet.in/tell/automata/Automata/): Es un conjunto de applets para la simulación de distintos dispositivos. Dispone de un simulador de autómatas finitos, de autómata de pila, máquina de Turing y el algoritmo CYK.
- *Anagra* (http://webdiis.unizar.es/~ezpeleta/COMPI/compiladoresI.htm): Mientras que las anteriores herramientas estaban orientadas en su mayor parte a la simulación de autómatas, esta aplicación está orientada a la simulación y análisis de gramáticas (independientes del contexto y regulares). Es una herramienta que permite el análisis sintáctico asociado a gramáticas. Es capaz de realizar tanto el análisis ascendente como el descendente. También permite transformar gramáticas de un tipo a otro según la jerarquía establecida por Chomsky. Anagra es una herramienta muy completa, implementa gran cantidad de transformaciones que podemos aplicar sobre nuestras gramáticas. Sin embargo, desde nuestro punto de vista, esta limitada por la escasez de información, ya que da por hecho que el usuario conoce toda la información y no muestra los pasos intermedios de las transformaciones realizadas. Podría resultar muy útil para comprobar si un ejercicio se ha resuelto correctamente, pero no para aprender su funcionamiento.

De todas estas herramientas las más completas son las dos últimas, y las que estamos tomando de inspiración en el desarrollo de Thoth. El punto fuerte de JFLAP es que cubre una amplia variedad de dispositivos de reconocimiento. Anagra la complementa con su amplia variedad de algoritmos de análisis para gramáticas independientes del contexto. Thoth aspira a integrar las funcionalidades de ambas, corrigiendo algunas de sus carencias y mejorando su interfaz gráfica.

En su versión actual la funcionalidad de nuestra aplicación es perfectamente competitiva con JFLAP en le referente a las operaciones relativas a autómatas, expresiones regulares y a gramáticas. La interfaz gráfica es mucho más conveniente y además incorpora funcionalidad adicional para la conversión entre expresiones regulares y autómatas finitos y el análisis de cadenas.

La internacionalización, Thoth permite elegir entre cuatro lenguajes: español, inglés, francés y alemán, es otro punto a favor de nuestra aplicación ya que la hace mucho más atractiva para usuarios de habla no inglesa, como por ejemplo en Latinoamérica o Europa.

4. Algoritmos implementados en Thoth

Uno de los aspectos de diseño de THOTH ha sido la facilidad de uso. THOTH presenta en su interfaz de usuario un montón de funcionalidades que no están presentes en otras herramientas similares. Es posible agrupar y mover partes del autómata para facilitar que usuario minimice los cruces. También permite distribuir los estados de un autómata de forma automática en el área de dibujo (en esta primera versión simplemente los distribuye en torno a un círculo). Muchas de las operaciones más habituales no necesitan cambiar de herramienta de dibujo. Los arcos no se limitan a ser líneas rectas como en otras aplicaciones, con lo que es más fácil evitar los cruces de arcos.

4.1. Trabajando con gramáticas

Thoth permite introducir una gramática utilizando la misma sintaxis que la utilizada por bison [8]. Una vez introducida la gramática, determina el tipo de la misma y los alfabetos de terminales y no terminales (Figura 1, izquierda), entonces podemos llevar a cabo varias operaciones sobre la gramática (Figura 1, derecha).

Existen tres tipos de operaciones que podemos hacer con una gramática. Transformarla en otra que represente el mismo lenguaje, pero en la que sus producciones sigan un tipo particular de estructura. Utilizarla para determinar si una cadena particular pertenece al lenguaje que representa. Si la gramática es regular, obtener el autómata finito que reconoce el lenguaje que la gramática representa. En THOTH las posibles transformaciones tienen que ver con la

eliminación de: *producciones unitarias* (producciones que únicamente transforman un no terminal en otro no terminal), *producciones epsilón* (producciones que generan la palabra vacía, hacen que un no terminal desaparezca de la forma sentencial), *símbolos no alcanzables* (símbolos que no van a aparecer en ninguna forma sentencia que se pueda derivar desde el axioma de la gramática), *símbolos no terminables* (símbolos que nunca podrán transformarse en una cadena de terminales), *recursión directa* e *indirecta* por la izquierda (las gramáticas con este tipo de recursión no son adecuadas para el análisis descendente). En cuanto al análisis de cadenas, en THOTH es posible utilizar un algoritmo general, el *algoritmo de Cocke-Younger-Kasami* (Figure 2, izquierda) y un análisis de tipo descendente (la parte derecha de la Figura 2 muestra la tabla de análisis sintáctico predictivo utilizada en este tipo de análisis).

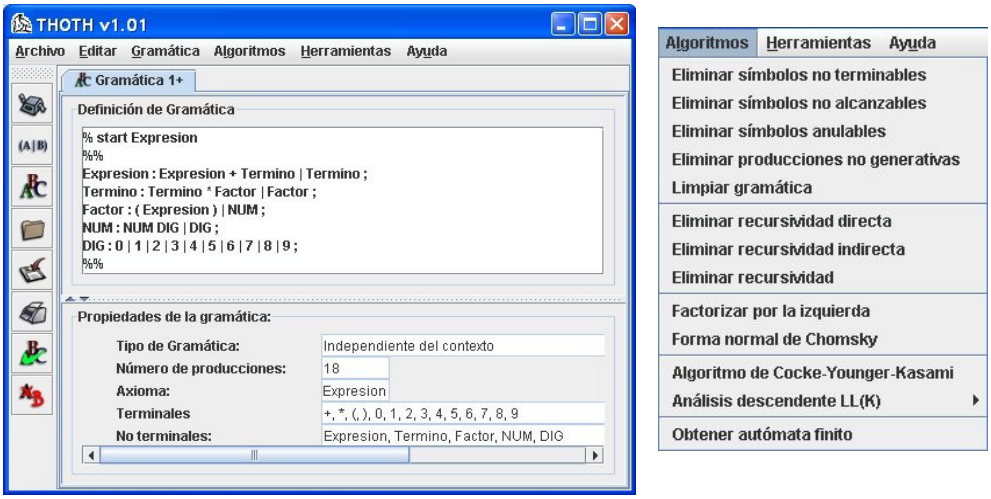

Figura 1. Izquierda: Entrada de gramática. Derecha: Algoritmos sobre gramáticas.

4.2. Trabajando con expresiones regulares

Cuando tratamos con expresiones regulares podemos tener en mente tres tipos de operaciones: obtener el autómata finito que reconoce el lenguaje representado por la expresión regular, determinar si una palabra en particular pertenece a dicho lenguaje, o determinar si dos expresiones regulares representan el mismo lenguaje. Estas dos últimas operaciones no las podemos llevar a cabo directamente en Thoth, aunque si de forma indirecta obteniendo primero el autómata finito que reconoce el lenguaje representado por la expresión regular, y entonces comprobando si la una cadena pertenece al lenguaje o si el autómata obtenido es equivalente al obtenido de otra expresión regular.

Thoth implementa tres algoritmos distintos para obtener el autómata finito equivalente a una expresión regular: el método de las derivadas [7], el método que Aho, Sethi y Ullman explican su su conocido "libro del dragón" [9] (algoritmo 3.5, que presenta una conexión muy estrecha con el método descubierto simultáneamente por Glushkov [10] y por McNaughton y Yamada [11]), y el método de Thompson's [12]. En la Figura 3 se puede ver el resultado de aplicar los primeros métodos a la expresión regular $a^*b|c^*$. Ninguno de estos dos métodos se ha encontrado implementado en las herramientas analizadas en la sección 3.

4.3. Trabajando con autómatas

Cuando se usa Thoth para trabajar con un autómata finito podemos o bien obtener primero un autómata a partir de una expresión regular, utilizando cualquiera de los métodos comentados en la sección previa, o bien dibujar directamente el diagrama de transiciones del autómata.

Dibujar un diagrama de transiciones con Thoth es un método muy directo. Se dispone de una herramienta para dibujar estados y otra para conectarlos mediante transiciones (en realidad el proceso se ve acelerado dado que no necesitamos cambiar la herramienta de dibujo en absoluto, es posible simplemente hacer doble click sobre el estado origen y arrastrar al estado de destino). Podemos mover estados individuales o agruparlos y moverlos todos juntos, de esta forma podemos minimizar el cruce de transiciones y presentar un diseño claro del autómata. Es posible incluso hacer que Thoth reubique los estados por nosotros. Todas estas características dan a Thoth una clara ventaja frente a otras herramientas para trabajar con autómatas finitos.

Una vez disponemos de un autómata podemos comprobar si reconoce una determinada palabra, podemos eliminar el no determinismo o minimizar el número de estados cuando esto es posible. Podemos también comprobar su equivalencia con otro autómata. Finalmente, podemos obtener la gramática o la expresión regular que representan el mismo lenguaje que el autómata reconoce. Thoth obtiene la expresión regular equivalente a un autómata usando el método de las ecuaciones características (también conocido como el método algebraico de Brzozowski [7]). Este es un método que no se ha encontrado implementado en otras herramientas. En la Figura 4 podemos ver el diagrama de transiciones de un autómata y el menú de algoritmos disponibles para trabajar con autómatas.

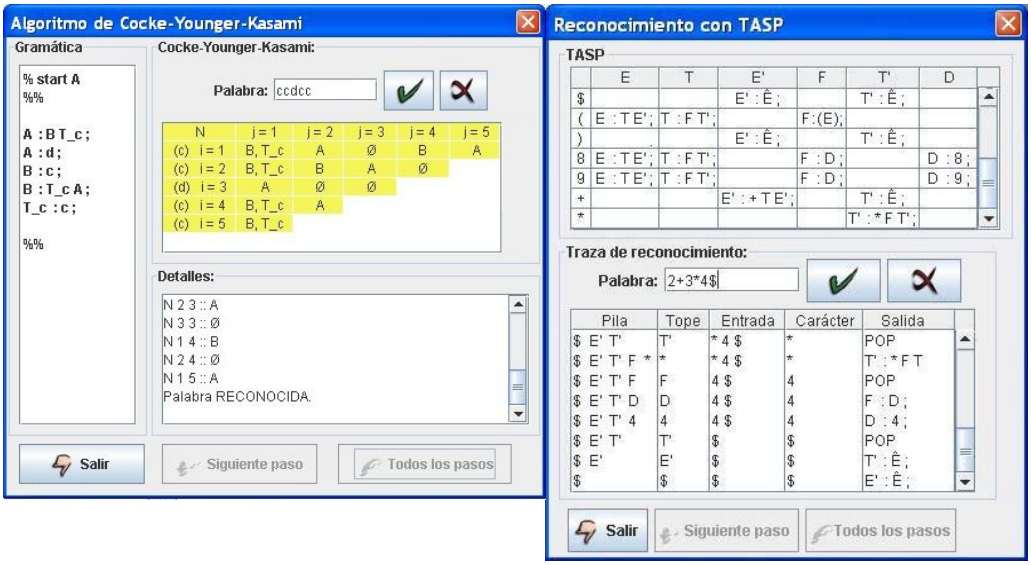

Figura 2. Izquierda: Algoritmo de Cocke-Younger-Kasami. Derecha: Tabla de análisis sintáctico predictivo.

5. Conclusiones y trabajo futuro

Hemos presentado una herramienta que pretende facilitar al alumno la interacción con algoritmos que siempre es más visual y menos pesada que las trazas con lápiz y papel, permitiendo la experimentación y observando la evolución paso a paso de los algoritmos.

La principal ventaja de Thoth frente a otras herramientas es su interfaz de usuario, que permite el fácil y rápido diseño de autómatas dibujando estados y conectándolos con transiciones. La evolución que presenta de los algoritmos esta orientada al aprendizaje de los mismos y no se limita a ser una simple herramienta de corrección de ejercicios. Además proporciona de forma integrada algoritmos no presentes en otras herramientas.

Sin embargo Thoth es una herramienta joven y con numerosas mejoras aún pendientes. Actualmente estamos trabajando en la

incorporación de los autómatas de pila y las máquinas de Turing. Además planeamos añadir los algoritmos de análisis ascendente.

También hemos identificado mejoras en su interfaz, en su versión actual no hay posibilidad de deshacer acciones, esperamos que en futuras versiones este sea otro de los puntos fuertes de su interfaz. Por último esperamos refinar la granularidad con la que se muestran los pasos de ejecución de los algoritmos para proporcionar al alumno la mayor información posible sobre los mismos.

Lamentablemente todavía no se dispone de una evaluación formal por parte de los alumnos, la asignatura de "Autómatas y Lenguajes Formales" en Burgos es una asignatura del primer cuatrimestre y la versión estable de la aplicación no estuvo disponible a tiempo como para poder haber sido utilizada intensivamente. Este es otro trabajo pendiente, junto con el de las mejoras previamente comentadas. La versión comentada en este artículo se encuentra disponible para su descarga en la dirección: http://pisuerga.inf.ubu.es/cgosorio/THOTH/ y cualquier comentario sobre la misma será bienvenido.

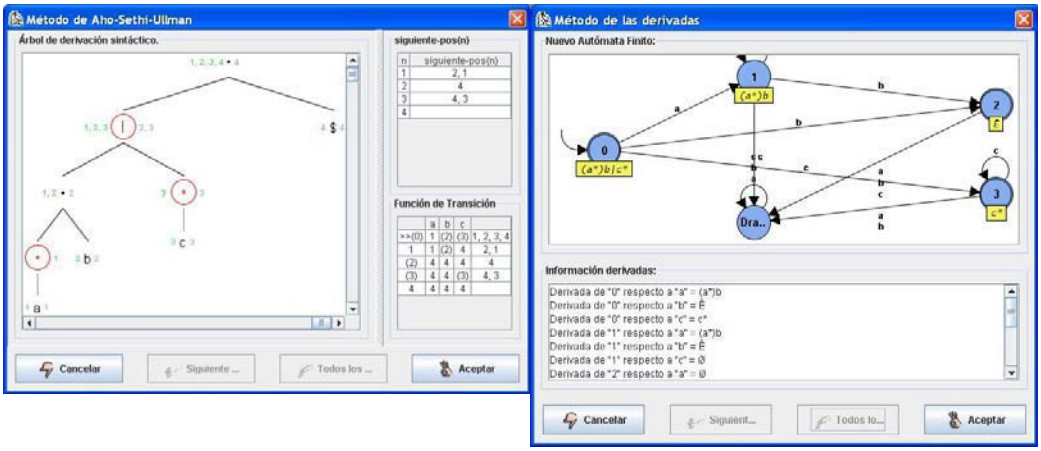

Figura 3. Izquierda: Método de Aho-Sethi-Ullman. Derecha: Método de las derivadas.

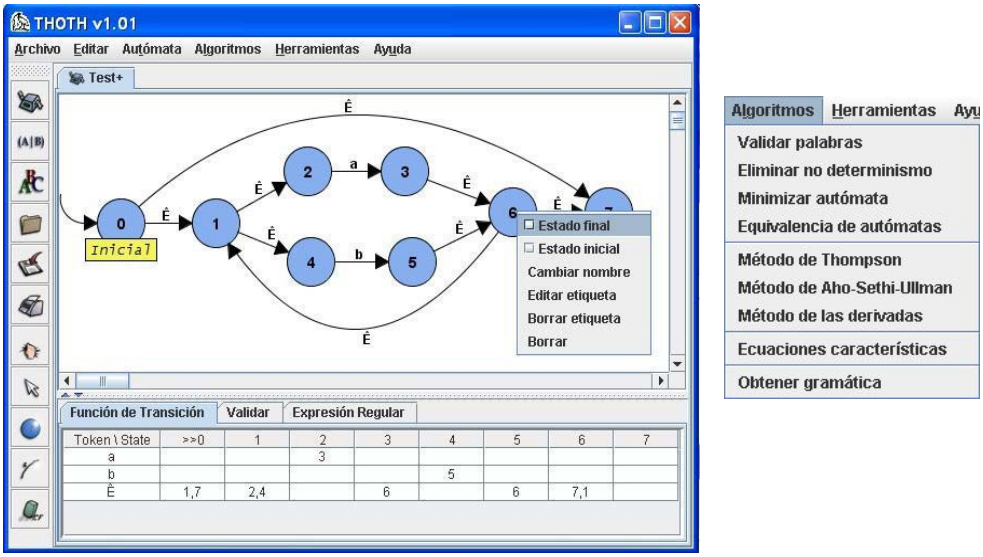

Figura 4. Izquierda: Entrada de autómata. Derecha: Algoritmos sobre autómatas.

Agradecimientos

La realización de este trabajo ha sido posible gracias al proyecto BU004B06 de la Consejería de Educación de la Junta de Castilla y León.

Referencias

[1] Dean Kelley. *Automata and Formal Languages: An Introduction.* Prentice Hall, 1998.

[2] Dean Kelley. *Automata and Formal Languages: An Introduction.* Prentice Hall, 1998.

[3] J. Glenn Brookshear. *Theory of Computation: Formal Languages, Automata, and Complexity.* Benjamin-Cummings Publishing Company, 1989.

[4] Manuel Alfonseca, Enrique Alfonseca y Roberto Moriyón. *Teoría de Autómatas y Lenguajes Formales*, McGraw Hill, 2007.

[5] Susan H. Rodger and Thomas W. Finley. *JFLAP: An Interactive Formal Languages and Automata Package.* Jones & Bartlett Publishers, 2006.

[6] D. H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189-208, 1967.

[7] T. Kasami. An eficient recognition and syntax algorithm for context-free languages. *Scientific Report* AFCRL-65-758, Air Force Cambridge Research Lab., Bedford, Massachussetts, 1965.

[8] Janusz A. Brzozowski. Derivatives of Regular Expressions, in *Journal of the ACM*, 11(4): 481-494, 1964.

[9] Charles Donnelly and Richard Stallman. *Bison, The yacc-compatible parser generator.* Free Software Foundation, 2006.

[10] Alfred V. Aho, Ravi Sethi y Jeffrey D. Ullman. *Compiladores: Principios, técnicas y herramientas*. Addison Wesley, 1990.

[11] V. M. Glushkov. "The abstract theory of automata", Russian Mathematical Surveys, 16:1-53, 1961.

[12] R. McNaughton y H. Yamada. "Regular expressions and state graphs for automata", IEEE *Transactions on electronic computers* 9(1):39-47, 1960.

[13] K. Thompson. Regular expression search algorithms, *Communication of ACM* 11(6): 419-422, 1968.