Teoría de la Computación

Grado en Ingeniería Informática - 2021/22

Práctica 1 (Expresiones regulares y autómatas)

El objetivo de esta práctica es reforzar los conceptos relacionados con las expresiones regulares, con los autómatas finitos, y con algunas de las transformaciones básicas que se pueden realizar entre estos formalismos.

Descarga el fichero ocaml-talf.tar.gz y descomprímelo en un directorio propio, porque algunas de las operaciones están ya implementadas:

- Se proporcionan dos funciones, *cadena_of_string* y *cadena_of_file*, para construir valores del tipo de dato simbolo list.
- Se proporcionan dos funciones, *er_of_string* y *er_of_file*, para construir valores del tipo de dato er.
- Se proporcionan dos funciones, *af_of_string* y *af_of_file*, para construir valores del tipo de dato af.
- Se proporciona también la función *dibuja_af*, para poder visualizar y/o imprimir la representación gráfica de un valor de tipo af.
- Se proporciona la función *escaner_af* : Auto.simbolo list -> Auto.af -> bool, que verifica si una cadena de símbolos terminales es aceptada o no por un autómata finito.

Se pide resolver los siguientes apartados:

- Implementa una función **traza_af** : Auto.simbolo list -> Auto.af -> bool, o bien modificando la salida, que no sólo verifique si una cadena de símbolos terminales es aceptada o no por un autómata finito, sino que además imprima por pantalla todas las configuraciones instantáneas, es decir, todos los pares de la forma (estados actuales, símbolos pendientes), por los que va pasando el proceso de reconocimiento de dicha cadena.
- Implementa una función **cartesiano_af** : Auto.af -> Auto.af, que dados dos autómatas finitos deterministas devuelve otro autómata finito capaz de simular en paralelo el funcionamiento de los dos autómatas anteriores.
- (Opcional) Encuentra un ejemplo de uso complejo y elaborado de la función anterior. Ej. Sobre el alfabeto {a,b}, el autómata a1 acepta cadenas de longitud par. El autómata a2 acepta cadenas que terminan en b. El autómata a12 acepta cadenas que cumplen ambas propiedades.

```
let a1 = af_of_string "0 1; a b; 0; 0; 0 1 a; 0 1 b; 1 0 a; 1 0 b;";;
 val a1 : Auto.af = ...

 let a2 = af_of_string "0 1; a b; 0; 1; 0 0 a; 0 0 b; 0 1 b;";;
 val a2 : Auto.af = ...

 let a12 = cartesiano_af a1 a2;;
 val a12 : Auto.af = ...

 escaner_af (cadena_of_string "a b b a b") a12;;
 - : bool = false

 escaner_af (cadena_of_string "a b a b") a12;;
 - : bool = true
```

Para ello, echa primero un vistazo al fichero **manual.pdf**, el cual contiene la definición de los tipos de datos involucrados, con sus correspondientes constructores. Realiza las implementaciones en un fichero de nombre p1.ml, que es el único que debes entregar después.