

## 11 Types structurés

**Type structuré** : type dont les éléments sont des agrégats de données.

### 11.1 Types structurés primitifs

- **Paire** : agrégat de deux données accessibles par leur position (première ou seconde).
- **Tableau** : agrégat d'un nombre quelconque prédéfini de données généralement homogènes (tableau d'entiers) accessibles via un index.
- **Enregistrement (record)** : agrégat de données généralement hétérogènes accessibles via un nom. Exemple, une personne est un agrégat d'un nom, d'un prénom (chaîne), d'un âge (int), etc.
- **Sac** : collection quelconque non ordonnée
- **Ensemble** : collection quelconque sans ordre ni répétition.
- **Séquence - Liste** : collection ordonnée soit par l'ordre d'insertion par une relation entre les éléments.

### 11.2 Les paires (doublets) en Scheme

**Pair (paire)** ou **Cons** est le type structuré de base.

Une paire, également dit “cons” (prononcer “conce”) ou doublet est un agrégat de deux données.

Exemple d'utilisation : représentation d'un nombre rationnel, aggrégation d'un numérateur d'un dénominateur.

Notation : “(el1 . el2)”.

**Fonctions de construction :**

1. **cons**

```
(cons 1 2)  
= (1 . 2)
```

```
(cons (+ 1 2) (+ 2 3))  
= (3 . 5)
```

**Fonctions d'accès :**

1. **car** accès au premier élément.

```
(car (cons 1 2))  
= 1
```

2. accès au second élément : **cdr**

```
(cdr (cons 1 2))  
= 2
```

## 12 Listes

Une liste est une collection de données ordonnée par l'ordre de construction (ordre d'insertion des éléments).

Une liste est une **structure de donnée récursive** : une liste est soit la liste vide soit un doublet dont le second élément est une liste.

Les fonctions de construction et de manipulation de base sont les mêmes que celles des doublets.

```
(cons 1 (cons 2 (cons 3 ())))  
= (1 . (2 . (3 . ())))
```

Notation simplifiée : (1 2 3).

- **car** rend le premier élément du premier doublet constituant la liste donc le premier élément de la liste.
- **cdr** rend le second élément du premier doublet c'est à dire la liste privée de son premier élément.
- **list** est une fonction n-aire réalisant n "cons" successifs.

```
(cons 1 ())  
= (1)  
(cons 1 (cons 2 (cons 3 ())))  
= (1 2 3)  
(define l2 (cons 4 (cons 6 ())))  
(cons 2 l2)  
= (2 4 6)  
(cons l2 l2)  
= (((4 6) 4 6)  
(list (+ 2 3) 4 6 (fact 5))  
= (5 4 6 120)
```

### 12.1 Calcul symbolique

#### 12.1.1 Symboles

Un programme de gestion d'un magasin doit par exemple gérer, faire des "calculs", sur des personnes, des marques, des collection de produits. Le calcul symbolique utilise des listes, des chaînes de caractères et des symboles.

**Symbole** : chaîne de caractères alphanumériques unique en mémoire utilisable :

- stricto sensu
- comme identificateur (de fonction ou de variable).

Exemples :

- un symbole évoquant une couleur : **rouge**
- une liste évoquant une jolie maison rouge : (une jolie maison rouge)
- une liste de plein de choses différentes : (jean dupond 20 marié jeanne 2 jeannette jeannot (2 bis rue des feuilles))
- une liste d'associations : ((dupont 10) (durand 16) (martin 12))
- une autre liste d'associations ((Hugo (ecrivain francais 19ieme)) (Moliere (...)) ...)
- une liste représentant la définition d'une fonction scheme : (define (fact n) ...)
- un symbole représentant une fonction scheme **fact**

### 12.1.2 guillemets, citations

En français mettre un texte ou un mot entre guillemets permet de faire une citation. Les guillemets empêchent le texte qu'ils délimitent d'être interprété de façon standard.

```
dis moi quel est ton age.  
dis moi : ‘‘quel est ton age’’!.
```

En programmation, les guillemets sont utiles dès lors qu'il y a ambiguïté possible dans l'interprétation d'une expression,

En Scheme, un appel de fonction se présente sous forme d'une liste mais toutes les listes présentes dans le texte d'un programme ne sont pas des appels de fonction.

Pour signifier à l'interpréteur qu'une expression parenthésée n'est pas un appel de fonction, il faut la mettre entre guillemets.

De même en Scheme, un identificateur est un symbole mais tous les symboles présents dans le texte d'un programme scheme ne sont pas des identificateurs. Pour indiquer à l'interpréteur qu'un symbole n'est pas un identificateur, il faut le mettre entre guillemets.

La structure de controle scheme permettant de mettre une expression entre guillemets est “*quote*”.

Soit  $e$  une expression de la forme “(quote exp)”, on a  $val(e) = exp$ .

```
> (+ 2 3)  
= 5  
> (quote (+ 2 3))  
= (+ 2 3)  
> '(+ 2 3) //racourci syntaxique  
= (+ 2 3)  
  
> (list (quote un) (quote joli) (quote chien))  
= (un joli chien)  
> (define un 1)  
> (define deux 2)  
> (list (quote un) deux)  
= (un 2)
```

## 12.2 Fonctions de manipulation de listes

- tests : null?, list?, cons?, pair? member?
- renversement, concaténation.  
(reverse '(1 2 3))  
= (3 2 1)  
(append '(1 2) '(3 4))  
= (1 2 3 4)
- *reverse* d'une liste de  $n$  éléments consomme  $n$  nouveaux doublets.
- *append* de  $l1$  de taille  $n$  et de  $l2$  de taille  $m$  consomme  $n$  nouveaux doublets.

## 12.3 Egalité physique et logique

Egalité physique : eq?

Egalité logique : `equal` ?

(list 'a 'b) est égal à (list 'a 'b) dit la fonction `equal` ?

(list 'a 'b) n'est pas égal à (list 'a 'b) dit la fonction `eq` ?.

Extension au problème de l'appartenance à une liste : `member` ? versus `memq` ?

## 12.4 Fonctionnelles - Itérateurs

Appliquer une fonction (n'importe quelle fonction) à tous les éléments d'une liste.

```
> (map number? '(1 'abc' (3 4) 5))  
= (#t #f #f #t)
```

```
(map fact '(0 1 2 3 4 5))  
= (1 1 2 6 14 120)
```

Il n'est pas indispensable de nommer une fonction pour la passer en argument à une fonctionnelle.

```
(map (lambda (x) x) '(a b c))  
= (a b c)
```

## 12.5 Listes généralisées

Liste généralisée ou liste non plate ou arbre : liste dont chaque élément peut être une liste.

Exemple : liste d'associations.

```
(define unDico '((hugo ecrivain) (pasteur médecin) (knuth  
algorithmicien)))
```

```
(assq 'pasteur unDico)  
= médecin
```

Arbres, voir chapitres suivants.

Le doublet (1 . 2)  $\longrightarrow$ 

1
---

