(Séquence 6.2

Liste d'associations

Les listes d'associations

- Association
- Liste d'associations
- (constructeur) Ajout d'une association
- (sélecteur) Recherche d'une association
- (sélecteur) Recherche d'une valeur

Une association

Définition : étant donnés deux types ***Clef*** et ***Valeur***, une ***association*** est un élément de type `COUPLE[Clef Valeur]`, représentée par une liste de 2 éléments.

```
; association de type COUPLE[string string]
        ("chat" "cat")
; associations de type COUPLE[Nat Nat]
        (2003512    192)
        (2003513    567)
; associations de type COUPLE[Nat string]
        (192   "Virginie")
        (567   "Paul")
```

Autre exemple d'association

Seulement pour ceux qui désirent approfondir la notion de citation.

```
; associations de type COUPLE[symbole fonction]
          (list  '+    +)
          (list  '*    *)
          (list  '^    puissance)
```

```
(list '+ +) →(+ #<primitive:+>)
```

```
'('+ +) → ('+ +) ou ((quote +) +)
```

Une liste d'association

Définition : ***une liste d'associations*** est une liste dont chaque terme est une **association (clef valeur)**

```
; LISTE[COUPLE[string string]]
(list (list "chat"  "cat")
      (list "chien" "dog"))
   ≡
(list '("chat"  "cat")
      '("chien" "dog"))
  ≡
'(("chat"  "cat")
  ("chien" "dog"))
            →  (("chat" "cat")("chien" "dog"))
```

Ajout dans une liste d'associations

Le constructeur de la liste d'associations vide est `(list)`
Ce constructeur ajoute une association en tête d'une ***Aliste***

```
;;; ajout : alpha * beta * LISTE[COUPLE[alpha beta]]
;;;                 -> LISTE[COUPLE[alpha beta]]
;;; (ajout cle valeur a-liste) rend la liste d'asso-
;;; -ciation obtenue en ajoutant l'association
;;; (cle valeur) en tete de a-liste.
(define (ajout cle valeur a-liste)
  (cons (list cle valeur)
        a-liste) )
```

```
(let ((mon-dico '(("souris"  "mouse")
                  ("chat"    "cat")
                  ("chien"   "dog")
                  ("fromage" "cheese"))))
  (ajout "chat" "tabby" mon-dico))
```

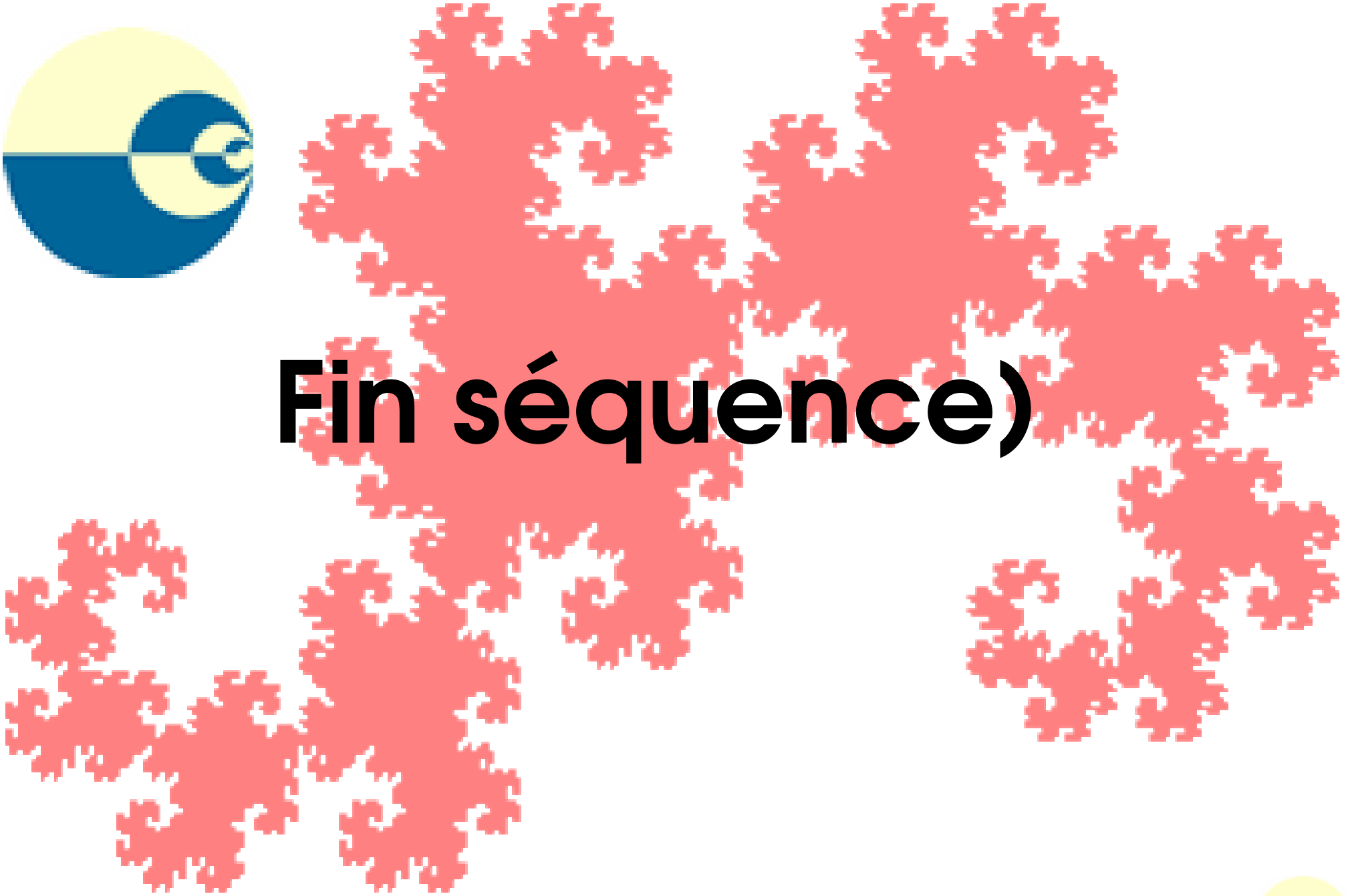
Recherche

(cf. carte de référence)

```
;;; assoc:
;;;   alpha * LISTE[COUPLE[a b]] -> COUPLE[a b] + #f
;;; (assoc cle aliste) rend la 1ère association de
;;; aliste dont le 1er élément est égal à cle. Rend
;;; la valeur #f en cas d'échec.
```

- Rend la première association de la liste (raison d'efficacité) et donc la plus récente (et la plus à gauche)
- C'est un ***semi-prédicat***, elle rend :
  - `#f` lorsque l'association n'existe pas,
  - et sinon la valeur `Vrai` sous la forme de l'association elle-même.

Rappelons qu'en Scheme tout ce qui n'est pas `#f` est Vrai. Une association vaut donc toujours vrai.



Fin séquence)