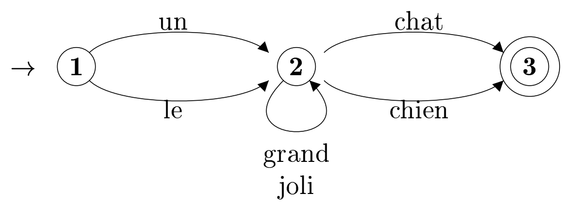
Programmation fonctionnelle

Laura Kallmeyer

TD du 18.10.00

1. Pour l'automate



on peut introduire des préterminaux:

```
(setq preterminals
      '((det un le) (adj grand joli) (noun chien chat)))
```

L'automate peut être défini en utilisant des préterminaux:

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

Définissez un prédicat `is-preterminal-of` pour tester si une certaine catégorie préterminale est la catégorie d'un certain mot. (On peut utilisez `member` et `assoc`.)

```
> (is-preterminal-of 'det 'un)
(UN LE)
> (is-preterminal-of 'det 'la)
NIL
```

2. Au lieu de définir une fonction spécial pour un seul automate comme dans le cours, définissez une fonction `recognize` qui prend un automate et une liste et qui retourne `T` si la séquence representée par la liste est acceptée par l'automate. Sinon, elle retourne `NIL`.

```
> (recognize french-np '(un grand grand chien))
T
> (recognize french-np '(pas de syntagme nominal))
NIL
```

Il suffit que la fonction marche pour les automates finis déterminés (c'est-à-dire pour les automates où il n'y a qu'un seul nouvel état pour chaque paire d'ancien état et de mot prochain).

On peut définir `recognize` tel que `recognize` appelle une deuxième fonction `recognize-step` avec l'état initial de l'automate, l'automate et la liste des mots. En plus on peut définir une fonction `recognize-transition` qui cherche la transition qui doit être choisie dans un certain état avec un certain mot prochain.

On peut aussi se servir des fonctions pour extraire l'état initial, la liste des états finaux et les transitions d'un automate comme définies dans le cours:

```
(defun initial-state (fsa) (car fsa))

(defun final-states (fsa) (cadr fsa))

(defun transitions (fsa) (cddr fsa))
```