Méthodes Empiriques et Langages de Script
TP 6

G. A. Musillo
musillo4@etu.unige.ch

December 13, 2005

1 Exercice 1: un concordancier

Un concordancier est un outil d'exploration de données textuelles, qui, étant donné un corpus de textes et un mot, extrait les occurences de ce mot ainsi que les contextes gauche et droit dans lesquels ce mot apparaît.
On vous demande de programmer un tel concordancier. Si on fournissait, par exemple, à votre concordancier le paragraphe précédent et le mot `un`, il afficherait le résultat suivant:

```
                              un  concordancier est un outil d'expl...
         un concordancier est  un  outil d'exploration de donne...
        ..., qui, etant donne  un  corpus de textes et un mot,...
    ...un corpus de textes et  un  mot, extrait les occur...
```

2 Exercice 2: l'évaluation de références anonymes, les expressions régulières et les modules

Perl permet de créer des références sur des objets anonymes. Par exemple, le code suivant

```
$r_ano_array = [ 'a', 'b', 'c' ];
```

crée un tableau `('a', 'b', 'c')` et retourne une référence affectée à `$r_ano_array`. Perl interprète donc `[ 'a', 'b', 'c' ]` comme une référence sur un tableau. On dit que ce tableau est anonyme, car on ne peut y accéder que *via* la référence `$r_ano_array`.
On peut former des structures anonymes plus complexes: il suffit qu'un élément du tableau soit une référence sur un tableau anonyme. Par exemple, le code suivant

```
$r_ano_array = ['*', 'a', ['-', 'b', 'c', 'd']];
```

crée une structure enchassée qu'on peut interpréter comme un arbre et sur laquelle le scalaire `$r_ano_array` pointe.
La fonction `eval` de PERL est magique: elle interprète son argument, qu'elle considère comme du code PERL, et retourne sa valeur. Elle permet donc d'appeler l'interpréteur PERL dans le corps d'un programme. Il s'ensuit que si on lui passait une chaîne telle que

```
‘‘[’*’, ’a’, [’-’, ’b’, ’c’, ’d’]]’’
```

elle retournerait une référence sur la structure anonyme

```
[’*’, ’a’, [’-’, ’b’, ’c’, ’d’]].
```

On vous demande de coder un programme

- qui prend en entrée la représentation parenthésée d'un arbre syntaxique (vous trouverez dans le fichier `syns.dat` quelques exemples à traiter),
- qui la transforme au moyen d'expressions régulières de telle sorte qu'elle puisse être interprétée par la fonction `eval` comme une structure complexe
- et, enfin, qui affiche cette structure complexe en appelant la fonction `print_tree` que le module `PrettyPrinting.pm` exporte.