8. LES P-LISTES

Jusqu'à maintenant nous avons utilisé les *symboles*, c'est-à-dire les *atomes littéraux*, soit comme des *noms*, soit comme des *paramètres* de fonctions utilisateurs. Dans ce deuxième cas, les symboles peuvent avoir des *valeurs* : les valeurs auxquelles ils sont liés à l'appel des fonctions dont ils constituent des paramètres. On appelle la valeur de liaison la *C-valeur* ou la *cell-value*. Le langage LISP a également prévu des mécanismes de bases de données associatives grâce à la *P-liste* ou *liste de propriétés*. C'est ce que nous étudierons dans ce chapitre.

8.1. LES SYMBOLES

Regardons d'abord la représentation interne des symboles. Un symbole se distingue d'un autre principalement par trois caractéristiques : son *nom* (sa représentation externe), sa *valeur*, (la valeur à laquelle il est lié), et sa *P-liste*. La valeur d'un symbole est donnée par la fonction **SYMEVAL**. La P-liste d'un symbole est donnée par la fonction **PLIST**. Ainsi, l'appel

(SYMEVAL 'XYZZY)

calcule la C-valeur du symbole **XYZZY**,[1] et l'appel

(PLIST 'XYZZY)

ramène la P-liste du même symbole. Notons que la fonction **PLIST** peut optionnellement avoir un deuxième argument. Dans ce cas, la liste deuxième argument devient la nouvelle P-liste du symbole premier argument.

Une des particularités de VLISP est d'avoir une symétrie entre l'accès aux listes et l'accès aux symboles : la valeur d'un symbole est, en VLISP, le **CAR** du symbole, la P-liste d'un symbole est son **CDR**. Vous pouvez donc, en VLISP, demander le **CAR** ou le **CDR** d'un symbole, et, si le symbole **XYZZY** est lié à la valeur numérique **100**, les valeurs de l'expression LISP :

(1+ (CAR 'XYZZY))

et de l'expression

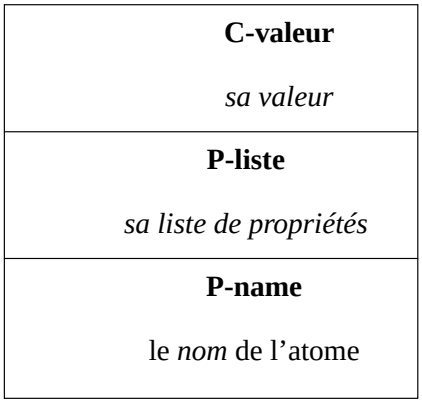
(1+ XYZZY)

sont identiques. Dans les deux cas la valeur sera **101**. De temps à autre, cette caractéristique peut être fort utile.

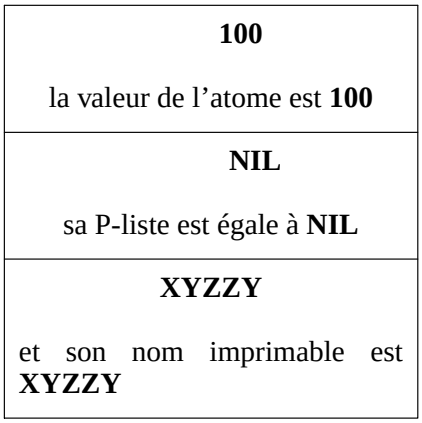
Voici à présent comment un symbole peut être représenté de façon interne :

---

[1] **XYZZY** est un mot magique bien connu des joueurs d'*adventure*. Si vous ne le connaissez pas encore, essayez le, admirez l'effet instantané. Essayez aussi le mot magique **PLUGH**.



Initialement, la C-valeur d'un symbole est la valeur **INDEFINI**, indiquant que le symbole n'est pas lié à une valeur et que toute demande d'acces à sa valeur provoquera une *erreur* : **atome non défini**. La P-liste est initialisée à la valeur **NIL**. Naturellement, le *P-name* - ou *print-name* - sera la suite de caractères donnant le nom de l'atome. Voici la représentation interne du symbole **XYZZY** vu ci-dessus.



8.2. L'ACCES AUX P-LISTES

Très souvent il arrive qu'un symbole devra avoir plus d'une seule valeur. Par exemple, pour implémenter une base de données de relations familiales, il peut être nécessaire de représenter quelque part le fait que **PIERRE** à un père, nommé **JEAN**, une mère, nommée **JULIE**, et un fils de nom **GERARD**. Cela veut dire que la valeur de la propriété *père* de **PIERRE** est **JEAN**, que la valeur de la propriété *mère* de **PIERRE** est **JULIE** et que la valeur de la propriété *fils* de **PIERRE** est **GERARD**. Evidemment, les valeurs associées à **PIERRE** sont multiples et dépendent de la caractéristique particulière qu'on interroge. Pour implémenter de telles structures, LISP met quatre fonctions, agissant sur la P-liste, à votre service : les fonctions **PUT**, **GET**, **REMPROP** et **ADDPROP**.

Notons qu'en LE_LISP, de même que dans quelques autres dialectes de LISP, **PUT** s'appelle **PUTPROP** et **GET** s'appelle **GETPROP**.

Regardons d'abord un exemple d'utilisation de ces fonctions, ensuite leurs définitions. Voici à présent la suite d'instructions LISP pour implémenter les relations familiales de **PIERRE** :

(PUT 'PIERRE 'PERE 'JEAN)

(PUT 'PIERRE 'MERE 'JULIE)

(PUT 'PIERRE 'FILS 'GERARD)

Ces instructions ont comme effet de mettre sur la P-liste du symbole **PIERRE** sous l'indicateur **PERE** la valeur de l'expression **'JEAN** (i.e.: l'atome **JEAN**), sous l'indicateur **MERE** l'atome **JULIE** et sous l'indicateur **FILS** l'atome **GERARD**.

La P-liste d'un symbole est donc une suite d'indicateurs et de valeurs associées à ces indicateurs :

(*indicateur*$_1$ *valeur*$_1$ *indicateur*$_2$ *valeur*$_2$ . . . *indicateur*$_n$ *valeur*$_n$**)**

Ainsi, après les quelques appels de la fonction **PUT** ci-dessus la P-liste du symbole **PIERRE** aura la forme suivante :

(PERE JEAN MERE JULIE FILS GERARD)

Les P-listes peuvent être imaginées comme des tableaux de correspondance qu'on associe aux symboles. Ainsi la P-liste donnée ci-dessus, peut être représentée comme :

PIERRE

indicateur	**MERE**	**FILS**	**PERE**
valeur	**JULIE**	**GERARD**	**JEAN**

Pour connaître la valeur associée à un indicateur on utilise la fonction **GET**. Ainsi pour connaître le père de Pierre, il suffit de demander :

(GET 'PIERRE 'PERE)

ce qui vous ramène en valeur l'atome **JEAN**.

Voici la définition de la fonction **PUT** :

(PUT *symbole indicateur valeur***)** $\rightarrow$ *symbole*

ce qui a comme effet :

> **PUT** met sur la P-liste su symbole *symbole*, sous l'indicateur *indicateur*, la valeur *valeur*. Si l'indicateur *indicateur* existe déjà, la valeur associée préalablement est perdue. *Tous* les arguments sont évalués (c'est pourquoi nous les avons **quotés**). La valeur ramenée par la fonction **PUT** est la valeur de son premier argument, donc de *symbole*.

En **MACLISP**, le dialecte LISP tournant - entre autre - sur la machine Multics, la fonction **PUTPROP** ramène *valeur* en valeur. En LE_LISP les arguments de **PUTPROP** sont dans l'ordre :

(PUTPROP *symbole valeur indicateur***)**.

Notez également que dans quelques versions de LISP, la fonction **PUT** admet un nombre quelconque d'arguments et est défini comme suit :

(PUT *symbole indicateur*$_1$ *valeur*$_1$ . . . *indicateur*$_n$ *valeur*$_n$**)** $\rightarrow$ *symbole*

Voici alors la définition de la fonction **GET** :

(GET *symbole indicateur***)** $\rightarrow$ *valeur*

GET ramène la valeur associée à l'indicateur *indicateur* sur la P-liste du symbole *symbole*. Si l'indicateur ne se trouve pas sur la P-liste du *symbole*, **GET** ramène la valeur **NIL**. *Il n'y a donc pas de possibilité de distinguer la valeur* **NIL** *associée à un indicateur de l'absence de cet indicateur*.

La fonction **REMPROP** enlève une propriété de la P-liste. Ainsi, après l'appel :

(REMPROP 'PIERRE 'FILS)

le couple **FILS - GERARD** sera enlevé de la P-liste de l'atome **PIERRE**, et l'instruction

(GET 'PIERRE 'FILS)

suivante ramènera la valeur **NIL**, indiquant l'absence de l'indicateur **FILS**.

(REMPROP *symbole indicateur***)** $\rightarrow$ *symbole*

REMPROP enlève de la P-liste du symbole *symbole*, l'indicateur *indicateur*, ainsi que la valeur associée. **REMPROP** ramène le symbole donné en premier argument.

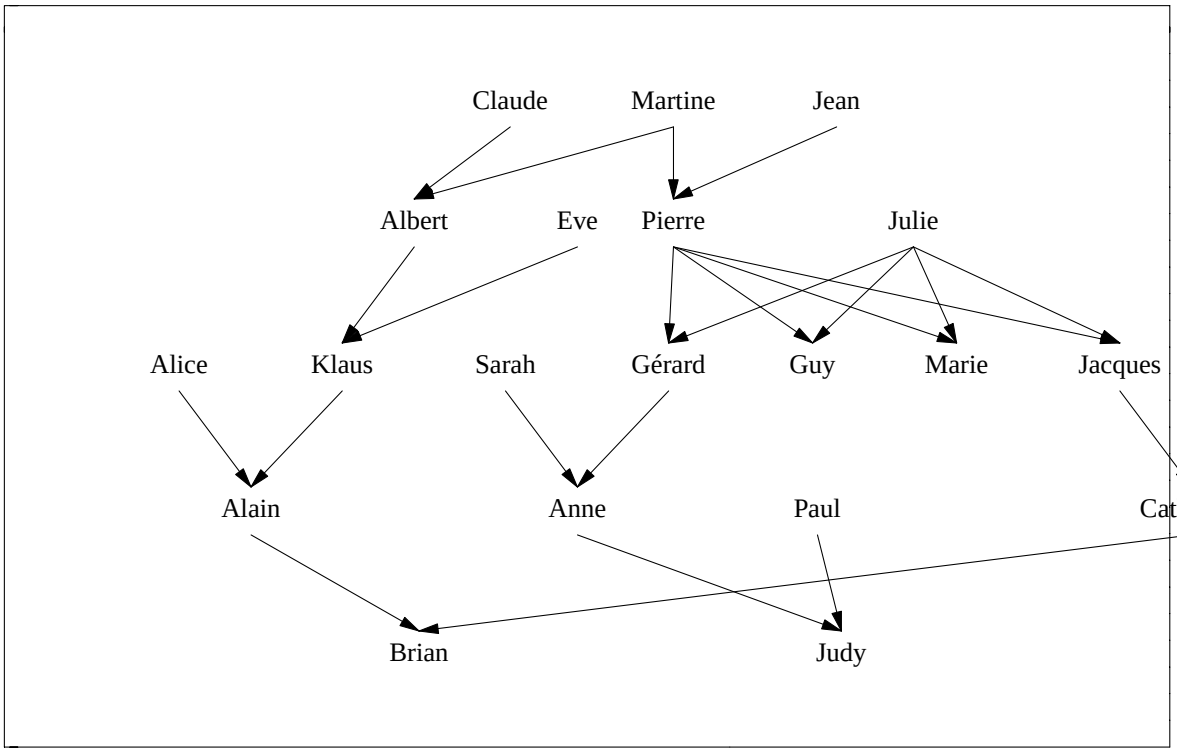
8.3. EXERCICES

1. Traduisez en LISP, à l'aide des fonctions **PUT**, **GET** et **REMPROP**, les phrases suivantes :

a. Pierre est le père de Guy.
b. Pierre a aussi un enfant nommé Marie.
c. Pierre a également un enfant nommé Jacques.
d. Pierre est de sexe masculin.
e. Marie est de sexe feminin.
f. Guy et Jacques sont de sexe masculin.
g. Judy est de même sexe que Marie.
h. Judy a 22 ans.
i. Anne a 40 ans.
j. Sarah a le même age que la somme de l'age de Judy et Anne.

2. (projet) Supposez que chaque livre d'une bibliothèque soit représenté par un atome et que la bibliothèque entière soit représentée par une liste contenant ces atomes. Supposez également que chaque livre a une propriété nommée *titre*, une nommée *auteur* et une nommée *mots-clefs*. Construisez alors une petite base de données bibliographiques, avec des fonctions d'interrogation de cette base de données respectives à ces trois propriétés. Naturellement, si pour une requête plusieurs réponses existent, la fonction d'interrogation doit ramener une liste de toutes les réponses possibles. Ainsi, si vous demandez le/les livres de Karl Kraus, par exemple, et que vous avez dans la bibliothèque les deux titres *Die letzten Tage der Menschheit* et *Von Pest und Presse*, la fonction doit ramener la liste **((DIE LETZTEN TAGE DER MENSCHHEIT)(VON PEST UND PRESSE))**. (Pour avoir une idée de tous les problèmes associés à de telles bases de données, regardez donc un livre sur la *documentation automatique*).

3. Ci-dessous un petit arbre généalogique, dont les flèches vont des parents vers leurs enfants. Représentez les relations familiales simples (père et mère) et le sexe sur les P-listes des atomes des noms des differentes personnes. Ensuite, écrivez un programme qui y trouve des relations non directement représentées, telles que les relations frères, soe;urs, fils, filles, oncles, tantes, cousins et cousines, grand-pères et grand-mères.



Définissez également une fonction **ANCETRES**, qui vous ramène la liste de tous les ancêtres connus d'une personne.

chapitre précédent

solutions

chapitre suivant