Pràctica Estructures de la informació

Codificador - Fase 5

Construcció i persistència del codi

Biel Moyà

2013-2014

En aquest punt de la pràctica ja tenim construït l'arbre que ens permet generar les seqüències de codificació / descodificació.

Abans de recorrer l'arbre ens serà necessàri seguir completant el package definit en la fase anterior. En aquesta fase definirem les estructures i fitxers que ens permetran guardar i carregar el nostre codi en la memòria.

Degut a que aquestes eines només son emprades en les operacions del paquet codificador, no ens interessa que siguin conegudes fora d'aquest.

Codificació / Decodificació

Codificació

Per tal de guardar els codis binàris que ens permeten codificar un text emprarem una taula d'accés directe de la següent forma:

Símbol	Codificació
a	10
b	11
c	0

Figura 1: Taula de codificació.

Observau que tal com hem fet a la taula de freqüències, ens pot ser molt pràctic que els index de la taula de codificació siguin els símbols amb els quals codificam.

Un cop tenim aquesta taula construïda serà necessàri escriure-la en un fitxer, amb l'extensió **.co*

Decodificació

Per tal de poder decodificar un text, transformarem el nostre arbre en un autòmat. Degut a la pròpia estructura de l'arbre codificador, l'autòmat que construirem serà finit determinista.

Necessitam:

- Definir que és una transició del nostre autòmat. Una transició es una tupla de 4 valors.
  - Estat inicial
  - Símbol (0/1)
  - Estat final
  - Símbol (caràcter)
- Definir un fitxer sequencial capaç d'emmagatzemar transicions.
- Definir una estructura senzilla que un cop generat el codi, i cada vegada que necessitem decodificar un text ens permeti carregar les transicions (autòmat en forma de dues matrius).

En el següent esquema podeu observar les tres passes anteriors:

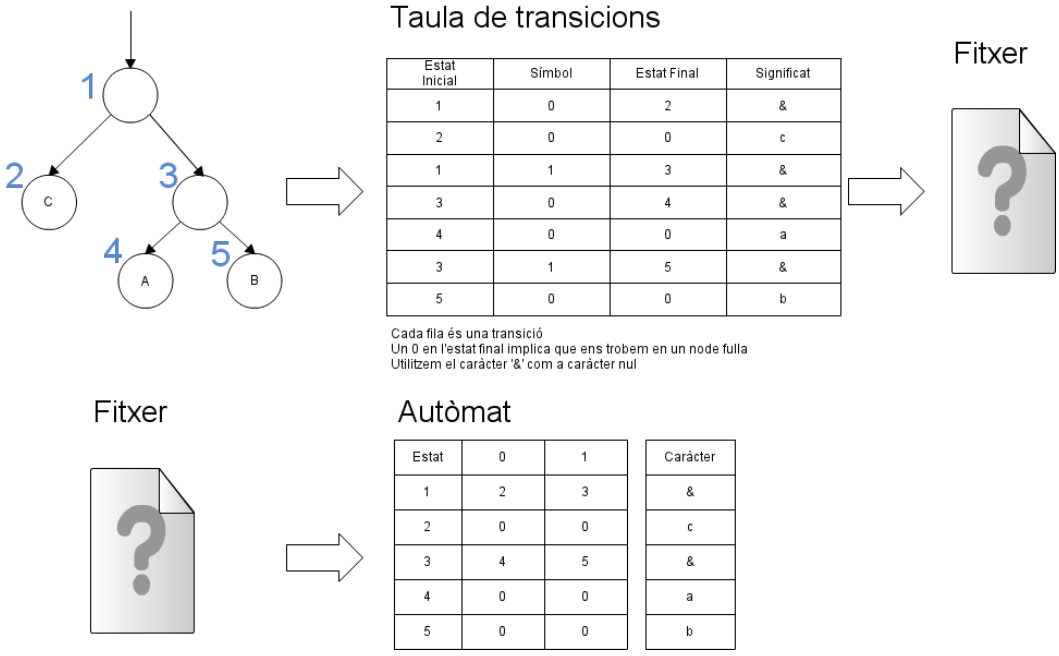

Figura 2: Passes per la decodificació

El fitxer de decodificació haurà de tenir l'extensió **.de*

Recorregut de l'arbre

Un cop tenim tot el necessàri ja només queda: recorrer l'arbre, extreure i emmagatzemar la informació que necessitam per codificar / decodificar.

L'esquema bàsic del recorregut és el seguent:

```
profunditat(a: arbre)
   si (a.tipus == pare) llavors
      generar_transicio
      profunditat(n.fe)
      generar_transicio
      profunditat(n.fd)
   si (a.tipus == fill) llavors
      generar_codi
   fi si
end profunditat;
```

Recomanacions:

- Encapsular la funció recorregut en una segona funció que faci les tasques d'obertura i clausura dels fitxers.
- Generar una funció privada que doni valor a una estructura transició i la escrigui en un fitxer.
- Recordar que cal numerar els estats per a la creació de les transicions, per això és recomana usar una variable global.

Sequential_io

Ada.Sequential_IO és un paquet de la biblioteca estàndar per realitzar tasques d'entrada sortida de fitxers sequencials.

Podem realitzar instàncies d'aquest paquet per poder escriure de forma còmode els tipus o estructures del nostre interés. En concret en aquesta pràctica, ens serà molt pràctic poder escriure i llegir transicions o la taula de codificació emprant les operacions *get* i *put*.

La seva instanciació és fa de la mateixa manera que qualsevol altra paquet genèric:

```
package Nom is new Ada.Sequential_IO(Tipus);
use Nom;
```