▶ **MP – Option Informatique**
TP Caml
sl@jachiet.com
http://louis.jachiet.com/tps
Vendredi 14 mars 2014

Automates d'arbres

Les automates d'arbres sont une extension des automates classiques (qui travaillent sur des chaînes de caractères) aux arbres. Il existe plusieurs variantes d'automates travaillant sur les arbres mais nous allons nous intéresser à la variante "ascendante binaire" qui ne s'occupe que des arbres binaires où les nœuds sont annotés par des labels.
Un automate d'arbre est défini par le quadruplet (Q, F, i, Δ) :

- Q est l'ensemble des états de l'automate
- $F \subset Q$ est l'ensemble des états finaux
- i est l'état initial (l'état des feuilles donc)
- $\Delta \subset Q \times \text{Label} \times Q \to Q$ est la fonction de transition. Ici on considère un automate déterministe et complet donc elle prend deux états (associés au nœud de gauche et de droite), un label (celui du nœud) et renvoie un état (celui associé au nœud)

1 Reconnaissance par automate d'arbre déterministe

1.1 Arbres

On pose tout d'abord le type d'arbre suivant :

```
type arbre =
                Feuille
              | Noeud of arbre*string*arbre
;;
```

Un exemple de tel arbre où les nœuds sont étiquetés "Rouge" ou "Noir" :

```
let a =
  Noeud(
   Noeud(Feuille,"Rouge",Feuille),
   "Rouge",
   Noeud(
       Feuille,
       Noir,
       Noeud(Feuille,"Rouge",Feuille)
       )
)
```

1.2 Automates

```
type automate =
{ initial : int ;
  labels : string array ;
  transition : int array array array ;
  final : bool array }
;;
```

Dans notre automate, les états sont des `int` et les labels des chaînes de caractères. Afin de pouvoir manipuler facilement les labels, on veut une fonction qui associe à un label un entier.

▶ **Question 1** *Écrire une fonction* `trad` *qui associe à un label et à un automate a, l'indice du label dans le tableau* `a.labels`.

1.3 Reconnaissance

▶ **Question 2** *Écrire une fonction qui prend en paramètre un arbre et un automate et qui renvoie si l'arbre est reconnu par l'automate.*

▶ **Question 3** *Donner un automate qui teste si un nœud annoté "Rouge" est fils d'un nœud lui aussi annoté "Rouge".*

2 Automates non déterministe

On relâche la contrainte de déterminisme. Le type de l'automate est donc maintenant :

```
type automate_nondeterministe =
{ initial : int ;
  labels : string array ;
  transition : int list array array array ;
  final : bool array }
;;
```

▶ **Question 4** *Écrire une fonction qui prend un automate et un arbre et qui teste si l'arbre est reconnu par l'automate.*

▶ **Question 5 *** *Écrire une fonction qui prend un automate non déterministe et renvoie un automate déterministe en utilisant l'algorithme des parties.*