Implementación de una clase para gramáticas regulares

Doble Grado Ingeniería Informática y Matemáticas
Modelos de computación

David Cabezas Berrido
Francisco Miguel Castro Macías

08/11/2018

Esta práctica consiste en, a partir de un código en python, implementar una clase para representar gramáticas regulares (tipo 3). Se detalla a continuación el formato que deben presentar estas gramáticas para la correcta lectura y escritura, así como la forma en que se han implementado las diversas funciones que se pedían.

1. Formato

Según nos dice el guión de la práctica, las gramáticas de tipo 3 se representarán así:

- Una línea con las variables. La primera de ellas será la variable inicial.
- Una línea con los símbolos terminales.
- Una línea por cada producción. Cada producción será de la forma 'A `->` α'.

2. Clase y funciones implementadas

Se ha creado la clase `RegularGrammar`, que tiene como atributos:

- `variables`: las variables que hay en la gramática.
- `alphabet`: el alfabeto.
- `productions`: un map en el que se guardan las producciones. Por cada variable almacenamos todas las producciones que salen de ella junto a la etiqueta.

```
'source': ['labeldestiny', 'labeldestiny', ...]
source -> labeldestiny
```

En esta clase se han creado las siguientes funciones:

- `readFromFile(d)`: si `d` es un fichero que contiene una gramática, la lee.
- `writeToFile(d)`: escribe la gramática en un fichero con el formato especificado.
- `checkLinearIzdaDcha()`: comprueba si la gramática que contiene es linear por la izquierda o linear por la derecha. Devuelve un par `(i, d)` donde `i=true` si es linear por la izquierda y `i=false` en otro caso. Análogo para `d`.
- `checkRegular()`: devuelve `true` si la gramática es regular y `false` en otro caso.
- `toAFND()`: genera un objeto de la clase `afd`, en concreto un autómata finito no determinista con transiciones nulas, que acepta el lenguaje representado por la gramática. Como precondición se impone que la gramática sea linear por la derecha.

- `fromAFND(aut)`: construye la gramática linear por la derecha que genera el lenguaje aceptado por el autómada `aut`.

Debemos señalar que somos conscientes de que también se puede pasar de gramática a autómata aun si la gramática es linear por la izquierda. Sin embargo, esto conlleva invertir la gramática, obtener el autómata, invertir el autómata y obtener la gramática de este último autómata.

3. Test y ficheros de prueba

Se proporciona por último una serie de ficheros de prueba y una función `testGrammar()` donde se leen las gramáticas de estos ficheros y se ejecutan las funciones implementadas.