



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA



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 | leandrogrvega@gmail.com |

Índice

| | |
|--|----------|
| 1. Descripción de la Implementación | 2 |
| 1.1. Explicación del armado del autómata finito: | 2 |
| 1.2. Explicación del armado de un automata para hallar su complemento: | 2 |

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 continuación:

- 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 función *armarConcat*, *armarPlus* o *armarOr* según corresponda, de esta manera, cada una se encargará 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 recursivamente 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.