Escuela Politécnica Superior de Lleida

Practica Autómata

Jerarquía de memoria

Autor:
Martin, Francisco Manuel

DNI:
48057095k

April 7, 2020

Índex

1 Introducción

En esta practica he realizado el proceso de reconocimiento de un autómata, su lectura, la minimización y la determinación de alguno de ellos. He realizado esta practica con el lenguaje de programación *python*, he creado cuatro clases las cuales me ayudaran a desglosar los problemas que la practica requería. Me he basado mayormente en una programación orientada a la inmutabilidad y en la simplicidad de las funciones. Las clases que he creado y que mas tarde explicare con mas detalle son : **dfa**(*deterministic finit automaton*), **fa** (*finit automaton*), **nfa**(*non deterninistic finit automaton*) y por ultimo **StateFa**.Estas definirán el comportamiento de nuestros autómatas.

2 Interfaces para los autómatas

Para dar un comportamiento homogéneo en todas las clases creo una interfaces común para todas las clases que antes había nombrado. Esto ayudara en la herencia de los métodos i poder reutilizar todos los métodos en la clase *fa*.

- **Dictionary**: Nos devolverá una tabla de *hash* o diccionario del propio autómata
- **Read**: Todos nuestros autómatas tienen que tener la capacidad de leer una palabra i devolver si el lenguaje usado es aceptado o encaso contrario decir que no lo es.
- **Automaton**: devuelve un diccionario mas detallado sobre el autómata, es el valor de entrada de este.
- **States**: Devuelve una lista de los estados que contiene.
- **Alphabet**: Alfabeto que acepta el autómata.
- **DotDictionary**: creara un fichero dot que la libreria **Graphviz** transformara en una imagen de un grafo grafo.

3 Clase para los estados

Cree esta clase para encapsular los valores de cada estado, es una clase anémica, solo nos provee de la información almacenada en sus instancias, por ello la única lógica que tiene es el retorno booleano en el caso que preguntemos si es un estado aceptador o un estado inicial.
Esta clase podría estar anidada en alguna de las clases creadas pero eso haría un código muy acoplado y dejaría poco margen para testear i poner mas comportamiento en esta clase, por ello pensé que desacoplar-la seria la mejor opción. Algo que nos provee python es el tipado dinámico que ayuda en el momento que queremos almacenar las funciones de transición en los estados, ya que en un autómata determinista estará formado por un símbolo y en el caso de un no determinista estará formado por una lista.

4 Automata Finito Determinista

Esta clase dará comportamiento a los autómatas finitos deterministas. Para inicializarla crearemos una tabla de hash que es la estructura de datos en la que me apoyare, una lista de los estados que serán inicializados dentro de esta clase y los valores del alfabeto. Aparte de las funciones descritas en el apartado de la interfaz esta clase también tendrá la función de minimizar.

4.1 Leer

Para comenzar en esta función cojo el estado inicial y mediante *tail recursion* voy cogiendo el primer elemento de la palabra y recorro los estados hasta que no tengo ningún elemento mas que leer, en ese momento devuelvo si el estado actual es un estado aceptador. En el caso que no se un símbolo valido devolverá **VaueError**.

4.2 Minimizar

Este método se encargara de devolver un autómata finito determinista minimizado. Primero con conjuntos organizo los estados aceptadores y los que no lo son. Mas tarde de forma recursiva con los conjuntos los encapsularemos en una lista y hasta que los conjuntos dentro de esta lista no paren de dividirse volveremos a ejecutar la función. Para dividir los estados hacemos una función que determine si según las transiciones del estado continúan perteneciendo al mismo conjunto, para ahorrarnos eliminar los conjuntos vacíos creo un filtro. Para terminar usamos una función para adjudicar las funciones de transición a cada nuevo estado, esta función es la que ha requerido mas tiempo de toda mi implementación, para que esta sea consistente requiere que los símbolos que les identifiquen se puedan ordenar para trabajar con la lista que inicializamos al principio. Por ultimo, como quería que mi código fuera inmutable, creo otra instancia de autómata finito determinista con las nuevas propiedades.