# Práctica PRDE – Implementación de teoría de autómatas

## 1. Introducción

El propósito de esta práctica es implementar tipos de datos en Haskell que representen los autómatas (deterministas, no deterministas y no deterministas con transiciones épsilon) y las expresiones regulares de manera que podamos aplicar los algoritmos que se estudian en teoría de autómatas a estos (los cuales se implementaran también). Estos algoritmos incluyen la detección de una cadena en una máquina (autómata o expresión regular), las conversiones entre máquinas (de cualquier tipo a cualquier tipo) y un repertorio de varios algoritmos más.

## 2. Metodología

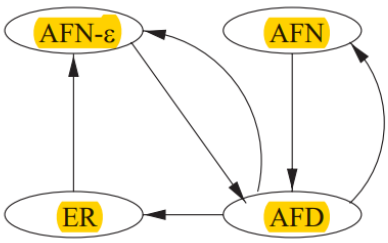
La primera parte del trabajo fue decidir como quería que fuesen los tipos de los autómatas y las expresiones regulares. Las expresiones regulares dentro de la propia teoría de autómatas no tienen una definición con un tipo por debajo, sino que se definen en base a constructoras por lo que parece intuitivo utilizar el data de Haskell para implementarlas. Sin embargo, los autómatas en teoría de autómatas se definen como una 5-tupla donde si tenemos definidos los conjuntos no necesitamos ningún tipo más, por lo que parece intuitivo también implementarlos usando el type de Haskell.
El único problema con la implementación entonces sería como definir el tipo de los estados, para mi hay dos opciones:

1- Son un String y los conjuntos de estados tenemos que verlos como el String que te representa el conjunto.
2- Definimos un data Estado = String | Conj (Cjto Estado).

La versión 2 es la que me parece mejor porque te ahorras tener que ordenar los conjuntos de estados (para poder comparar cuando se conviertan a String), pero me daba algunos fallos, por lo que opte por la versión 1 la cual funciona perfectamente.

Una vez definidos los tipos lo siguiente que hice fue programar los algoritmos que te permiten detectar cadenas para cada tipo, sin tener que convertir entre máquinas. Como todos funcionaban bien supuse que la implementación de los tipos estaba bien.

Lo siguiente que hice fue programar las conversiones entre tipos, para ello usé el siguiente esquema que aparece en el libro de Hopcroft, implementando las conversiones como se hace en ese libro:




Por último, hice una lista con algoritmos del libro de Hopcroft y de las hojas de ejercicios de la asignatura de autómatas y los implemente.

## 3. Resultados y conclusión

La implementación funciona como se esperaría. Al hacer las conversiones entre tipos quedan autómatas feos, pero haciéndolo a mano pasa igual.
Por último, para hacer pruebas con el programa, he incluido un fichero instrucciones.txt donde explico como usarlo.