Departamento de Computación,
Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires

Trabajo Práctico 1

Implementación de AFDs

Teoría de Lenguajes

Primer Cuatrimestre de 2015

Grupo: Autores del Autómata Automático Autodestructivo

Apellido y Nombre	LU	E-mail
Matayoshi, Leandro	79/11	leandro.matayoshi@gmail.com
Unbekant, Ignacio	722/10	ignacio.niesz@gmail.com
Vega, Leandro	698/11	leandrogvega@gmail.com

Índice

1. Descripción de la Implementación

1.1. Explicación del armado del autómata finito:

Para realizar esta función nosotros partimos de un lenguaje regular parseado, el cual está explicado en la sección **Parseo de Regex**. A este mismo lo leeremos e iremos armando el autómata finito correspondiente. La forma de generar nuestro autómata finito es por medio del método de Thompson. Básicamente el algoritmo funciona de forma recursiva y lo detallaremos a continuacion:

- Se fija si debe realizar *CONCAT*, *PLUS*, *STAR*, *OR*, *OPT* o *SIMBOLO*.
- Si es *SIMBOLO* crea un automata que tiene dos estados y una transición con el símbolo correspondiente.
- En caso de no ser *SIMBOLO* va mirando los argumentos y, en cada caso, llama recursivamente a la función para que cree el autómata del mismo. Luego dentro del nivel en el que nos encontramos usamos la funcion *armarConcar*, *armarPlus* o *armarOr* según corresponda, de esta manera, cada una se encargarán de armar el autómata tal como lo describe Thompson. Cabe destacar que no hemos hecho un *armarStar* ni *armarOpt*, esto se debe a que el primero sólo agrega una transición más con respecto a *armarPlus*, y el segundo es un caso idéntico al *armarOr* definiendo uno de los parámetros como un automata con transición lambda.
- En el caso del *CONCAT* y *OR* que pueden tener n argumentos, hemos optado que el *armarConcat* y el *armarOr* sólo puedan ir armando de a dos autómatas y luego, al obtener el armado de ambos, buscar recurisvamente el autómata del siguiente argumento y repite el paso anterior sucesivamente.

1.2. Explicación del armado de un automata para hallar su complemento:

Para esta función tuvimos un problema, ya que para hallar el complemento de un autómata necesitamos, para un estado y un alfabeto, saber su transición aunque ésta no exista. Para solucionar esto agregamos un estado *"trampa"*, y nuestro algoritmo se encargará de generar las transiciones ausentes de nuestro autómata y llevarlas a este nuevo estado. Como sabemos este estado *"trampa"* debe cumplir con su definición, no debe ser un estado final y tampoco se puede salir de él por medio de alguna transición.

1.3. Equivalencia de autómatas

Para poder obtener la equivalencia pensamos diferentes estrategias:

- La primera fue la de ir recorriendo todos los caminos de los autómatas y ver si pasaban por las transiciones en las cuales ambos tenían el mismo caracter. La descartamos porque era engorrosa y difícil de implementar.
- La segunda idea que surgió fue la usar el complemento de alguno de los dos autómatas e intersecarlo con el otro, de esta manera, si ambos eran equivalente, el resultado será vacío. Con esto parecía que teníamos resuelto el ejercicio, pero notamos casos bordes donde no se satisfacía muy bien este método. Por ejemplo: ¿Qué pasa si uno de los dos autómatas (A1) era inicialmente vacío y el otro no (A2), y además, tomamos el complemento del último mencionado (A2)? Bueno, al intersecar ambos automatas el resultado sería vacío por definición misma de los conjuntos, y concluiríamos que son equivalentes, lo cual es un error.
- La tercera y última opción fue la siguiente, llamemos a uno de los dos autómatas A1 y al otro A2. Si "tomamos el complemento de A2 y lo intersecamos con A1"(A3), y "tomamos el complemento de A1 y lo intersecamos con A2"(A4), entonces si tanto A3 y A4 dieron vacío podremos afirmar que son equivalentes. Se puede ver fácilmente que este caso cubre al mencionado en la segundo idea, puesto que si miramos la parte agregada recientemente nos comprobará que no son equivalente. Esto es, tomamos al que inicialmente era vacío (A1) su complemento, esto nos da un automata que acepta todas las cadenas, y lo intersecamos con uno no vacío (A2), el resultado es exactamente el autómata A2. Entonces, dado esta nueva propuesta, podemos concluir que no son

equivalente, lo cual efectivamente es verdad. Además, para esta tercera opción, pudimos encontrar una demostración que indica que nuestra propuesta marca formalmente una equivalencia entre ambos autómatas.