PRÉSERVER
SAUVEGARDER
VALORISER

MANUEL : LISP A GENTLE INTRODUCTION TO SYMBOLIC COMPUTATION

FICHE N° 6570

Période de fabrication : 1975-1999
Fabricant : fabricant non renseigné
Domaines : Informatique et Communication
Sous-domaines : Logiciel
Organisme : ACONIT
Ville : Grenoble (Isère)
Modèle :
Matériaux : Carton, Papier

Description

Le document "LISP a gentle introduction to symbolic computation" est inclus dans un coffret ExperLisp, une boite avec les 4 autres documents, fournissant un logiciel de "Système Expert" (édité pour les produits d'Apple par la société "ExperTelligence" situé à Santa Barbara) pour être utilisé sur les ordinateurs personnels Apple Macintosh, avec deux disquettes 3". Il est basé sur la version 1,5 du langage LISP qui date de 1962.
On y trouve une introduction aux usages de LISP, en particulier pour les applications non-mathématiques. On trouve aussi dans cet ouvrage les particularités qui confèrent à LISP son qualificatif de "symbolique" ou "orientation objet" :
- la notion de "prédicat" (page 15),
- les cascades de fonctions (page 25),
- les objets (page 29),
- les listes qui sont des types de données particulièrement importantes dans LISP,
- le système d'adressage des données de LISP par ses deux parties CAR et CDR (pages 39),
- la construction des données avec CONS (page 47),
- la fonction EVAL (page 67), les variables (page 119),
- les "applicative operator" qui sont une fonction prenant une autre fonction comme donnée d'entrée (page 165),
- la "récursion" dans les arborescences (page 191),
- les "itération" (page 243).

L'ouvrage "LISP, langage d'un autre type" (fiche Aconit AC 17847) constitue un bon complément à ce document.

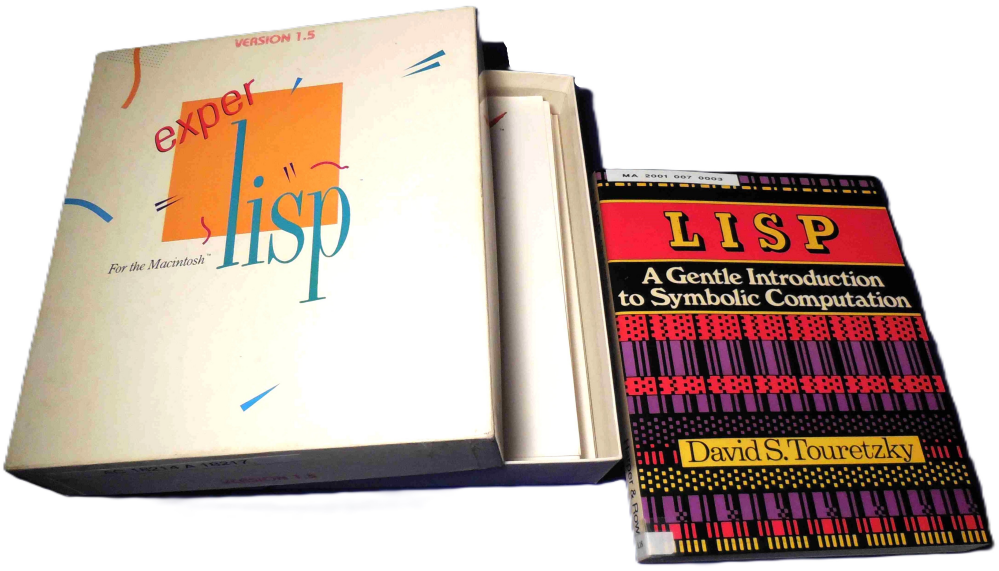
Utilisation

Ce document est destiné à faire comprendre les mécanismes de fonctionnement du langage LISP qui est utilisé par le logiciel ExpertLisp pour le développement de systèmes experts. A cette fin, en complément de la description des mécanismes de LISP, il propose des exercices de prise en main pour chaque élément du langage.
En 1985, avec 25 ans d'existence, LISP est l'un des plus anciens langage de programmation, il a pris une place centrale pour les applications d'intelligence artificielle. Cependant, encore à cette époque, il n'a pas encore été standardisé et est toujours en pleine évolution. Lisp a d'abord été un langage de "traitement de listes de caractères" (LISt Processor), des listes qui sont encore le type principal des données à traiter (voir page 35) et dont chaque élément dans la mémoire de l'ordinateur correspond à une adresse mémoire, laquelle donne accès à un élément de donnée élémentaire (souvent un mot de plusieurs caractères) et à laquelle est mis en complément l'adresse de l'élément suivant dans la liste. Les fonctions de traitement des listes de mots permettent le traitement de phrases (une phrase est une liste de mots) du langage, d'où l'usage de LISP pour la création d'automatismes en "Communication Parlée". Par ailleurs, le caractère "symbolique" correspond à attribuer à un mot (une chaîne de caractères) la fonction d'un objet ayant des propriétés particulières (voir les "Property Lists").

LISP

A Gentle Introduction to Symbolic Computation

David S. Touretzky



Preface

This book is about learning to program in Lisp. About twenty-five years old, Lisp is one of the oldest computer languages, yet its importance and popularity are increasing today as never before. One reason Lisp is so important is that it is the *lingua franca* of artificial intelligence research, or "AI." AI is moving from the laboratory to the everyday world, and tremendous changes are in store for our society as a result. Lisp's popularity is also growing because it offers a friendly, interactive programming environment, an invaluable aid to beginning programmers.

When I wrote the book I had three types of readers in mind, and I would like to address each in turn:

- The humanities student taking his or her first programming course. For you, let me stress the word *gentle* in the title. I assume no prior mathematical background beyond arithmetic. Even if you don't like math, this book may teach you to enjoy computer programming. I've avoided technical jargon. There are many examples. Plenty of exercises are interspersed with the text, the answers to which may be found in appendix D.
- Psychologists, linguists, and other persons interested in AI and cognitive science. As you begin your inquiry into artificial intelligence, you will see that almost all research in this field is carried out in Lisp. This book can be your doorway to a deeper understanding of the technical literature as well as a quick introduction to the central tool of AI.
- Computer hobbyists. Several high-quality mini-Lisp systems are available today for personal computers, such as the Apple and many Z-80 based machines. These systems are adequate for learning Lisp and writing simple programs, but currently their power is limited by a lack of memory. When more powerful personal computers become affordable—for example, those with 24- or 32-bit address spaces and virtual memory capability—hobbyists will be able to run the same Lisp software that is used in today's AI research labs. The prospect of putting research-quality Lisp systems into the hands of computer hobbyists is truly exciting. This book will teach you to use the Lisp systems available today, but more important it will help you prepare for the technology that is coming tomorrow.

There has never been a standard Lisp. Throughout its history the language has been evolving and gradually maturing; the end of this process is not yet in sight. Consequently there are a large number of Lisp dialects around. Lisp 1.5 is the closest thing there is to a "standard" Lisp, but it's a 1962 dialect that lacks much of the power of modern Lisp. The major dialects today are MacLisp, Interlisp, UCI Lisp, and Lisp Machine Lisp; Franz Lisp and Common Lisp are MacLisp offshoots that are developing rapidly and will soon be important in their own right. This book is based primarily on MacLisp and Common Lisp, with a few borrowings from other dialects. Appendix C gives information on how to make virtually any Lisp system compatible with the one used in the book.

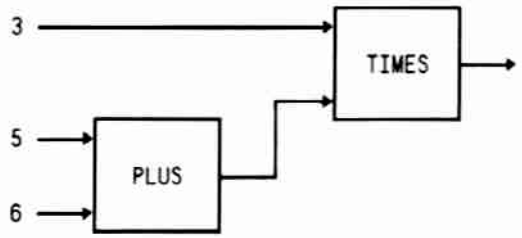
3. LENGTH OF LISTS

The length of a list is the number of elements it has. Length is independent of the complexity of the elements. The following lists all have exactly three elements, even though in some cases the elements are themselves lists. The three elements are shown by underlining.

```
(ALPHA BRAVO CHARLIE)

((A A) (B B) (C C))

(A (B X Y Z) C)

(FOO 937 (GLEEP GLORP))

(((ROY)) ((TWO WHITE DUCKS)) ((MELTED BUTTER)))
```

4. EVAL NOTATION CAN DO ANYTHING BOX NOTATION CAN DO

It should be obvious that any expression we write in box notation can also be written in EVAL notation. The expression



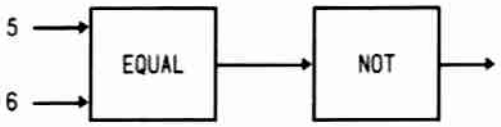
can be represented in EVAL notation as

```
(TIMES 3 (PLUS 5 6))
```

Conversely, the EVAL notation expression

```
(NOT (EQUAL 5 6))
```

is represented in box notation as



Not only can EVAL notation do everything box notation does, it can do much more. In succeeding chapters we will learn programming constructs made possible by EVAL that have no analog in box notation.

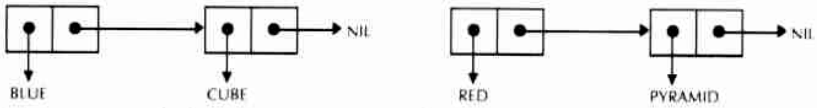


Figure 2.5a The lists (BLUE CUBE) and (RED PYRAMID).

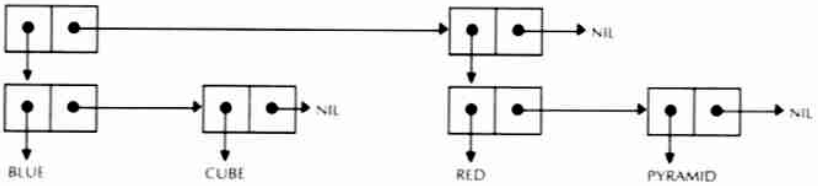


Figure 2.5b The list ((BLUE CUBE) (RED PYRAMID)).

Figure 2.6 shows how cons cell structure gets deeper as depth of parenthesization increases. The lists (PHONE HOME), ((PHONE HOME)), and (((PHONE HOME))) require one, two, and three levels of cons cells, respectively.

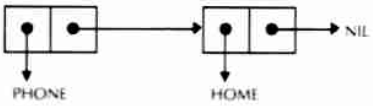


Figure 2.6a The list (PHONE HOME).

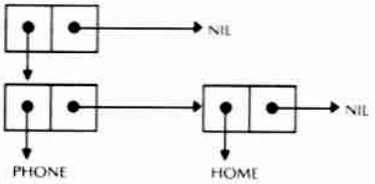


Figure 2.6b The list ((PHONE HOME)).

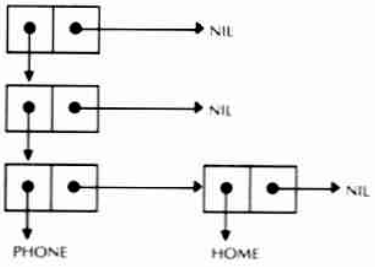


Figure 2.6c The list (((PHONE HOME))).