Cours 7 - Programmation Lisp (III)

Marie-Hélène Abel

Table des matières

Structure interne des listes en machine

I

A. Structure interne des listes en machine et table des symboles

Après quelques rappels nous précisons le mode de stockage des listes en Lisp.

1. Rappels

Rappel : Représentation interne d'une liste

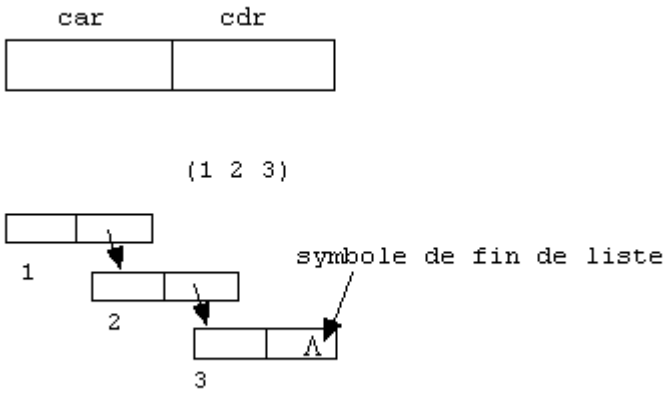

image1

Méthode

- Lorsque l'utilisateur donne une liste à Lisp, le programme de lecture construit la structure correspondante.
- Il passe ensuite le pointeur sur cette structure à l'évaluateur, qui traite la liste en fonction de son contenu.
- Le résultat (une structure interne) est traduit sous forme d'une chaîne de caractères qui est imprimée par le programme d'impression.

Exemple

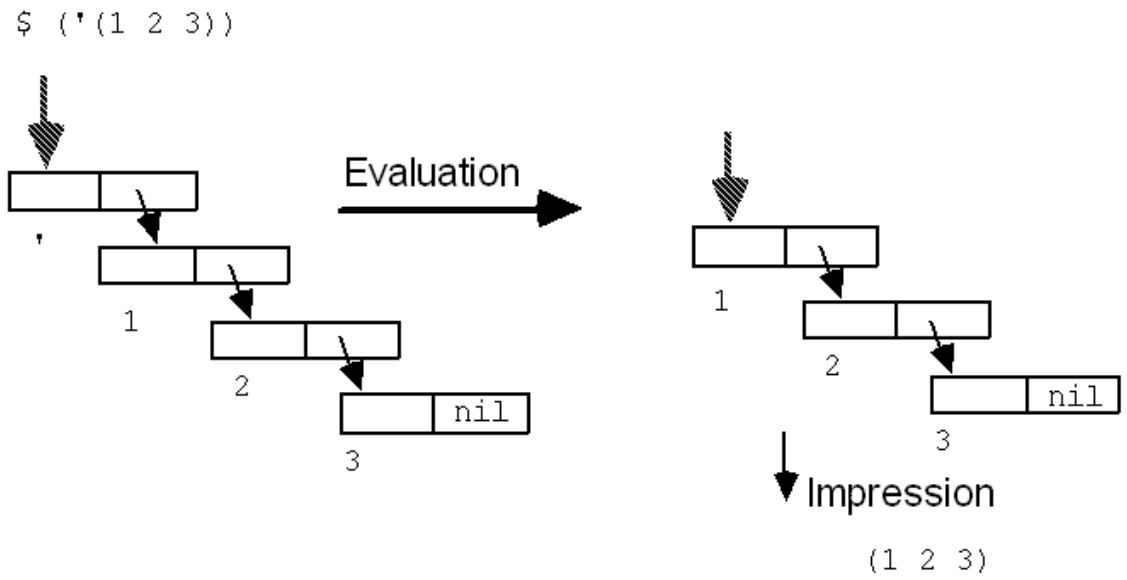

Image 2

2. Stockage des listes

Où se trouvent les têtes de listes, quand celles-ci sont conservées comme valeurs de symboles ?

En pratique, la mémoire utilisée par l'environnement Lisp est divisée en un certain nombre de zones :

- Une contenant le code des primitives,
- Une contenant les structures définies par l'utilisateur,
- Une contenant la pile d'appels de fonction,
- Une contenant les symboles : la table des symboles,

...

La table des symboles

La table des symboles sert à retrouver toutes les structures définies par le programmeur.

Remarque

Lisp, comme tout langage interprété, conserve le nom des variables au moment de l'exécution (contrairement aux langages compilés qui remplacent le nom des variables par une adresse mémoire au moment de la compilation).

Remarque

Les variables ou symboles ou atomes ont donc une structure interne associée, conservée dans la table des symboles.

Exemple

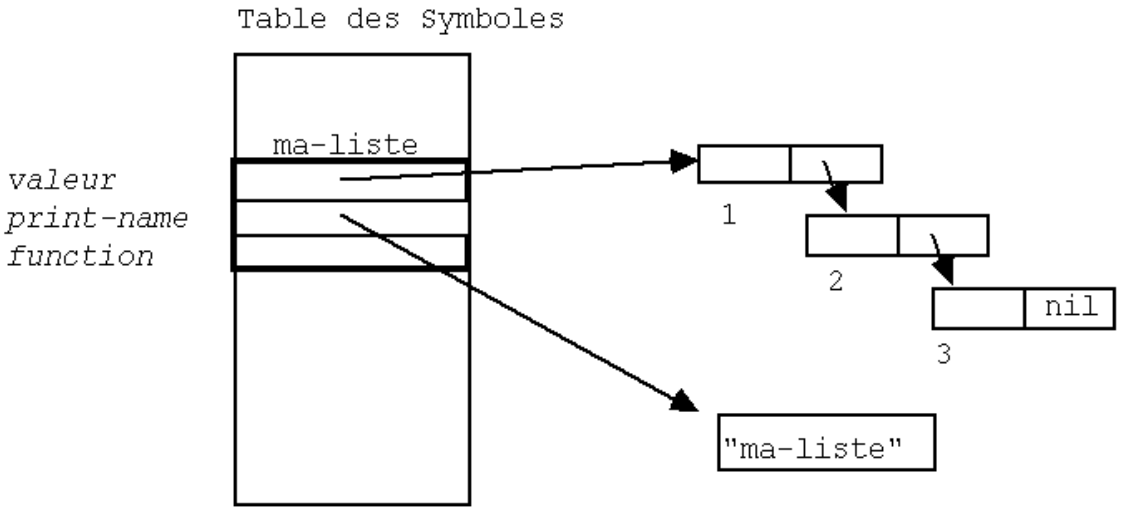

image3

Attention

Une liste de symboles est essentiellement une structure de pointeurs. Le système économise la place qui correspondrait aux chaînes de caractères en ne représentant qu'une seule fois les informations en mémoire.

Exemple : (setq objet '(chaise couleur rouge))

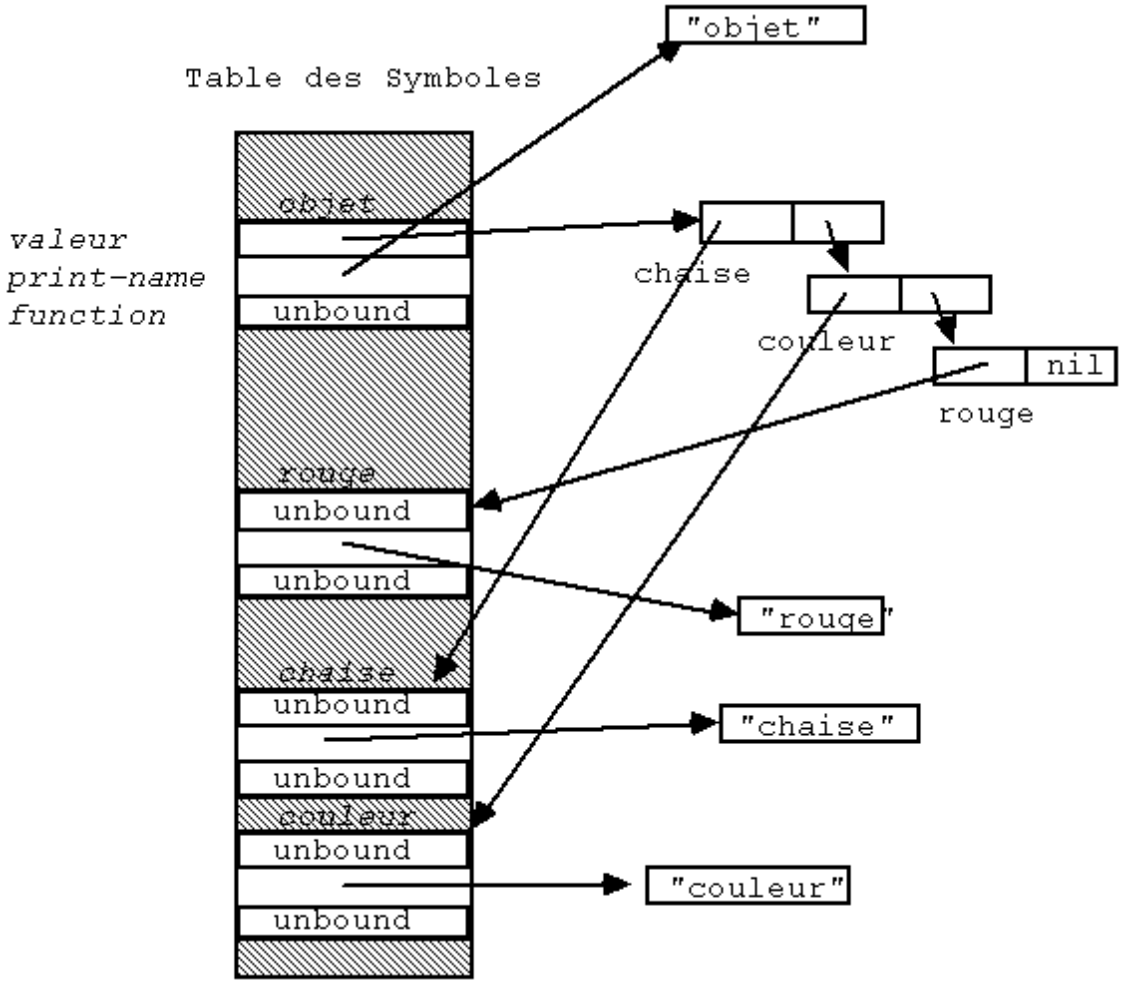

image4

Exemple : (setq objet-2 '(chaise couleur bleu))

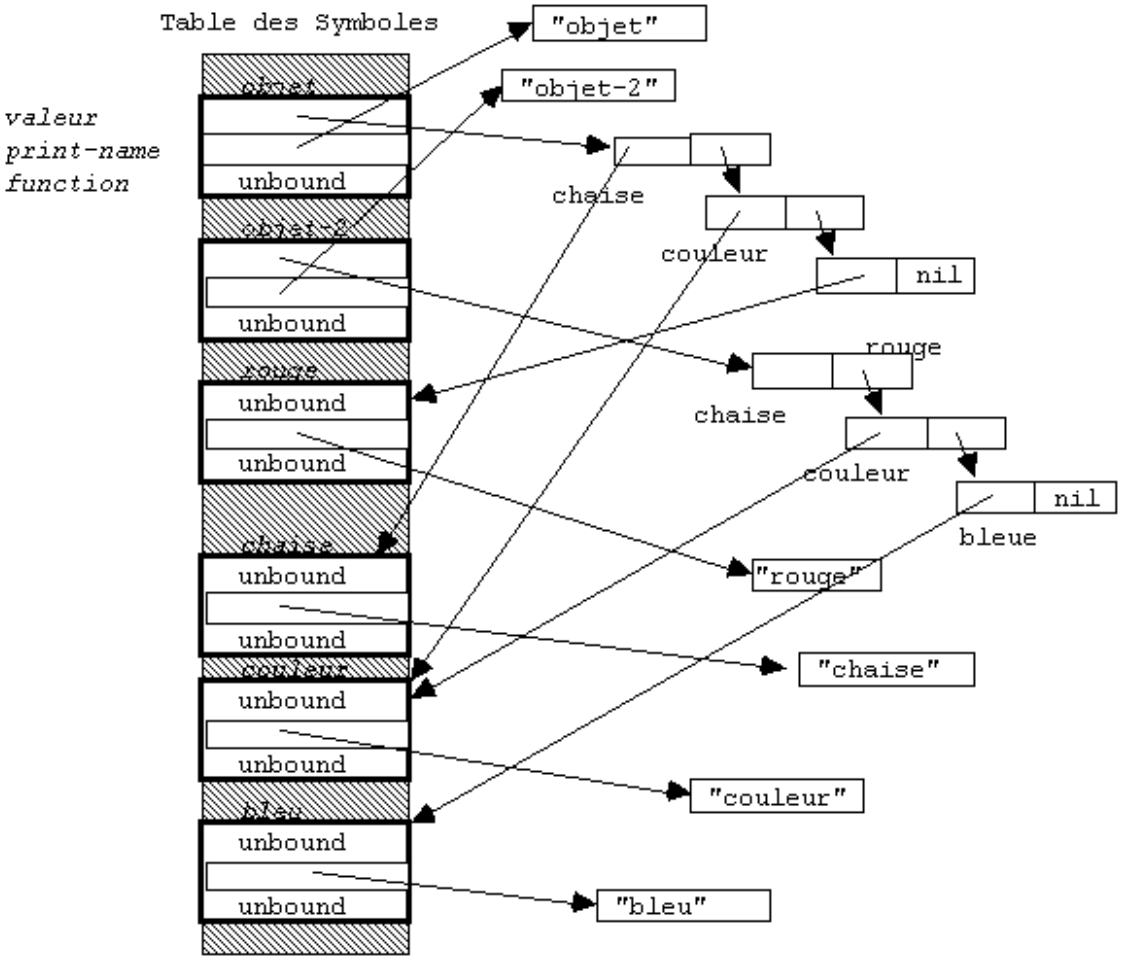

image5

Remarque

La création d'une nouvelle liste n'a entraîné que la création de deux symboles supplémentaires : **objet-2** et **bleu**

Exemple : (setq objet-2 (cdr objet))

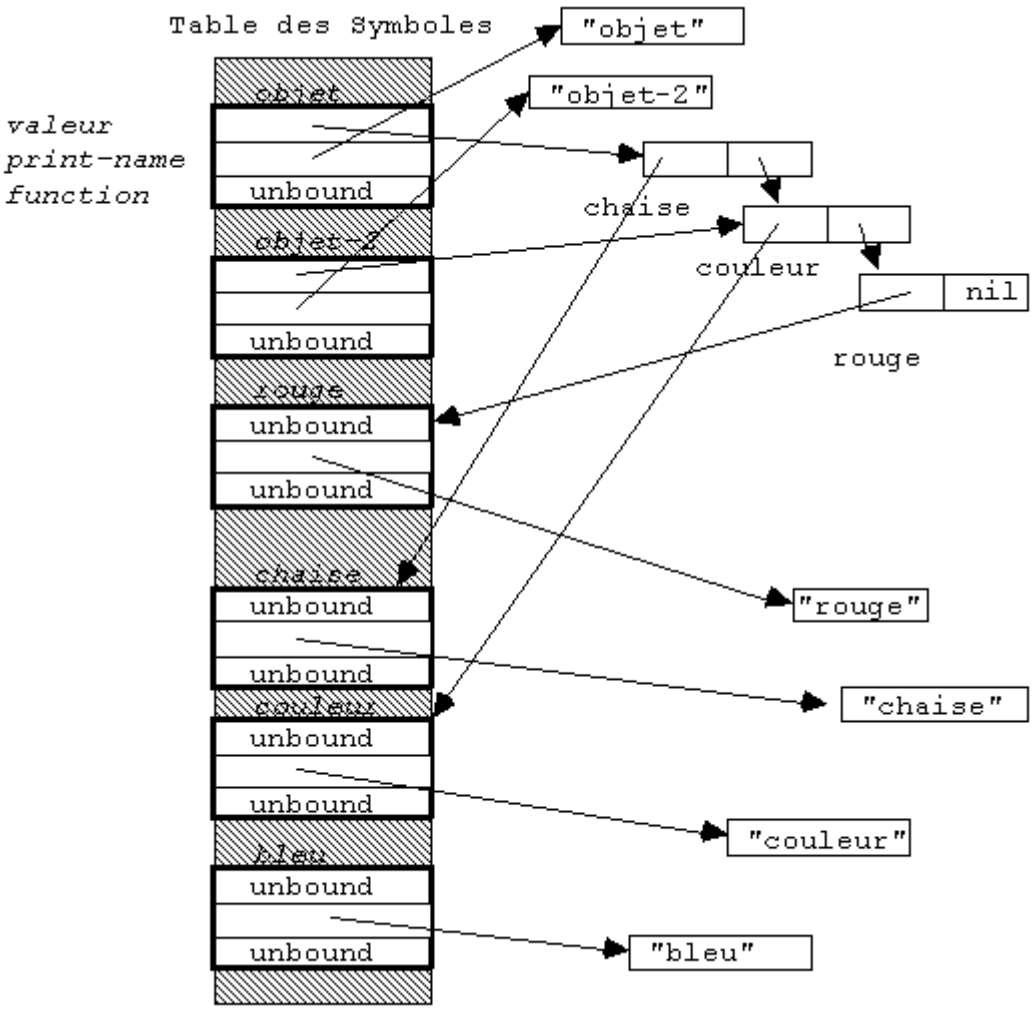

image6

Remarque

La sous-liste (couleur rouge) est partagée par les objets **objet** et **objet-2**.

Attention

L'utilisation de primitives destructrices de structures peut être dangereuse.

Remarque

Le symbole **bleu** est isolé, il reste toutefois présent dans la table des symboles (Lisp utilise dans ce cas le mécanisme de garbage collector)

B. Structure interne des listes en machine et paire pointée

Remarque

La représentation interne permet de construire des structures qui ne correspondent pas exactement à des listes.

1. Paires pointées

Définition : Paire pointée

Cette structure est une cellule dont les deux éléments pointent sur des atomes :

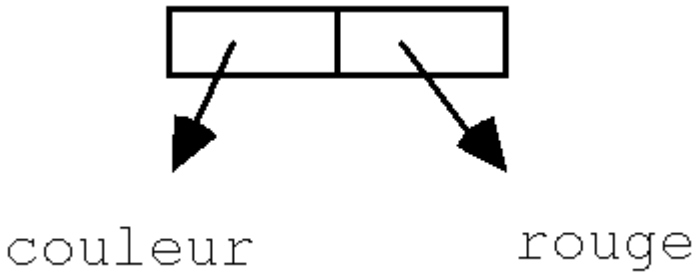

image7

Exemple

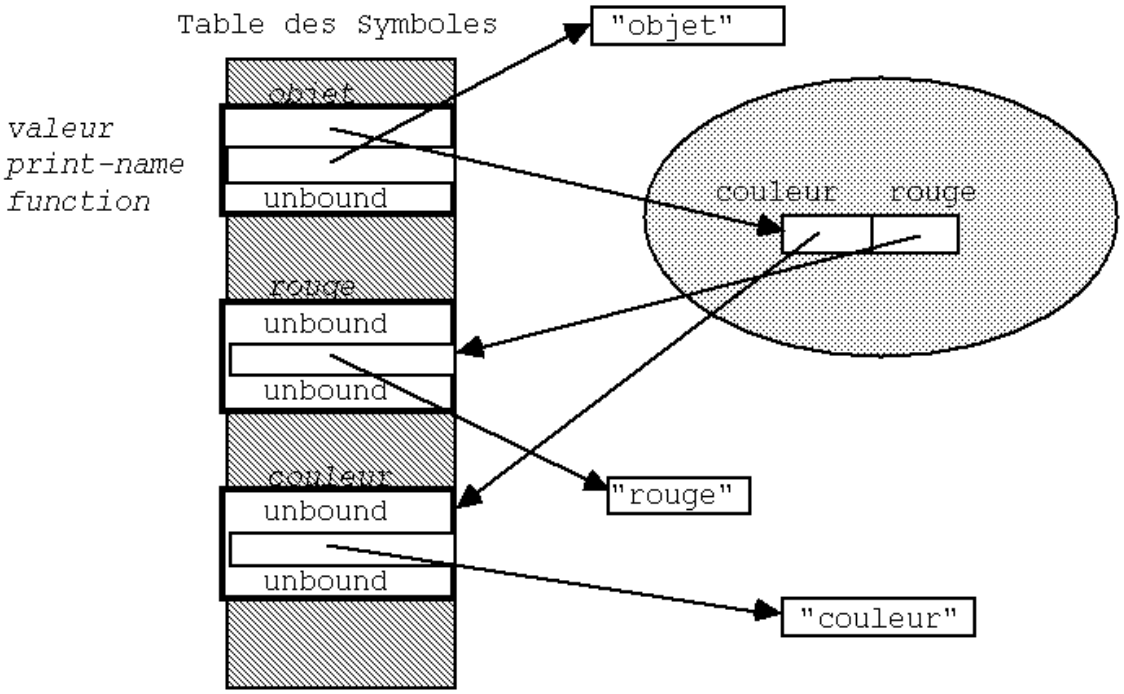

image8

2. Notation

Exemple

(Couleur . Rouge)
(A B . C)

Attention

Certaines expressions ne correspondent pas à des structures physiques possibles :
(A . B C D)

Remarque : Notation multiple

Une liste classique peut être notée de façon multiple en introduisant la notation pointée.

Exemple : (1 2 3)

- (1 2 3)
- (1 2 3 . NIL)
- (1 2 . (3 . NIL))
- (1 . (2 . (3 . NIL)))

3. Listes d'association : a-listes

Définition

Une liste d'association est une liste de paires pointées.

Exemple

```
(setq a-list '((a . 1)(b . 2)(c . 3)))
```

La fonction assoc

Une **a-liste** est une structure qui regroupe clés et données dans une liste.

La fonction **Assoc** permet de retrouver des données en fonction d'une clé.

Exemple

```
> (assoc 'b a-list)
(B . 2)
```

Remarque

Assoc retourne toujours une paire afin de pouvoir faire la distinction entre la non existence de la paire correspondante et le cas ou les données sont NIL.

La forme SETF

II

Remarque

En lisp, il est possible d'affecter une valeur à n'importe quel endroit de la mémoire en utilisant la forme **setf**.

Exemple

```
>(setf l '(a b c d))
(a b c d)
>(setf (car (cdr l)) 'e)
e
>
```

Remarque

Setf fonctionne pour n'importe quel chaînage de formes, pourvu qu'il s'agisse d'une forme d'accès à la mémoire.

Propriétés de symboles : PLIST

III

Remarque

A chaque symbole sont associées des propriétés, comme son nom ou sa valeur.
Il est possible de définir de nouvelles propriétés.

Fonction GET

L'accès à ces nouvelles propriétés se fait au moyen de la fonction GET

Méthode

(get <symbol> <propriété>)

Exemple

```
>(setf (get 'foo 'couleur) 'rouge)
ROUGE
>(get 'foo 'couleur)
ROUGE
>(symbol-plist 'foo)
(couleur rouge)
>
```