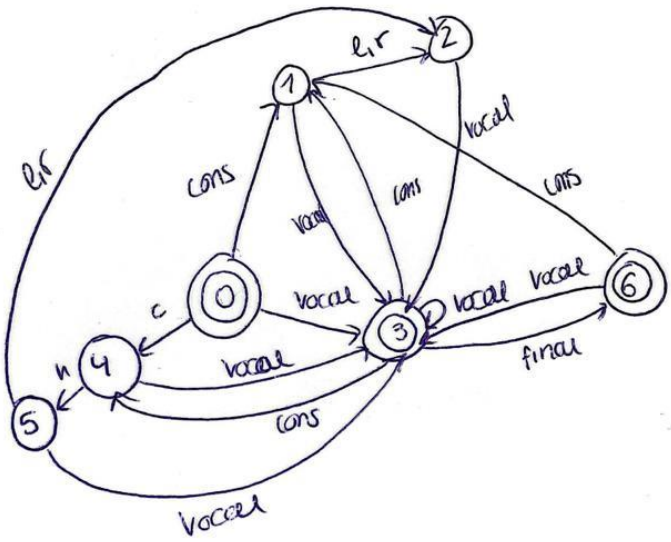
Programación Funcional

Práctica 4: jugando con las palabras **Lucía Jiménez**

Implementación

Para realizar la práctica se han implementado 4 módulos:

1. AFD
   En este módulo implementamos un autómata finito determinista, con las reglas dadas en el enunciado, que decidirá si una cadena es correcta o no.
   La gramática construida es esta:
   Tenemos que saber que hay varios tipos de letras: consonante(cons), vocal(v) y letras finales de sílaba(f).
   S -> consA | vocalC | cD |λ
   A -> lB | rB | vocalC
   B -> vocalC
   C -> vocalC | fF |consA | cD | λ
   D -> hE | lB |rB | vocalC
   E -> lB | rB | vocalC
   F -> vocalC | consA | λ
   Con esta gramática conseguimos que nuestro autómata acepte palabras que cumplan las reglas especificadas en el enunciado.
   Para que la letra 'ch' se interprete como una sola, hemos tenido que separar casos, añadiendo estados extras.



   En la implementación del autómata definimos la función de transición que vamos a utilizar y el autómata , con su alfabeto, sus estados y sus estados finales. Hemos tenido que definir un estado fantasma (Q 7) , al que se transita cuando no se ha encontrado ningún otro sitio al que transitar , nos sirve para detectar palabras que no cumplan el patrón , ya que si no lo añadimos generaremos una excepción en el programa. La función accept

nos devuelve true cuando el último estado al que ha transitado el AFD es estado final y false cuando el último estado no es final.

2. Anagrama
   En este modulo implementamos las funciones necesarias para generar anagramas, y para comprobar que los anagramas generados son aceptados por el autómata, y además otra que compruebe que están en el diccionario. Además hemos implementado otra función que quita las palabras duplicadas ya que en caso de que una palabra tenga una letra repetida, nos va a generar 2 anagramas iguales, y esto luego tendrá un coste computacional mayor a la hora de buscarlas en el diccionario.
3. Diccionario
   Hemos descargado un diccionario que contiene un fichero txt por cada letra del abecedario, en estos ficheros hemos hecho ciertos cambios para que funcione mejor como cambiar las letras con tildes a sin tildes y la ñ por ni, ya que nuestro lenguaje no acepta las tildes ni la ñ. Hay una limitación y es que las palabras que son adjetivos vienen escritas en este formato guapo, a , por tanto es mas difícil reconocerlas y hemos borrado la coma , por lo que nuestro diccionario no contiene todos los adjetivos.
   Para buscar si una palabra esta en el diccionario , se abrirá el txt con la primera letra de la palabra y se sacará la primera palabra y se comparará , en caso de que no coincida se eliminará(no del fichero), y se sacará la siguiente, y así hasta encontrar una que coincida. En caso de llegar a la lista vacía se devolverá que la palabra no se encuentra.
   Esto hace que comprobar la lista de anagramas que están en el diccionario sea una tarea muy lenta, ya que cada vez que cambie la primera letra de la palabra habrá que abrir otro txt.
4. PF_Practica4
   Aquí implementamos el menú del programa, habrá que compilarlo y llamar a main. Como recomendación para hacer pruebas:
   1. Para ver si una palabra es correcta, no hay limitación en cuanto a la longitud de la palabra
   2. Para generar anagramas si, por una parte una palabra de longitud n tiene n! anagramas , por tanto si introducimos una palabra de más de 6 caracteres el programa tardará tiempo en ejecutarse, y esto se suma a que luego tendrá que ir palabra por palabra , abriendo un fichero distinto (ya que cada una comienza por una letra distinta) y buscando , para comprobar que esta en el diccionario, lo que supone mucha carga para el programa.
   3. Al igual que para comprobar las palabras correctas , aquí no hay limite de longitud.

   Por tanto recomendamos no utilizar palabras de más de 5 letras , ya que a demás los resultados se verán mas claros.