Una Herramienta para la Simulación de Autómatas Traductores en la Enseñanza de Teoría de la Computación

Agustín Esmoris **Carlos Iván Chesñevar**

Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur – Av. Alem 1253
B8000CPB Bahía Blanca – Argentina
Tel. 0291-459-5135 / Fax 0291-459-5136
Email: cic@cs.uns.edu.ar, agustinesmoris@softhome.net

Palabras clave: Informática Educativa, teoría de lenguajes formales y autómatas, autómatas traductores
Remitido a: II Workshop de Tecnología Informática Aplicada en Educación - IX CACIC

Resumen

Este trabajo presenta las características de **TAGS**, un simulador gráfico para autómatas traductores que puede usarse como herramienta, para enseñar distintos aspectos de la teoría de estos autómatas en un curso tradicional de teoría de la computación. **TAGS** fue desarrollado para simular autómatas traductores, permitiendo que el estudiante integre los principales conceptos teóricos y prácticos asociados a ellos. A través de **TAGS** el usuario puede definir un autómata traductor arbitrario de una forma interactiva, siendo capaz de simularlo y trazar su comportamiento bajo distintas *vistas*. Las vistas involucran la descripción formal, gráfica y en lenguaje natural del proceso de simulación, el cual puede enriquecerse opcionalmente a través de un software para reproducción de voz. **TAGS** también permite resolver problemas típicos asociados a autómatas traductores, tales como la minimización de un autómata dado y la transformación de autómatas de Moore en autómatas de Mealy (y viceversa). Estas operaciones pueden ser desarrolladas paso a paso, facilitando al estudiante el seguimiento y comprensión de los algoritmos subyacentes involucrados.

1 Introducción y Motivaciones

Los lenguajes formales y la teoría de autómatas son temas centrales en la currícula de Ciencias de Computación. A menudo estos temas presentan la complejidad de ser difíciles tanto de enseñar como de aprender, en virtud de que el profesor enfrenta el desafío de presentar clases motivadoras vinculadas a temas abstractos y complejos. Por otro lado, muchas veces los estudiantes tienen problemas para entender estos nuevos conceptos, o incluso las ideas intuitivas subyacentes, ya que deben enfrentar una nueva notación y lenguaje técnico al cual deben acostumbrarse. Esta situación ha motivado el desarrollo de diferentes programas denominados "*simuladores de computadora*" (en inglés *computer simulators)* como herramientas educacionales que permiten que el estudiante implemente muchos de estos tópicos, los que tradicionalmente se estudian bajo una perspectiva matemática antes que computacional o algorítmica.

Este trabajo presenta a **TAGS**, una herramienta de software educativo desarrollada para ayudar a los estudiantes de un curso de teoría de la computación a diseñar y probar una clase particular de

autómatas de estado finito (AEF) denominados *autómatas traductores* (en inglés *transducer automata)* [9]. **TAGS** incluye ciertas características que lo hacen atractivo como herramienta pedagógica en un curso de Lenguajes Formales y Teoría de Autómatas (LFTA). Algunas de estas características incluyen la capacidad de diseñar y simular un autómata traductor de Mealy o de Moore arbitrario, y desarrollar la conversión de un autómata de Mealy en un autómata de Moore equivalente (o viceversa). A través de **TAGS** pueden minimizarse tanto autómatas de Moore como de Mealy utilizando un algoritmo que incluye efectos de animación junto con una explicación paso a paso. Una característica particularmente motivadora de **TAGS** es la capacidad de detectar un software de procesamiento de voz en el sistema operativo donde se está corriendo el programa. De estar disponible este software, **TAGS** puede "contarle" al estudiante cómo procede el autómata a medida que procesa una determinada cadena de entrada, o bien cómo lleva a cabo los distintos pasos intermedios del algoritmo de minimización a través del cómputo de clases de estados equivalentes.

Este trabajo se estructura como sigue: la sección 2 presenta los aspectos básicos centrales correspondientes a la teoría de autómatas traductores. Se mencionan asimismo algunas propiedades y teoremas relevantes que son modelados por **TAGS**. En la sección 3 se desarrollan las principales características de **TAGS** y se presenta una breve comparación con trabajos relacionados. Finalmente la sección 4 presenta las conclusiones obtenidas.

2 Autómatas Traductores: fundamentos teóricos y propiedades

Seguidamente sintetizaremos algunos conceptos básicos de la teoría de autómatas traductores, así como algunas propiedades especiales y resultados de equivalencia.

Un autómata traductor (*determinístico)* de estado finito *(ATEF)* es un dispositivo similar a un AEF determinístico, excepto que su propósito no es aceptar cadenas, sino transformar cadenas de entrada en cadenas de salida. Informalmente, un ATEF T comienza en un estado inicial designado s_0 y se mueve de estado en estado, dependiendo de su entrada, de manera análoga a lo que hace un autómata determinístico. En cada paso, T emite (o escribe en una cinta de salida) una cadena de cero o más símbolos, dependiendo del estado en curso s_i y del símbolo de entrada recibido. El diagrama de estados para un ATEF determinístico se parece al de un AEF determinístico, excepto que o bien cada **estado** o bien cada **arco** están etiquetados con un símbolo de salida *s*, produciéndose *s* como salida cada vez que se alcanza el estado en cuestión (o se recorre el arco asociado). En el primer caso, el autómata es denominado **autómata de Moore** [12]; en el segundo caso, el autómata es llamado **autómata de Mealy** [11]. Formalmente:

Definición 1: Un autómata traductor *(determinístico)* de estado finito *(ATEF)* es una 6-tupla $T = (S, \Sigma, O, \delta, s_0, f_0)$ donde S es un conjunto finito de estados, $S \neq \varnothing$, Σ es el alfabeto de entrada, O es el alfabeto de salida, $\delta : S \times \Sigma \rightarrow S$ es la función de próximo estado o de transición, s_0 es el estado inicial del autómata, $s_0 \in S$ y f_0 es una función de salida definida como $f_0 : S \rightarrow O$ (si T es un autómata de Moore) o $f_0 : S \times \Sigma \rightarrow O$ (si T es un autómata de Mealy). Diremos que un ATEF (*determinístico*) $T = (S, \Sigma, O, \delta, s_0, f_0)$ es un *autómata traductor suficientemente especificado* siempre que la función de transición $\delta : S \times \Sigma \rightarrow S$ sea una *función total.*

La figura 1 ilustra informalmente a los dos tipos de autómata traductor presentados. A la izquierda se muestra un autómata de Moore T. Nótese que dada la secuencia de entrada "`aabba`", T produce como salida la cadena "`11010`". Debe notarse que no se considera la salida asociada al estado

inicial s_0. A la derecha se muestra un autómata de Mealy T' que se comporta de la misma manera que T. Nótese que en el primer caso los estados fueron etiquetados utilizando símbolos de salida, mientras que en el segundo caso los símbolos de salida han sido asociados a los arcos que vinculan los nodos del autómata.

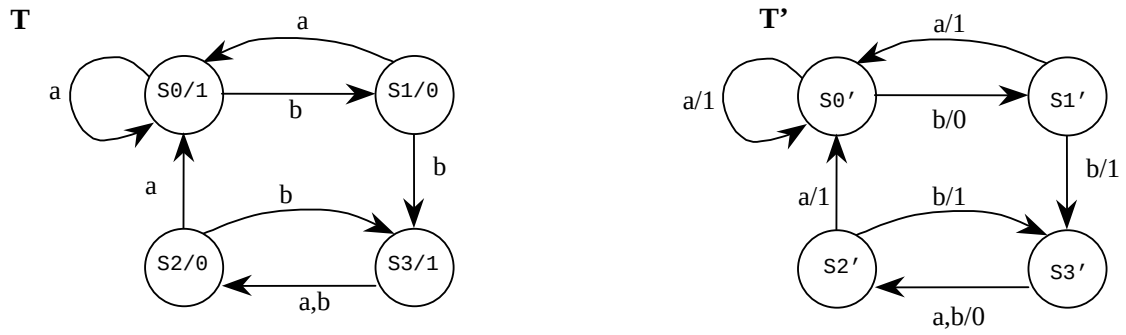

Figura 1: autómatas de Moore y Mealy

Los estados de cualquier autómata T pueden ser *clasificados* de acuerdo a cómo se comportan con respecto a las diferentes cadenas de entrada. Así, por ejemplo, un estado s_k es llamado *estado transitorio* si no existe ninguna secuencia de símbolos α, tal que $\delta(s_k, \alpha) = s_k$. Análogamente, un estado s_j, $j \neq 0$, es llamado *estado inalcanzable* si no existe una secuencia de entrada α, tal que $\delta(s_0, \alpha) = s_j$. Considérese por ejemplo a los autómatas T y T' definidos en la figura 1. Claramente ningún estado es inalcanzable y los estados s_0 y s_0' no son estados transitorios.

Dado un autómata traductor arbitrario $T = (S, \Sigma, O, \delta, s_0, f_0)$ y una cadena de entrada $\alpha = c_1c_2c_3...c_j$, $1 \leq k \leq j$, $c_k \in \Sigma$, se define la *traducción de α via T* como la salida producida por T ante dicha entrada (i.e. $T(\alpha)$ es la salida producida por T ante la secuencia de entrada α). Formalmente resulta $T(\alpha) = T(c_1c_2c_3...c_j) = a_1a_2a_3...a_j$, $1 \leq i \leq j$, $a_i \in (O \cup \{\lambda\})$, donde λ representa a la cadena nula. En el caso del autómata de Moore de la figura 1, resultará por caso T(aabba)=11010.

Minimización de autómatas traductores

Si F es un autómata de estado finito, un resultado notorio en teoría de autómatas se refiere al cálculo de la versión *minimizada* de dicho autómata. Es decir, hallar un autómata F' con estados S' tal que F y F' se comporten del mismo modo, pero F' posea un cantidad *minimal* de estados. Un resultado similar se extiende a los autómatas traductores.

Definición 2: Dado un autómata traductor $T = (S, \Sigma, O, \delta, s_0, f_0)$, se define $L(T) = \{ w \mid T(\alpha) = w, \forall \alpha \in \Sigma^* \}$. Nótese que L(T) es *el lenguaje de salida para T* (esto es el conjunto de todas las cadenas de salida que pueden ser generadas por T para toda posible cadena de entrada α).

Definición 3: Un autómata traductor $T' = (S', \Sigma, O, \delta', s_0', f_0')$ es la *versión minimizada* de T si $L(T') = L(T)$, $T(\alpha) = T'(\alpha)$, $\forall \alpha \in \Sigma^*$ y además T' posee un número minimal de estados.

El algoritmo de minimización para hallar T' a partir de T se basa en el cálculo de clases de equivalencia sobre el conjunto de estados original S. Véase [3,7,12,14] para más detalles.

Teoremas de equivalencia

Nótese que las cadenas de salida obtenidas tras procesar *cualquier* cadena de entrada por medio de un autómata traductor de Mealy T y un autómata traductor de Moore T' nunca pueden ser equivalentes, dado que el largo de $T(\alpha)$ será siempre uno menos que el largo de $T'(\alpha)$ para cada α posible. No obstante, podemos descartar la respuesta inicial de un autómata de Moore y decir que un autómata de Mealy T y un autómata de Moore T' son equivalentes si para todas las entradas α, $xT(\alpha) = T'(\alpha)$, siendo x la salida de T' para su estado inicial. De acuerdo a esta convención podemos enunciar los siguientes teoremas [9]:

Teorema 1: Si $T_1 = (S, \Sigma, O, \delta, s_0, f_0)$ es un autómata traductor de Moore, entonces existe un autómata traductor de Mealy T_2 equivalente a T_1 .

Teorema 2: Si $T_1 = (S, \Sigma, O, \delta, s_0, f_0)$ es un autómata traductor de Mealy, entonces existe un autómata traductor de Moore T_2 equivalente a T_1 .

3 La herramienta TAGS: características

3.1 Interacción con TAGS: diseño de un autómata traductor

TAGS es una aplicación independiente que puede ser fácilmente instalada y ejecutada sobre cualquier PC basada en el sistema operativo Windows. Al iniciarse **TAGS**, se pone a disposición del usuario una *ventana de diseño* sobre la cual puede dibujarse el grafo correspondiente a un autómata traductor (de Moore o Mealy). A la derecha de esta ventana aparece una *barra de herramientas de diseño* (ver figura 2), ofreciendo íconos para acceder directamente a diferentes opciones del programa (sus funcionalidades se listan a continuación):

Figura 2

- Guardar un autómata en disco
- Cargar un autómata previamente guardado
- Agregar un nuevo estado al autómata actual
- Activar o desactivar el uso del sistema de voz (*speech-engine*)
- Definir los alfabetos de entrada y salida
- Tomar una "fotografía" del autómata actual en cualquier etapa de ejecución y guardarla en disco usando el formato JPG

Adicionalmente, **TAGS** también provee un conjunto de prácticos atajos de teclado. En consecuencia, resulta sumamente sencillo diseñar el grafo correspondiente a un autómata traductor arbitrario (de Moore o Mealy). El usuario puede por ejemplo agregar un nuevo arco entre dos estados simplemente haciendo doble click sobre el estado origen y luego otro click sobre el estado destino. Debe notarse también que los arcos pueden ser reformados dinámicamente a través del ratón. Las etiquetas de arcos y estados pueden ser fácilmente arrastradas a nuevas posiciones del

mismo modo. Asimismo, el usuario puede cambiar la apariencia del autómata (color de los estados y arcos, ancho de los arcos, tamaño de los círculos correspondientes a estados, etc.) El usuario puede también eliminar un arco o estado simplemente seleccionándolo con el ratón y presionando luego *Delete* en el teclado.

3.2 Ejecución de un autómata

Una vez que el autómata traductor ha sido completamente diseñado, el usuario puede comenzar el proceso de simulación de la ejecución del autómata para una cadena de entrada arbitraria (dada según el alfabeto Σ originalmente definido). **TAGS** provee al usuario de una *barra de herramientas*

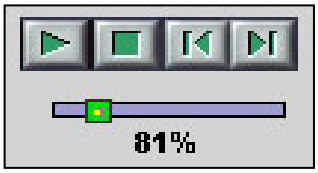

Figura 3: barra de ejecución y barra deslizante de velocidad

de ejecución ubicada debajo de la barra de herramientas de diseño. La barra de herramientas de ejecución ofrece cuatro botones. Presionando el primer botón (Play) **TAGS** comienza a simular el comportamiento del autómata para la cadena de entrada especificada y continúa hasta que todos los símbolos de entrada han sido procesados. Una vez que el Botón Play ha sido presionado, **TAGS** entra en *modo simulación* (esto es, el autómata no puede editarse mientras este funcionando). El segundo botón simplemente detiene el proceso de animación, haciendo que **TAGS** retorne al *modo de diseño* original. El tercer y cuarto botón (retrasar y adelantar animación) permiten animar la ejecución en un estilo paso por paso. De este modo, el usuario puede rebobinar o adelantar la ejecución actual tantos pasos como él desee. El usuario puede también establecer la velocidad de animación deseada por medio de una *barra deslizante (slide bar)*, proporcionada junto a la barra de herramientas de ejecución (ver figura 3).

3.3 Capacidad de diferentes vistas para el mismo autómata

Como se sugiere en [10], utilizar simuladores educativos para complementar los diferentes tópicos de un curso de LFTA posee un fuerte impacto pedagógico a partir de la noción de *vistas*. Para el caso particular de autómatas a pila (AP), Jennifer McDonald propone cuatro vistas diferentes: (1) ***vista de la cinta*** (mostrando la cinta de entrada actual); (2) ***vista de la pila*** (mostrando la pila actual); (3) ***vista de la traza*** (mostrando una traza de los pasos requeridos para procesar una cadena);y (4) ***vista del autómata*** (visualización del AP como grafo). En **TAGS** estas vistas fueron adaptadas parcialmente a los autómatas traductores y adicionalmente extendidas de la siguiente manera:

1) **Vista del diagrama de transiciones:** cuando se trabaja en el modo de diseño, **TAGS** provee una representación estilo grafo del autómata. En esta vista, el estudiante puede diseñar su propio autómata casi tan naturalmente como si lo estuviese dibujando sobre un pizarrón. Nótese que el autómata debe ser un *autómata suficientemente especificado*. Cuando se simula

un autómata, se destacan siempre tanto el estado actual como el arco atravesado al procesar el símbolo en curso.

2) **Vista de entrada / salida:** esta vista suministra una visión general tanto de la cadena que está siendo procesada como de la cadena de salida actualmente producida por el autómata. En esta vista, la cadena de entrada es mostrada dentro de un *campo de texto* donde el último símbolo leído es claramente distinguido. Esta vista es práctica para que el usuario pueda conocer la parte de la cadena de entrada que ya ha sido procesada.

3) **Vista del trayecto:** esta ventana desplazable (scrolling window) reporta un sumario de cada acción llevada a cabo durante el proceso de simulación. Dicho sumario involucra la descripción del estado actual, el símbolo leído, la salida obtenida y el nuevo estado alcanzado tras procesar el último símbolo.

4) **Vista de lenguaje natural:** esta vista provee una explicación completa de cada acción realizada por el autómata durante el procesamiento de una cadena de entrada. La explicación es presentada al estudiante en un lenguaje natural (como si lo que realiza el autómata estuviese siendo explicado por un tutor o instructor humano). La vista de lenguaje natural puede complementarse con un sistema de voz integrado (*speech-engine*).

5) **Vista Teórica:** esta vista involucra un diseño tabular en el cual se muestra la definición formal del autómata actual. En contraste a las tres vistas precedentes, ésta es estática, complementando a la vista desde el diagrama de transiciones por medio de la definición formal del autómata.

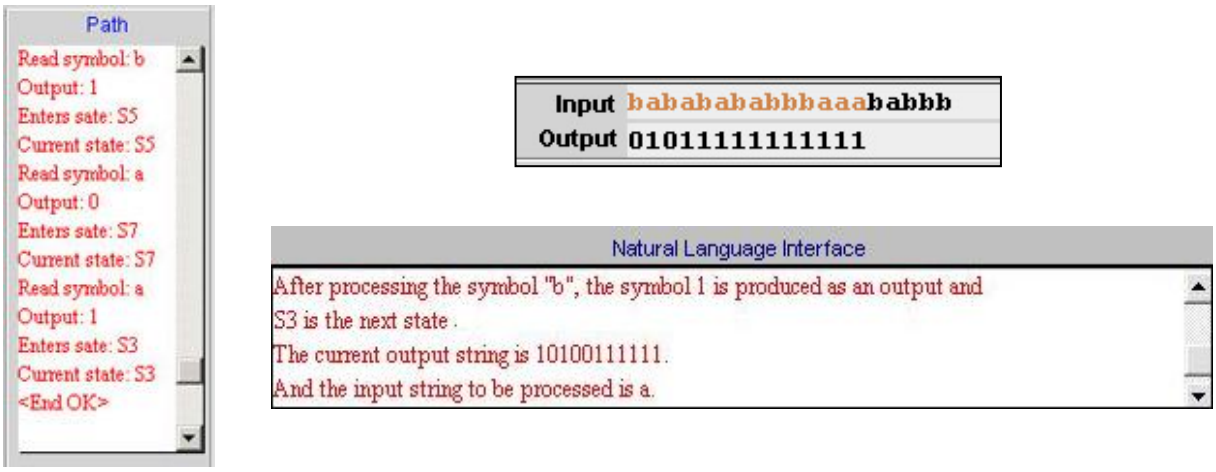

Figura 4: a) Vista del trayecto, b) Vista de entrada/salida, c) Vista de Lenguaje Natural

3.4 Uso de TAGS para resolver distintos aspectos de la currícula de LFTA

Como se planteo en la Sección 2, existen varios problemas prácticos y temas teóricos que pueden ser identificados en el contexto de los autómatas traductores. **TAGS** pretende proveer una herramienta de software que ayude al estudiante a afrontar estas situaciones, haciendo sencilla y más motivadora la tarea de resolver ejercicios típicos de teoría de autómatas traductores. Algunas opciones especiales incluidas en **TAGS** con la intención de conseguir este objetivo son las siguientes:

- **Minimización de Autómatas: TAGS** puede simular el notorio algoritmo de minimización sobre un autómata traductor arbitrario (ver figura 5). Esto puede aplicarse tanto a autómatas de Moore como de Mealy. Inicialmente, **TAGS** identifica todos los estados inalcanzables, eliminándolos del grafo. Posteriormente las clases *k*-equivalentes correspondientes son halladas (i.e. todos aquellos estados que se comporten de manera equivalente para cadenas cuya longitud sea menor o igual a *k*). Siempre que una clase es hallada, **TAGS** resalta todos los estados pertenecientes a dicha clase con un color que no ha sido utilizado para representar a alguna otra clase anteriormente. Esto, ayuda a que el estudiante reconozca fácilmente las diferentes clases dentro del autómata a lo largo del proceso de minimización. **TAGS** dará una explicación hablada del proceso de minimización en caso de tener un sistema de procesamiento de voz disponible en el sistema operativo.

- **Transformación de Autómatas Traductores:** dado un autómata traductor de Moore, **TAGS** permite al estudiante transformarlo en un autómata traductor de Mealy equivalente, aplicando el

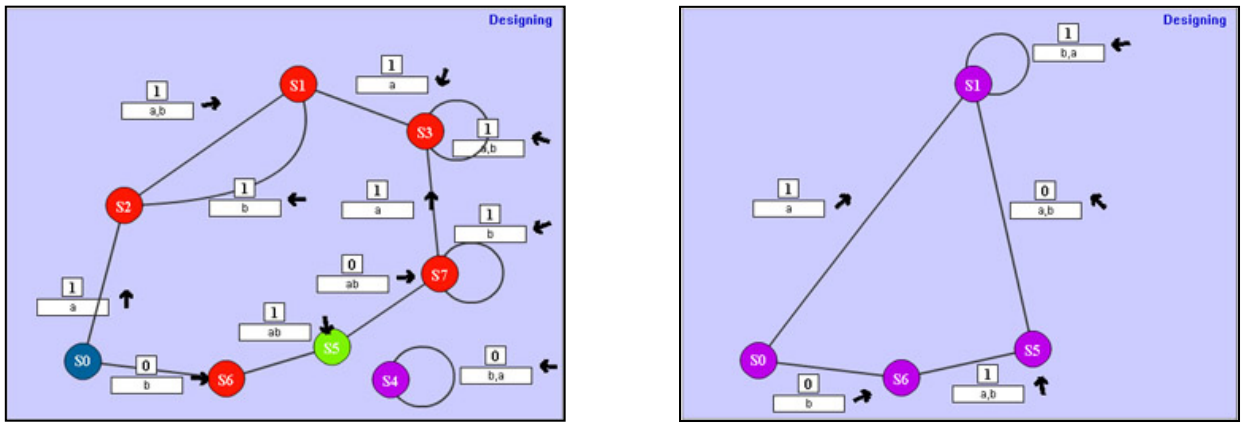

Figura 5: AT de Mealy antes y después de aplicarle el algoritmo de minimización

algoritmo correspondiente [3]. Inversamente, **TAGS** es capaz de transformar cualquier autómata traductor de Mealy T en un autómata traductor de Moore T' equivalente. Para este último caso, **TAGS** muestra una matriz con la definición completa del autómata T' en lugar de transformar directamente su grafo. Esto es debido al posible gran número de estados y transiciones que podrían generarse luego de la transformación de Mealy a Moore.

- **Análisis de las propiedades de los estados:** cuando se está trabajando sobre la ventana de diseño, después de mantener el puntero del ratón sobre un estado por 2 segundos, aparece una

ventana emergente (pop-up window). Esta ventana reporta diferentes hechos acerca del estado bajo el puntero del ratón (tales como la lista de estados alcanzables a partir de este estado, etc.).

- **Generador de HTML:** esta opción crea un documento HTML conteniendo la definición formal y una foto completa del autómata actualmente en la ventana de diseño.

- **Efectos sonoros asociados a reconocimiento de cadenas:** si lo desea, el usuario de **TAGS** puede asociar una nota musical a cada estado de un autómata de Moore o arco de un autómata de Mealy. Esto posibilita que el autómata interprete el procesamiento de una cadena dada como una pequeña "pieza musical", brindando un atractivo adicional que complementa la animación gráfica asociada a este proceso. Para ciertos patrones (como por ejemplo subcadenas que se repiten varias veces en una cadena dada) se generarán secuencias de notas repetidas, reforzando con un estímulo sonoro lo que puede apreciarse a nivel gráfico.

- **Actualizador (Updater): TAGS** posee la capacidad de realizar una actualización automática desde su sitio actual en Internet (http://cs.uns.edu.ar/~cic/fcc.htm).

- **Capacidad Multilenguaje: TAGS** incluye una opción que habilita al usuario a cambiar el idioma del entorno de trabajo. Actualmente **TAGS** puede ser ejecutado en inglés o español.

4 Conclusiones y Trabajos Relacionados

La experiencia reciente, ha mostrado que los simuladores por computadora proveen un vínculo interesante entre teoría y práctica. En este sentido, los entornos de simulación para LFTA refuerzan el significado de cuestiones teóricas cuando se resuelven ejercicios prácticos. En nuestra opinión el tratamiento detallado de los autómatas traductores ha sido dejado de lado en la mayoría de los entornos de software multipropósito para LFTA (tales como DEM [5] o JFLAP [8]) o bien no proveen un espectro amplio de posibilidades como el provisto por **TAGS**.

El software **TAGS** hizo posible analizar nuevos acercamientos para enseñar conceptos tradicionales (tales como el algoritmo de minimización). La inclusión de distintas *vistas* (tales como la vista en lenguaje natural y el tutor parlante) lo hicieron particularmente atractivo.

Entendemos que los autómatas traductores resultan un tema particularmente interesante para muchos estudiantes de Computación, y pensamos que la disponibilidad de un software como **TAGS** lo hace aún más atractivo. En particular, aquellos estudiantes cuyos estudios se orientan a Ingeniería en Sistemas tienden a estar interesados en este tipo de autómatas cuando se les comenta que los mismos se utilizan en aplicaciones de diseño lógico de circuitos y tópicos similares. En este sentido, cabe señalar que en el diseño de **TAGS** se integraron distintas consideraciones teóricas y propiedades formales de autómatas traductores que no se encuentran presentadas unificadamente en los libros de textos tradicionales en la materia (como [6] ú [9]). El software resultante sintetiza y compendia por ende varios aspectos teóricos y prácticos sobre autómatas traductores que no se encuentran disponibles en la bibliografía, presentando un recurso motivador para la enseñanza de este tema en un curso de LFTA. Como trabajo futuro se pretenden integrar a **TAGS** las definiciones teóricas y propiedades formales de estos autómatas, para que el usuario pueda visualizarlas si lo desea como una ayuda en línea.

Finalmente, es de remarcar que **TAGS** está disponible como software gratuito (freeware) en página web de la asignatura "*Fundamentos de Ciencias de la Computación*" de la Universidad Nacional del Sur (http://cs.uns.edu.ar/~cic/fcc.htm). Para ejecutarlo, sólo se requiere una configuración de PC típica y una versión actualizada del sistema operativo Windows (Windows 95 o superior).

Bibliografía

[1] Augusto, J.C. *Fundamentos de Ciencias de la Computación – Notas de Curso*. Universidad Nacional del Sur, Argentina, (1995).

[2] Bilska, A., Leider, K., Procopiuc, M., Procopiuc, O., Rodger, S., Salemme, J. R., and Tsang, E. *A collection of tools for making automata theory and formal languages come alive*. SIGCSE Bulletin (ACM Special Interest Group on Computer Science Education) *29* (1997).

[3] Booth, T. L. *Sequential Machines and Automata Theory*. John Wiley & Sons, Inc., New York, 1967.

[4] Chesñevar, C. I., Cobo, M. L., and Yurcik, W. *Using Theoretical Computer Simulators for Formal Languages and Automata Theory*. ACM SIGCSE Bulletin, Vol. 35, No. 2, June 2003, USA.

[5] Chesñevar, C. I, Cobo, M.L - *Simulators for Teaching Formal Languages and Automata Theory: A comparative Survey*. En Procs. del VIII Congreso Argentino en Ciencias de la Computación, págs. 1089-1097. Buenos Aires, Argentina, Octubre 2002.

[6] Cohen, D. I. A. *Introduction to Computer Theory*. John Wiley & Sons, Inc., New York, 1997.

[7] Goldin, D., and Keil, D. Minimal sequential interaction machines, 2000.

[8] Gramond, E., and Rodger, S. H. Using JFLAP to interact with Theorems in Automata Theory. Thirtieth SIGCSE Technical Symposium on Computer Science Education (1999), 336--340.

[9] Hopcroft, J. and Ullman, J.. Introduction to Automata Theory, Languages and Computation, Addison-Wesley, 1979.

[10] McDonald, J. Interactive Pushdown Automata Animation. *ACM SIGCSE Bulletin 34*, 1 (2002), 376-380.

[11] Mealy, G. H. A method for synthesizing sequential circuits. *Bell Systems Journal 34* (5), pp. 1045–1079.

[12] Moore, E. F. Gedanken experiments on sequential machines. In *C. E. Shannon and J. McCarthy, eds., Automata Studies (Princeton University Press),* pp. 129–153.

[13] Rasala, R. Toolkits in first year computer science: A pedagogical imperative. *SIGCSEB: SIGCSE Bulletin (ACM Special Interest Group on Computer Science Education) 32* (2000).

[14] Watson, B. W. A taxonomy of finite automata minimization algorithms, Computing Science Note 93/44, Eindhoven University of Technology, The Netherlands, 1993.