Programmation fonctionnelle

Laura Kallmeyer

TD du 8.11.00

La solution des td du 18.10.00 (la fonction `recognize` pour un automate fini determiné) se trouve sous `http://talana.linguist.jussieu.fr/~lkallmey/LISP/fsa.lisp`.

Vous y trouvez aussi une variante qui accèptent aussi des automates non-determinés (la fonction `nd-recognize` et ses sous-fonctions).

1. `nd-recognize` ne permet pas encore un automate à transitions vides, c'est-à-dire des transitions d'un état à un autre sans lire un mot.

   Dans un automate, une transition vide va être symbolisée par `vide` comme catégorie de cette transition.

   Modifiez la fonction `nd-recognize-transitions` tel qu'elle fonctionne aussi pour des transitions vides.

   Testez `nd-recognize` avec la nouvelle variante de `nd-recognize-transitions` avec l'automate `new-french-np` suivant (qui fait toujours la même chose que l'ancien `french-np`).

   ```
   (setq new-french-np
         '(1 (4)
           (1 det 2)
           (2 adj 2)
           (2 vide 3)
           (3 noun 4)))
   ```

   Exemple:

   ```
   > (nd-recognize new-french-np '(le grand chien))
   T
   > (nd-recognize new-french-np '(un chat))
   T
   > (nd-recognize new-french-np '(un chat joli))
   NIL
   ```

2. Finite state transducers: Pour obtenir pas seulement un `T` ou `NIL` quand on teste si un automate accèpte une phrase, mais pour obtenir aussi dans le cas positif une analyse de la phrase, on peut ajouter un symbole de plus à chaque transition, un symbole d'édition (output). Un tel automate s'appelle un finite state transducer (FST).

   Le FST correspondant à notre automate `french-np` pourrait être comme suit:

   ```
   (setq fr-np-fst
         '(1 (4)
           (1 det 2 det)
           (2 adj 2 a)
           (2 vide 3 vide)
           (3 noun 4 n)))
   ```

   Définissez une variante `parse-fst` de `nd-recognize` qui prend un FST et une phrase (une liste de mots) comme arguments et qui retourne l'analyse de la phrase si la phrase est acceptée par le transducer. Sinon, la fonction rend `NIL`.

Exemple:

```
> (parse-fst fr-np-fst '(un joli grand chat))
(DET A A N)
> (parse-fst fr-np-fst '(le grand))
NIL
```

3. On peut redéfinir `preterminals` en distinguant entre singulier/pluriel et feminine/masculin:

```
(setq preterminals
      '((detsm un le) (detsf une la)
        (detp des les)
        (adjsm grand joli) (adjsf grande jolie)
        (adjpm grands jolis) (adjpf grandes jolies)
        (nounsm chien chat) (nounsf pomme poire)
        (nounpm chiens chats) (nounpf pommes poires))
```

Redéfinissez le FST `fr-np-fst` tellement qu'il accèpte `(les jolies poires)`, `(une grande pomme)`, etc. en donnant comme analyse toujours `(DET A* N)`:

```
> (parse-fst fr-np-fst '(la jolie grande poire))
(DET A A N)
> (parse-fst fr-np-fst '(la grande))
NIL
> (parse-fst fr-np-fst '(une poire))
(DET N)
> (parse-fst fr-np-fst '(les grandes pommes))
(DET A N)
```