UNA PROPUESTA PARA LAS PRÁCTICAS DE LA MATERIA "TEORÍA DE AUTÓMATAS Y LENGUAJES FORMALES".

José A. Gámez, Juan A. Guerrero, José M. Puerta

Departamento de Informática
Universidad de Castilla-La Mancha
Escuela Politécnica Superior
Albacete, 02071
e-mail: {jgamez,guerrero,jpuerta}@info-ab.uclm.es

RESUMEN: En este trabajo presentamos una propuesta para la realización de las prácticas relativas a la materia "Teoría de Autómatas y Lenguajes Formales", basadas en el uso de distintos programas que permiten no sólo el diseño y prueba de los distintos modelos de autómata y gramática, sino también la interacción con el alumno para la resolución paso a paso de distintos problemas (minimización de autómatas finitos, conversiones entre distintos modelos de autómata y/o gramática, etc...).

1.- INTRODUCCIÓN.

En los planes de estudios de Ingeniería Informática e Ingeniería Técnica en Informática de Sistemas, la materia troncal "Teoría de Autómatas y Lenguajes Formales" (TALF en adelante) tiene asignados 9 créditos. En los últimos años, siguiendo las directrices relativas a la proporcionalidad entre créditos teóricos y prácticos, se ha pasado de impartir estos 9 créditos de forma teórica en su totalidad, a tener que impartir un tercio de ellos de forma práctica.

Si bien las recomendaciones realizadas por las asociaciones ACM e IEEE para la docencia en prácticas de TALF, consisten en el uso de simuladores que permitan al alumno afianzar los conocimientos adquiridos en las clases teóricas mediante el diseño y prueba en un computador de los distintos modelos de autómata y gramática estudiados, la carencia de estos simuladores, la restringida capacidad de los mismos, o simplemente el desconocimiento de su existencia, motivó (y continua motivando en algunos casos) que los créditos prácticos se emplearan en la resolución de problemas en pizarra y en la realización de programas que implementen algunos de los algoritmos vistos en clase.

Desde nuestro punto de vista y teniendo en cuenta que en nuestro plan de estudios TALF se imparte en segundo curso, coincidiendo en el tiempo con asignaturas tales como Metodología de la Programación, Algoritmica y Estructuras de Datos, consideramos que no es TALF el lugar adecuado para la realización de prácticas de programación. Por otra parte, la resolución de problemas en pizarra es imprescindible en algunos temas, aunque nuestra experiencia nos indica que en aquellos en los que es posible sustituirlos por prácticas de laboratorio, esta sustitución es ventajosa, ya que el alumno se motiva más y tiene una mayor participación respecto a las clases

de problemas en pizarra, en las que (la mayoría) suele limitarse a copiar la resolución realizada por el profesor o por otro compañero.

Una vez que hemos planteado nuestra opinión, nos fijamos dos objetivos para el resto del trabajo:

1. Realizar una propuesta de planificación temporizada de las prácticas para TALF. Para ello hemos considerado un temario clásico de la asignatura (ver sección 5) que puede ser cubierto, por ejemplo, usando los libros de texto de Hopcroft y Ullman, Isasi y col., y Lewis y Papadimitriou. La temporización se ha realizado considerando que la asignatura es cuatrimestral con una duración de 14 semanas (4 horas de teoría + 2 horas de prácticas).

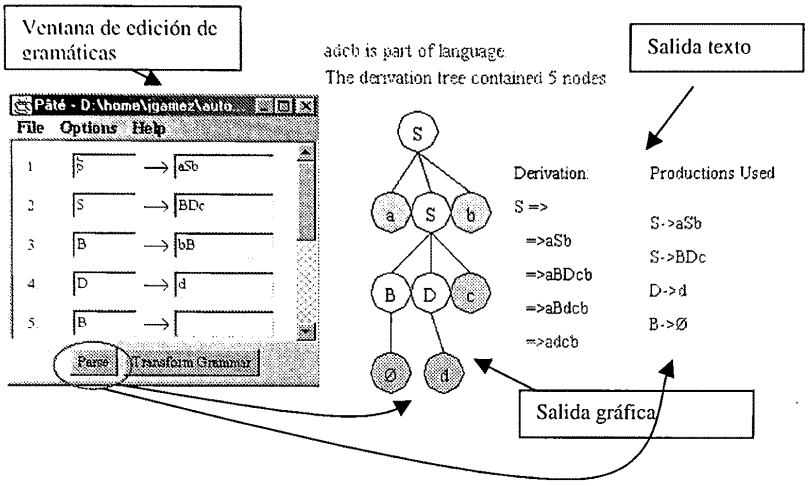
2. Asociar a cada una de las prácticas la herramienta con la cual llevarla a cabo. Estas herramientas han sido desarrolladas por el grupo de Susan Rodger en la Universidad de Duke (Rodger, 2000). Aunque estas herramientas pueden ser obtenidas en Internet, dada la grandeza (física) de la red, es difícil encontrarlas a menos que se tenga conocimiento de ellas. Nuestro objetivo, por tanto, es dar a conocer estas (a nuestro juicio) excelentes herramientas.

A continuación introducimos brevemente las herramientas utilizadas en las prácticas, proponemos una planificación temporizada (y sincronizada con la teoría) de las mismas, y finalmente presentamos algunas conclusiones.

2.- PÂTÉ.

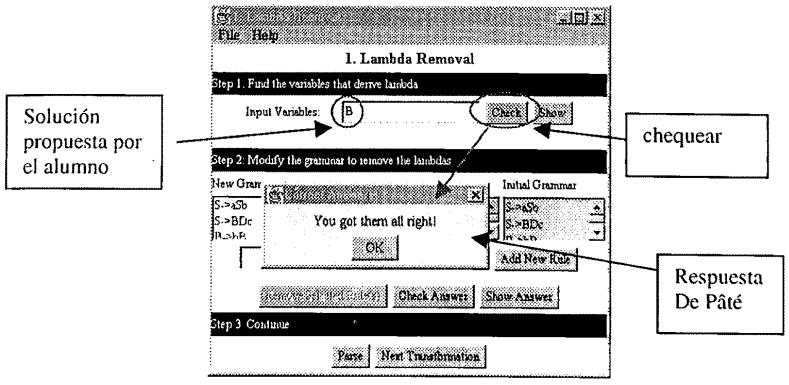
Pâté (**http://www.cs.duke.edu/~rodger/tools/pateweb**) es una herramienta visual e interactiva para el análisis y transformación de gramáticas desarrollada en la Universidad de Duke (Hung y Rodger, 2000). Pâté esta implementado en JAVA por lo que no necesita instalación y puede ejecutarse sobre cualquier plataforma en la que esté instalado el JDK 1.2.

La clase de gramáticas a analizar no se limita como ocurre en otras herramientas a las libres de contexto, si no que podemos escribir también gramáticas de tipo 1 o 0. En la siguiente figura podemos ver la ventana de edición de gramáticas y como al pulsar sobre el botón `parse` podemos proceder al análisis de una cadena. Inicialmente obtenemos un mensaje indicando si la cadena pertenece o no al lenguaje generado por la gramática. En caso positivo podemos ver (completamente o paso a paso) el árbol de análisis o la secuencia de derivaciones que genera la cadena.



En relación con la transformación de gramáticas, Pâté permite "limpiar" la gramática inicial mediante la eliminación de producciones nulas, producciones unitarias y producciones inútiles, así como la transformación de la gramática ya limpia a Forma Normal de Chomsky.

Evidentemente, introducir la gramática y pulsar un botón para obtener la gramática transformada no tiene mucho interés didáctico, a menos que el alumno compare el resultado con el obtenido por él. Sin embargo, Pâté permite la interacción por parte del alumno con el proceso de transformación, permitiendo al alumno introducir la solución parcial y chequear si su respuesta es válida. La siguiente figura muestra un ejemplo en el proceso de eliminación de producciones nulas, para lo que el alumno debe (1) identificar el conjunto de símbolos anulables, y (2) dar el conjunto de reglas de producción resultantes.



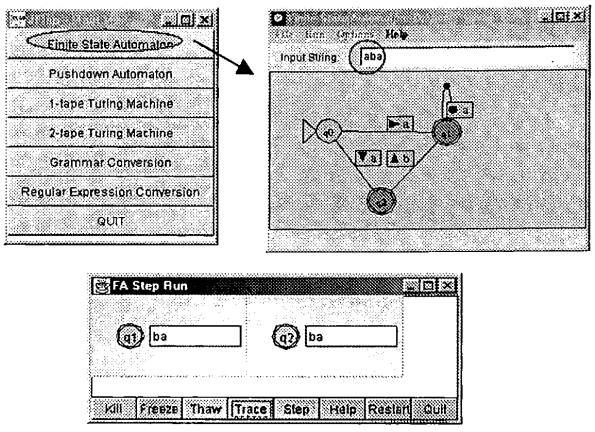
3.- JFLAP.

JFLAP (Java Formal Language & Automata Package) es un paquete de herramientas gráficas para asistir la docencia en TALF. Al igual que Pâté ha sido diseñado e implementado en JAVA

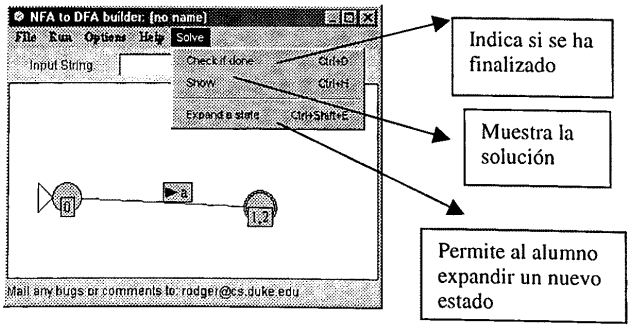
en la Universidad de Duke (Bilska y col., 1997). La última versión (1999) puede ser descargada de **http://www.cs.duke.edu/~rodger/tools/jflap.**

Las capacidades de JFLAP pueden dividirse en dos grupos:

1. Diseño y prueba (simulación) de diversos modelos de autómata: autómatas finitos (deterministas, no deterministas, y con transiciones nulas), autómatas con pila, máquinas de Turing (con una y dos cintas), expresiones regulares y gramáticas (regulares y libres del contexto). La siguiente figura muestra el menú principal de JFLAP, la ventana de edición de autómatas y un ejemplo tomado de la traza del proceso de reconocimiento de la cadena "aba" tras leer la primera "a".



2. Conversiones entre los distintos modelos. Así, podemos pasar de un tipo de autómata finito a otro, transformar un autómata en gramática regular o en expresión regular (y viceversa), minimizar autómatas finitos deterministas y, transformar autómatas con pila en G.L.C. (y viceversa). Al igual que ocurría en Pâté la ventaja (de cara a la docencia) es la posibilidad de que el alumno vaya construyendo su solución y verificándola con la ayuda de JFLAP. La siguiente figura forma parte de la transformación del AFND anterior en un AFD.



4.- MTURING.

MTURING es un programa desarrollado por Luis Jiménez Linares (Jiménez, 2000) en la Escuela Superior de Informática de la Universidad de Castilla-La Mancha. El programa dispone de una interfaz modo texto en la que se puede visualizar el estado y evolución de la cinta, así como la posición de la cabeza lectora/escritora, el estado actual y la transición que se está ejecutando en cada momento. MTURING permite definir (mediante la especificación de un fichero texto conteniendo sus transiciones) autómatas finitos y máquinas de Turing. En nuestro caso no usaremos MTURING con este fin, ya que JFLAP incluye estas capacidades, sino que usaremos MTURING exclusivamente para la última práctica de nuestra propuesta: *computación de funciones mediante esquemas de máquinas de Turing*. MTURING permite definir esquemas siguiendo la notación propuesta en (Lewis y Papadimitriou, 1998), es decir, permite definir un esquema como un AFD en el que los estados son máquinas de Turing (previamente definidas y etiquetadas).

Para hacer factible el desarrollo de la práctica, se facilitará al alumno un fichero con la definición de las máquinas básicas, es decir, máquinas de impresión de símbolos (P_*, P_1), máquinas de desplazamiento (R, L, R_*, L_*) y máquinas de copia y desplazamiento de cadenas (K, T), solicitándosele al alumno que implemente esquemas para la computación de funciones.

5. PLANIFICACIÓN DE LAS PRÁCTICAS.

Las prácticas de laboratorio se desarrollan de forma paralela a las de clase, pero con un desfase de al menos una semana entre la teoría explicada y la práctica correspondiente, con el objetivo de que el alumno tenga tiempo de asimilar la materia y de trabajar sobre las hojas de ejercicios que luego se usarán en las sesiones prácticas. En cada práctica se proponen dos tipos de tareas: comprobación y simulación de ejercicios resueltos previamente (diseño de autómatas, gramáticas, transformaciones, etc.), y resolución de ejercicios en los que, o bien se requiere gran cantidad de cálculos, o bien es interesante ver el proceso de resolución (minimización de AFD's, análisis y síntesis de lenguajes mediante el teorema de Kleene, etc.). En el siguiente cuadro se resume la temporización propuesta:

Semana	Teoría	Práctica
1	0. *Presentación* (1h) 1. *Introducción* (1h) 2. *Lenguajes Formales* (2h)	Preparación y organización de los grupos de prácticas
2	3. *Gramáticas Formales* (3h) Definición, conceptos, jerarquía de Chomsky.	
3	4. *Autómatas Finitos* (4h). AFD's, minimización de AFD's, AFND's.	Pr. 1: Construcción (y prueba) de gramáticas. [PÂTÉ]
4	4. *Autómatas Finitos* (3h). AFND-ε. Equivalencia AFD↔AFND↔AFND-ε. Equivalencia AF↔Gram. Regular. 5. *Otros modelos de AF* (1h) Máquinas secuenciales.	Pr. 2: Diseño y prueba de distintos AFDs. [JFLAP]
5	5. *Otros modelos de AF* (3h). Equivalencia Moore↔Mealy. Aut. Probabilísticos. 6. *Expresiones Regulares* (1h). Definición, ejemplos y propiedades.	Pr. 3: Minimización de AFDs (comprobación). Diseño de AFNDs. [JFLAP]
6	6. *Expresiones Regulares* (4h) Equivalencia: AF ↔ ER (Teor. De Kleene), AF→ER (ecuaciones), ER→AF (derivadas).	Pr. 4: Diseño de AFND-ε. Transformaciones AFD↔AFND↔AFND-ε (comprobación). [JFLAP].
7	7. *Propiedades de los Leng. Reg.* (4h). Lema de bombeo, propiedades de cierre, algoritmos de decisión.	Pr. 5. Gram. regulares y AFs. AFs y expresiones regulares (Kleene). (Comprobación) [JFLAP]
8	8. *Gramáticas Libres del Contexto* (4h) Transformación (limpieza) de GLCs. Formas normales (Chomsky y Greibach)	Pr. 6: AF→ER, ecuaciones características. ER→AF, derivadas. (comprobación) [JFLAP]
9	8. *Gramáticas Libres del Contexto* (1h) 9. *Autómatas con Pila* (3h) Motivación, conceptos y ejemplos.	Pr. 7: Transformación de Gramáticas libres de contexto. (comprobación) [PÂTÉ]
10	9. *Autómatas con Pila* (2h). Equivalencias $L_F \leftrightarrow L_V$, GLC↔AP 10. *Propiedades de los L.L.C.* (2h) Propiedades de iteración (lema de bombeo).	Pr. 8: Diseño y prueba de distintos autómatas con pila. [JFLAP]
11	10. *Propiedades de los L.L.C.* (2h) Propiedades de cierre y algoritmos de decisión. 11. *Máquinas de Turing* (2h) MT como reconocedor de lenguajes, conceptos y ejemplos. Variaciones al modelo de MT.	Pr. 9: Equivalencias $L_F \leftrightarrow L_V \leftrightarrow L_{VF}$. GLC↔AP (Comprobación). [JFLAP]
12	11. *Máquinas de Turing* (4h) MT y lenguajes de tipo 0. Aut. Linealmente acotados. MT como computador de funciones.	Pr 10: Diseño y prueba de MTs para reconocer diversos lenguajes. [JFLAP]
13	11. *Máquinas de Turing* (4h) Esquemas de MT. MT Universal, problema de la parada.	Pr. 11: Diseño y prueba de MTs para computar distintas funciones [JFLAP]
14	12. *Funciones Recursivas* (4h) Recursión primitiva, μ-recursión, equivalencia con MT.	Pr. 12: Computación de funciones usando esquemas de MT. [MTURING]

6. CONCLUSIONES.

Hemos pretendido, sobre todo, manifestar que la docencia de TALF puede apoyarse en un sistema de prácticas sin tareas de programación adicional. Para ello, hemos propuesto el uso de un conjunto de herramientas que permite tanto la simulación como la resolución-visualización

de problemas: PÀTÉ, JFLAP y MTURING. Además, hemos diseñado un temario de prácticas que permite usar el entorno, de forma regular, a lo largo de todo el curso, sin descartar ninguno de los temas esenciales.

7. REFERENCIAS.

(Bilska y col., 1997) A. Bilska, K. Leider, M. Procopiuc, O. Procopiuc, S. Rodger, J. Salemme y E. Tsang. *A Collection of Tools for making Automata Theory and Formal Languages Come alive*. Twenty-eight SIGCSE Technical Symposium on Computer Science Education, pp. 15-19. 1997.

(Hopcroft y Ullman, 1979) J.E. Hopcroft y J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

(Hung y Rodger, 2000) T. Hung y S. Rodger. *Increasing Visualization and Interaction in the Automata Theory Course*. Thirty-first SIGCSE Technical Symposium on Computer Science Education, pp. 6-10. 2000.

(Isasi y col., 1997) P. Isasi, P. Martínez y D. Borrajo. *Lenguajes, Gramáticas y Autómatas. Un enfoque práctico*. Addison-Wesley, 1997.

(Jiménez, 2000) L. Jiménez. Página en Internet. **http://oreto.inf-cr.uclm.es/personas/ljimenez/index.html**

(Lewis y Papadimitriou, 1998) H. Lewis y C. Papadimitriou. *Elements of the Theory of Computation (2ª ed.)*. Prentice-Hall, 1998.

(Rodger, 2000) S. Rodger. Página en Internet. **http://www.cs.duke.edu/~rodger**