Implementación de una herramienta didáctica para la manipulación de autómatas finitos y expresiones regulares

Joan Cabós [1], Josep M. Miret [2], Magda Valls [2]

[1] Dept. Informàtica i Enginyeria Industrial
[2] Dept. Matemàtica
Escola Universitària Politècnica
Universitat de Lleida
{jcabos,miret,magda}@eup.udl.es

Resumen

En este trabajo presentamos una herramienta que pretende facilitar la labor docente de la asignatura Lenguajes, gramáticas y autómatas, *y en particular el uso y manejo de autómatas finitos y expresiones regulares. Esta aplicación permite completar el temario de la asignatura ofreciendo al estudiante la posibilidad de realizar prácticas en el laboratorio, encaminadas tanto a la manipulación y comprensión de los objetos presentados en el aula, como a la incorporación de nuevos algoritmos.*

1 Introducción

En la docencia de matemáticas que se ofrece en una ingeniería técnica en informática uno puede hallar, a parte de las asignaturas básicas de álgebra y cálculo, otras asignaturas específicas en las que se recogen aquellos conocimientos que un informático requiere para su formación. En este segundo grupo, a caballo muchas veces entre las matemáticas y las ciencias de la computación, se halla la asignatura de *Lenguajes, gramáticas y autómatas*, cuyo objetivo es el de ofrecer la base matemática y rigor necesarios para introducir al estudiante en el campo de la compilación.

Una de las partes más importantes de esta asignatura se centra en el estudio de los autómatas finitos y su relación con las expresiones regulares. Para la manipulación de tales autómatas resulta útil utilizar una representación gráfica de los mismos, que facilite su comprensión por parte de los estudiantes.

Para ello, creímos que sería de gran ayuda poder disponer de una interficie gráfica sencilla que permitiera visualizarlos y realizar operaciones sobre ellos de forma automática, así como manipular expresiones regulares. Tal herramienta facilitaría al alumno comprender la equivalencia existente entre autómatas y expresiones regulares.

En este trabajo recopilamos, en primer lugar, aquellos conceptos básicos necesarios para la comprensión de los métodos de conversión que abordamos en las secciones tercera y cuarta. Cabe señalar, que ambas secciones se han ilustrado con ejemplos que ayudan a clarificar su funcionamiento. Finalmente, hemos añadido varios comentarios sobre la implementación de la herramienta, sus funcionalidades y su uso en el aula.

2 Conceptos básicos

El objetivo de esta sección es el de situar el punto de partida de la presente exposición, ofreciendo las notaciones de aquellos conceptos básicos a los que nos referiremos posteriormente. Para las definiciones, ejemplos y resultados sobre dichos conceptos véase, por ejemplo, [6].

- Denotamos por Σ un alfabeto y por Σ^* el lenguaje universal sobre Σ.
- La estrella de Kleene de un lenguaje L la designamos por L^*. La concatenación y unión de los lenguajes L_1 y L_2 las denotamos por $L_1 \cdot L_2$ y $L_1 \cup L_2$, respectivamente.

- Un autómata finito determinista (AFD) con un conjunto finito de estados Q, sobre el alfabeto Σ, estado inicial q_0, conjunto de estados finales F y función de transición δ, lo denotamos por $M = (Q, \Sigma, \delta, q_0, F)$.

- De forma similar, un autómata finito no determinista (AFN) viene representado por $M = (Q, \Sigma, \delta, I, F)$, donde I es el conjunto de estados iniciales.

- Una expresión regular vendrá denotada por E. La expresión regular vacía la denotamos por $\emptyset$.

- Las operaciones concatenación, unión y estrella de Kleene de expressiones regulares las denotamos, respectivamente, por $E_1 \cdot E_2$, $E_1 + E_2$ y E^*.

- El lenguaje asociado a una expresión regular E lo denotamos por $L(E)$.

Es conocido que todo lenguaje reconocido por un autómata finito determinista es a su vez un lenguaje asociado a una expresión regular, y viceversa [5].

Finalmente, cabe añadir que, dado el alfabeto Σ, consideramos la gramática

$$G = (\{E\}, \{\cdot, +, (,), *\} \cup \Sigma, P, E)$$

de las expresiones regulares sobre Σ, donde el conjunto P de producciones está formado por

$$\begin{aligned} E &\longrightarrow (E) \\ E &\longrightarrow E \cdot E \\ E &\longrightarrow E + E \\ E &\longrightarrow E^* \\ E &\longrightarrow a, \forall a \in \Sigma. \end{aligned}$$

Entonces, el árbol de análisis sintáctico indica gráficamente cómo desde el símbolo inicial de una gramática deriva una cadena del lenguaje.

3 Conversión de una expresión regular a un autómata

Dada la equivalencia entre lenguajes y expresiones regulares, existen algunos métodos que permiten, a partir de una expresión regular, hallar un autómata finito que la reconoce. Tales procesos son útiles, puesto que permiten construir reconocedores de lenguajes específicos de cierta complejidad cuyo autómata asociado no sea, inicialmente, fácil de determinar.

El método que hemos utilizado en nuestra implementación [1] se basa en el estudio de ciertas funciones definidas sobre los nodos del árbol sintáctico T correspondiente a la expresión regular E. Una vez obtenido el árbol sintáctico, se enumeran sus hojas según la posición que ocupan en la expresión regular. Entonces, las funciones *anulable(n)*, *primerapos(n)* y *ultimapos(n)*, definidas sobre cada uno de los nodos de T, se pueden hallar a partir de las reglas siguientes:

- Si el nodo n es una hoja numerada con i, no es anulable. Las funciones *primerapos(n)* y *ultimapos(n)* toman el valor i.

- Si el nodo n representa el símbolo $+$ con hijo izquierdo c_1 y derecho c_2, es anulable si lo son c_1 o c_2. La función *primerapos(n)* toma el valor $\mathit{primerapos}(c_1) \cup \mathit{primerapos}(c_2)$, y la función *ultimapos(n)* toma el valor $\mathit{ultimapos}(c_1) \cup \mathit{ultimapos}(c_2)$.

- Si el nodo n representa el símbolo $\cdot$ con hijo izquierdo c_1 y derecho c_2, es anulable si lo son c_1 y c_2. Si c_1 es anulable entonces *primerapos(n)* toma el valor $\mathit{primerapos}(c_1) \cup \mathit{ultimapos}(c_2)$, y si no es anulable $\mathit{primerapos}(c_1)$. Similarmente, si c_2 es anulable entonces *ultimapos(n)* toma el valor $\mathit{ultimapos}(c_1) \cup \mathit{ultimapos}(c_2)$, y si no es anulable $\mathit{ultimapos}(c_2)$.

- Si el nodo n representa el símbolo $*$ es anulable. Las funciones *primerapos(n)* y *ultimapos(n)* toman los valores $\mathit{primerapos}(c_1)$ y $\mathit{ultimapos}(c_1)$, respectivamente.

La función *siguientepos(i)*, definida sobre cada una de las posibles posiciones de los símbolos de Σ en la expresión regular, se calcula a partir de un recorrido en profundidad de T:

- Si n es un nodo con el símbolo $\cdot$ con hijo izquierdo c_1 y derecho c_2, y i pertenece a $\mathit{ultimapos}(c_1)$, entonces todos los elementos de $\mathit{primerapos}(c_2)$ pertenecen a *siguientepos(i)*.

- Si n es un nodo etiquetado por $*$ y i pertenece a *ultimapos*(n), entonces todos los elementos de *primerapos*(n) estan en *siguientepos*(i).

A continuación, describimos el algoritmo que, a partir de las funciones anteriores, permite construir el autómata finito correspondiente.
Entrada: Una expresión regular E.
Salida: Un AFD que reconoce $L(E)$.

Algoritmo.

1. Construir el árbol sintáctico T para la expresión $E\#$, donde $\#$ es un marcador final. Etiquetar las hojas del árbol con los elementos del conjunto $S = \{1, 2, \ldots, l\}$ según la posición que ocupan en la expresión regular.

2. Obtener las funciones *anulable, primerapos, ultimapos* y *siguientepos.*

3. Los estados del autómata son elementos del conjunto partes de S, $\mathcal{P}(S)$, y el estado inicial es *primerapos*(n_0), donde n_0 es el nodo que representa la raíz de T.

4. Para hallar el estado $q = \delta(p, a)$, donde $p \in \mathcal{P}(S)$ es un estado y a es un símbolo del alfabeto Σ, consideramos todas aquellas posiciones $i \in p$ tales que estan ocupadas por a. Entonces *siguientepos*$(i) \in q$.

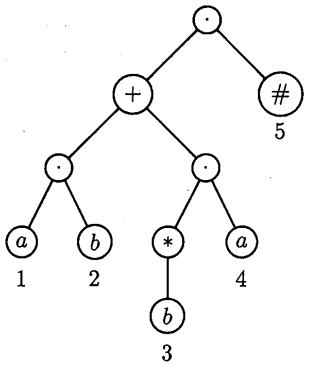
5. Aquellos estados p tales que contienen la posición correspondiente al símbolo $\#$, son estados finales.

Cabe señalar que si el autómata resultante es AFN, es preciso utilizar un algoritmo de determinización para obtener un AFD.

Ejemplo:

Dada la expresión regular $E = a \cdot b + b^* \cdot a$, utilizaremos el anterior procedimiento para determinar el autómata finito que reconoce el lenguaje regular asociado.

Para ello, construimos en primer lugar el árbol sintáctico correspondiente, cuyas hojas han sido numeradas según la posición que ocupan en la expresión $a \cdot b + b^* \cdot a\#$



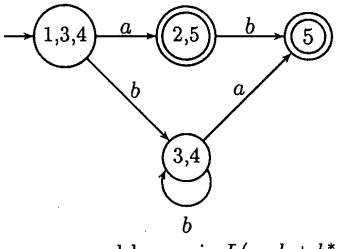
Las tablas siguientes reflejan el valor de las funciones *primerapos* y *ultimapos* para cada uno de los nodos del árbol, que hemos recorrido en post-orden.

Nodo	*primerapos*	*ultimapos*
a	$\{1\}$	$\{1\}$
b	$\{2\}$	$\{2\}$
$\cdot$	$\{1\}$	$\{2\}$
b	$\{3\}$	$\{3\}$
$*$	$\{3\}$	$\{3\}$
a	$\{4\}$	$\{4\}$
$\cdot$	$\{3,4\}$	$\{4\}$
$+$	$\{1,3,4\}$	$\{2,4\}$
$\#$	$\{5\}$	$\{5\}$
$\cdot$	$\{1,3,4\}$	$\{5\}$

Los valores de *siguientepos* para cada una de las posiciones de la expresión regular, resultan ser:

Posición	*siguientepos*
1	2
2	5
3	3,4
4	5
5	–

Y finalmente, el autómata que se obtiene es



que reconoce el lenguaje $L(a \cdot b + b^* \cdot a)$.

4 Conversión de un autómata a una expresión regular

Entre los diferentes métodos que existen para hallar la expresión regular correspondiente a un autómata finito, hemos implementado aquel que se basa en una aplicación del lema de Arden, por su eficiencia y sencillez [4]. Para ello el autómata se representa mediante un sistema de ecuaciones lineales cuyos coeficientes son ciertos lenguajes asociados al autómata.

Para determinar el sistema de ecuaciones lineales asociado al autómata, procederemos de la forma siguiente:

- Se enumeran los n estados del autómata desde 0 a $n-1$, y para cada uno de ellos se define el lenguaje

$$L_i = \{\omega \in \Sigma^* \mid \tilde{\delta}(q_i, \omega) \in F\}.$$

- Cada una de las ecuaciones del sistema es de la forma

$$L_i = A_{i,0}L_0 + \ldots + A_{i,n-1}L_{n-1} + B_i.$$

  El lenguaje $A_{i,j}$ está formado por los símbolos para los que existe una transición del estado q_i a q_j. Si no existe ninguna transición de q_i a q_j, entonces $A_{i,j} = \emptyset$. El lenguaje $B_i = \lambda$ si $q_i \in F$, y será $\emptyset$ en caso contrario. Notemos que L_0 es el lenguaje aceptado por el autómata.

De este modo, el sistema asociado al autómata es

$$\begin{pmatrix} L_0 \\ \vdots \\ L_{n-1} \end{pmatrix} = A \begin{pmatrix} L_0 \\ \vdots \\ L_{n-1} \end{pmatrix} + \begin{pmatrix} B_0 \\ \vdots \\ B_{n-1} \end{pmatrix}$$

donde A es la matriz

$$\begin{pmatrix} A_{0,0} & \cdots & A_{0,n-1} \\ \vdots & \ddots & \vdots \\ A_{n-1,0} & \cdots & A_{n-1,n-1} \end{pmatrix}$$

Ahora, para simplificar el sistema, se utiliza el lema de Arden en la última ecuación, obteniendo

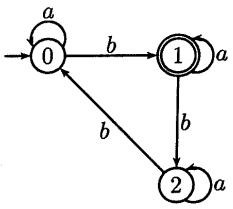
$$\begin{aligned} L_{n-1} = \; & A^*_{n-1,n-1}\,(A_{n-1,0}L_0 + \cdots \\ & \cdots + A_{n-1,n-2}L_{n-2} + B_{n-1}) \end{aligned}$$

que substituimos en las restantes ecuaciones del sistema. Este procedimiento se aplica sucesivamente hasta obtener una ecuación de la forma $L_0 = C_0 \cdot L_0 + C_1$. Finalmente, por el lema de Arden, se obtiene que

$$L_0 = C_0^* \cdot C_1.$$

Ejemplo:

Dado el autómata



el sistema lineal asociado es

$$\begin{cases} L_0 &= aL_0 + bL_1 \\ L_1 &= aL_1 + bL_2 + \lambda \\ L_2 &= bL_0 + aL_2 \end{cases}$$

cuya matriz ampliada es

$$\begin{pmatrix} a & b & \emptyset & \emptyset \\ \emptyset & a & b & \lambda \\ b & \emptyset & a & \emptyset \end{pmatrix}$$

que mediante las transformaciones indicadas se reduce

$$\begin{pmatrix} a & b & \emptyset \\ ba^*b & a & \lambda \end{pmatrix} \longrightarrow \begin{pmatrix} a + ba^*ba^*b & ba^* \end{pmatrix}$$

Finalmente, aplicando Arden a la ecuación $L_0 = (a + ba^*ba^*b)L_0 + ba^*$ se obtiene

$$L_0 = (a + ba^*ba^*b)^*ba^*$$

que es una de las posibles expresiones regulares asociadas al autómata.

5 Herramienta didáctica

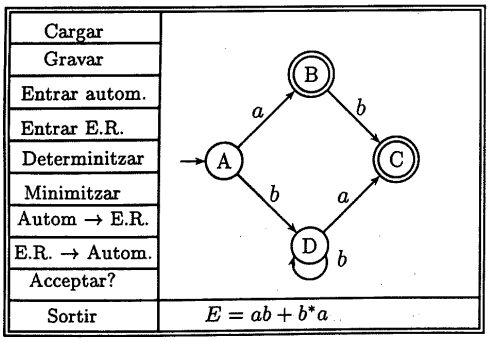
Uno de los objetivos del trabajo que presentamos ha sido diseñar e implementar una aplicación que, por una parte, proporciona al profesor una herramienta atractiva para motivar la asignatura y, por otra, facilita al estudiante la comprensión y el manejo de los autómatas finitos.

Las características y funcionalidades principales de la herramienta son:

- Edición gráfica e interactiva de un autómata finito.
- Estudio de las palabras que reconoce tal autómata.
- Análisis sintáctico de una expresión regular.
- Conversión de expresiones regulares a autómatas.
- Conversión de autómatas a expresiones regulares.
- Minimización y determinización de autómatas.

Asimismo, en el diseño del programa se han tenido en cuenta las posibles aplicaciones didácticas del mismo. No solamente puede ser utilizado en el aula para facilitar el contacto de los alumnos con la manipulación de autómatas y expresiones regulares (gracias a las funcionalidades establecidas), sino que ha estado diseñado modularmente de modo que los propios alumnos puedan incorporar nuevos módulos en el sistema, tales como posibilitar el uso de varios autómatas simultáneamente, las operaciones entre ellos, diseñar nuevos algoritmos de determinización y minimización, haciendo un estudio comparativo entre ellos, implementar y analizar otros métodos de conversión entre autómatas y expresiones regulares, etc.

El programa se ha realizado en C bajo el entorno MS-DOS, implementando tipos abstractos de datos [2, 3] para encapsular las operaciones de los distintos objetos, consiguiendo de esta forma un diseño del programa más simple y entendedor.



Finalmente, cabe señalar que esta herramienta ha sido incorporada como horas de laboratorio en la docencia de la asignatura, con una muy buena aceptación por parte del alumnado.

Referencias

[1] A. Aho, R. Sethi, and J. Ullman. *Compiladores. Principios, técnicas y herramientas.* Addison-Wesley Iberoamericana, 1990.

[2] X. Franch. *Especificació algebraica de tipus abstracte de dades.* Océ España S.A., 1991.

[3] X. Franch. *Estructura de dades. Especificación, disenys i implementació.* Edicions U.P.C., 1993.

[4] J. Gabarró. *Informàtica clàssica. Part I.* Centre de Publicacions del Campus Nord. U.P.C.

[5] J.E. Hopcroft and J.D. Ullman. *Introduction to automata theory, languages and computation.* Addison-Wesley, 1979.

[6] D. Wood. *Theory of computation.* Wiley, 1987.