LISP, Lore, and Logic

An Algebraic View of LISP Programming,
Foundations, and Applications

W. Richard Stark

LISP, Lore, and Logic

An Algebraic View of LISP
Programming, Foundations, and
Applications

With 51 Illustrations

Springer-Verlag
New York Berlin Heidelberg
London Paris Tokyo Hong Kong

W. Richard Stark
Department of Mathematics
University of South Florida
Tampa, Florida 33620
USA

Cover: The leaf-pattern making up the cover's background is suggestive of life processes, particularly morphogenesis. At the level of an individual cell, a structure-generating program written in the cell's DNA duplicates the cell. The molecules of the cell play the role of data, which, since they include the DNA, allows the program to create a copy of itself. Morphogenesis is defined in terms of itself, it is recursive. LISP exhibits two features of this process: recursion and the use of its data to express its programs. At its best, LISP programming is capable of graceful self-reproduction and self-modification suggestive of the metabolic processes of life.

Library of Congress Cataloging-in-Publication Data
Stark, W. Richard.

Lisp, lore, and logic / W. Richard Stark.
p. cm.

Includes bibliographical references.

ISBN-13: 978-0-387-97072-1

e-ISBN-13: 978-1-4613-8931-6

DOI: 10.1007/978-1-4613-8931-6

1. LISP (Computer program language) I. Title.

QA76.73. L23S72 1989

005. 13'3—dc20

89-19651

Printed on acid-free paper.

© 1990 Springer-Verlag New York, Inc.

All rights reserved. This work may not be translated or copied in whole or in part without the written permission of the publisher (Springer-Verlag New York, Inc., 175 Fifth Avenue, New York, NY 10010, USA), except for brief excerpts in connection with reviews or scholarly analysis. Use in connection with any form of information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed is forbidden.

The use of general descriptive names, trade names, trademarks, etc., in this publication, even if the former are not especially identified, is not to be taken as a sign that such names, as understood by the Trade Marks and Merchandise Marks Act, may accordingly be used freely by anyone.

While the advice and information in this book is believed to be true and accurate at the date of going to press, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Camera-ready copy prepared using LaTeX.

9 8 7 6 5 4 3 2 1

For my two most patient and enthusiastic supporters, my Mother and my Wife:

Anna Elizabeth May Stark,
and
Judy Kay Ingrao Stark.

Also, my son

Christopher Jon Stark
who always had a joke when one was needed.

Preface

In a jet – flying to Alaska, Europe, or some other equally interesting destination – the most exciting thoughts that come to my mind are of the beauty of the technology that makes it all possible. It is apparent in the evolution of ideas, from beginnings in engineering, physics, and mathematics to a dramatic and practical conclusion in the airplane. To see this particular vision of beauty, one needs to grasp as much as possible of the whole picture – the abstract theory, the practical technology, and the history of its evolution – all at once.

The impact of our abstract dreams is equally visible in the science of computing. Nowhere is this truer than in and around LISP. The theory and technology of nonnumerical computation reaches from esoteric branches of pure mathematics (algebras developed since the early 1930s) and automatic theorem proving (1950s) through LISP (since the late 1950s) to everyday examples of nonnumerical computation and machine intelligence (today). This beautiful stretch of philosophies, ideas, and technologies is the heart of this book. The body is LISP programming and computing.

Our material has developed out of LISP courses taught over the past ten years at the University of Texas at Austin, California State University at San José, the University of South Florida, and AT&T Bell Laboratories. The initial idea was to augment programming with the theoretical foundations of LISP computation – rather than applications in artificial intelligence (the traditional second theme). This idea was inspired by the professional LISP conferences. Could this deep theoretical approach be brought down to the level of an undergraduate course? Would the average LISP-type person really like it?

Students were enthusiastic from the beginning. Each time the material was taught, their interests directed its development. Given the necessary theoretical foundations, they were naturally drawn into the fantastic side of the subject. There is a good deal of that, so the basic design was embellished with exciting examples from LISP's folklore. Folklore lead to appropriately placed historical

commentary and photographs of LISP's creators. Students liked this too. Eventually, some real LISP experts took notice and suggested that the massive and wandering manuscript be dramatically simplified and streamlined. This is the result.

As the title LISP, Lore, and Logic suggests, this is a multifaceted presentation. In addition to programming and theory, there are colorful applications (organic chemistry, automatic reasoning, symbolic differentiation, translating natural language, ...), bits of the history (is LISP really a descendent of the lofty lambda calculus?) and a small family album, computational philosophy (unsolvable problems), bizarre consequences of LISP's exceptional power (self-processing, self-reproduction, ...), and more. These topics are tied together by the common theme of computing in LISP. In most cases, they are realized in programming examples and exercises. Individual topics were included for various reasons – for practical value, for beauty and mystery, and often to explain how and why LISP works so well.

The level of presentation here is appropriate for strong students with some experience in computing and at least a sophomore's background in mathematics. As a text, it is appropriate as a primary text for theoretically oriented courses in LISP or as a secondary text for more traditional courses in LISP, artificial intelligence with LISP, computing languages, or the abstract theory of computation. A good deal of material is covered without unnecessary prose. This conciseness may appeal to those who are put off by soft and wordy texts. The dialect is Common LISP.

Just as with the jet, the beauty and power of this subject is best appreciated by those who can understand and appreciate it, as well as use it. This little book is designed to provide all of this to any reader having an aptitude for computing and mathematics, and a taste for the exotic.

W. Richard Stark

working one summer late in the 20th century, in the peace of western New Jersey's farm country.

Acknowledgments

During the years of this text's development, I've received help in one way or another from dozens of friends and a few strangers. Both great and humble, they are

Woodie Bledsoe, Kim Bruce, Alonzo Church, Dan Freidman, Nick Gall, Oscar Garcia, Jim Gaughan, Henson Graves, David Gries, William Hatcher, the researchers of IBM archives, Stephen Kleene, Larry Landwebber, Dallas Lankford, Tony Llewellyn, John McCarthy, the researchers of the MIT Museum, Lynn Montz, John Pedersen, the researchers of the Princeton University archives, Nathaniel Rochester, Phillip Scott, Guy Steele, George Stockman, Cecelia Swift, Carolyn Talcott, Yoshiyasu Takefuji, John Teeter, Ralph Tindell, Robert A. Wallace, Eugene Wang, Daniel Weinreb, Elia Weixelbaum, Kris Wiesenfeld, and my students.

Without your help, this book would still be just an idea.

Contents

Preface	vii
Acknowledgments	ix
0 Introduction	xv
1 A First Look at LISP	1
1.1 The origins of LISP.	1
1.2 A glimpse into the structure of LISP.	4
1.2.1 Data.	5
1.2.2 Functions.	6
1.2.3 Expressions in the language.	8
1.2.4 QUOTE.	9
1.2.5 Logical terms.	11
1.2.6 Branching in computations.	12
1.2.7 Lambda functions.	14
1.2.8 Assignments.	16
1.3 Additional reading.	19
2 Elementary Programming	20
2.1 Programming straightforward computations.	20
2.2 Recursive and iterative computations.	25
2.2.1 A general view of recursion.	25
2.2.2 Programming recursive computations.	27
2.2.3 Rules for recursive programming.	33
2.2.4 Iteration.	35
2.3 External files.	39

2.4	Searching.	43
2.4.1	Is searching a sign of ignorance?	43
2.4.2	Kinds of search.	43
2.5	Extra variables and functions, for efficiency.	50
2.6	Other list functions.	54
2.6.1	MEMBER and more.	54
2.6.2	Functions for association lists.	56
2.6.3	Example: translation between Inuit and English.	57
2.7	The fixed-point style of computation.	61
2.8	Input and output.	64
2.9	The procedural and functional styles of computation.	68
2.10	Additional reading	70
3	Deeper into Essential Structure	71
3.1	LISP's data.	71
3.1.1	S-expressions.	72
3.1.2	Representations of data.	74
3.1.3	Atoms.	76
3.1.4	Data types.	81
3.2	Another look at familiar functions	82
3.2.1	CAR, CDR, and CONS: an alternate semantics.	82
3.2.2	EQ and EQUAL.	83
3.2.3	Options for LAMBDA functions.	85
3.2.4	SETF.	87
3.3	LISP syntax in detail.	88
3.3.1	Terms.	89
3.3.2	Variables and forms.	89
3.3.3	Syntax puzzles.	91
3.4	Self-processing.	92
3.5	Bindings, scopes, and environments.	96
3.6	Additional reading.	103
4	Computational Philosophy	104
4.1	Models of computation.	104
4.1.1	Mathematical formalisms for computation.	105

4.1.2	Church's thesis and computational completeness.	107
4.2	Pure LISP.	110
4.3	Types of recursion.	114
4.3.1	Fat recursion and tail recursion.	114
4.3.2	Compound recursion.	115
4.3.3	Monotonic and nonmonotonic recursion.	116
4.4	The limits of LISP: an unsolvable problem.	118
4.5	The folklore of fixed-point computation.	122
4.6	Additional Reading.	125
5	LISP Functions for Powerful Programming	126
5.1	Debugging tools.	126
5.1.1	The listener, access to local bindings, and BACKTRACE.	127
5.1.2	STEP and TRACE.	129
5.1.3	Other tools for debugging.	131
5.1.4	Application: debugging a LISP text (optional).	134
5.2	Applicative operators.	137
5.2.1	APPLY, FUNCALL, MAPCAR, and MAPLIST.	138
5.2.2	Application: symbolic differentiation.	141
5.3	Macros.	145
5.3.1	The basic idea.	145
5.3.2	Applications: INFIX and EVAL&TIME.	148
5.4	Structures, vectors, and arrays.	151
5.4.1	Vectors and arrays.	151
5.4.2	Structures.	153
5.4.3	Application: working with organic molecules.	156
5.5	Function closures.	163
5.5.1	Creating and using closures.	163
5.5.2	Application: COMPOSE and Currying CONS.	166
5.6	COERCion.	168
5.7	Surgical operations.	170
6	Interpreters: from Algebra to LISP	176
6.1	Algebras and interpreting: an abstract view.	176

6.1.1	Algebras of objects and algebras of terms.	177
6.1.2	Interpretation and computation.	179
6.2	LISP interpreters.	183
6.2.1	A sketch of the usual LISP interpreter.	184
6.2.2	A call-by-name interpreter for LISP.	190
6.2.3	Put the interpreter inside, keep the user out.	194
6.3	Compiled LISP.	198
7	Mathematical Foundations of LISP	200
7.1	The lambda calculus defined.	200
7.2	Straightforward computation in the lambda calculus.	205
7.3	Fixed-points for object functions.	208
7.4	Recursive functions in the lambda calculus.	210
7.5	Pure LISP in the lambda calculus.	213
7.6	Additional reading.	215
8	Automatic Reasoning, Algebraic Intelligence	216
8.1	Logics.	217
8.1.1	Propositional logic.	218
8.1.2	Fundamental theorems for the propositional calculus.	223
8.1.3	Other approaches to automatic reasoning in PC.	225
8.2	Predicate logic and unification.	229
8.3	PROLOG.	233
8.4	Speculation on algebraic intelligence.	236
8.5	Additional reading.	239
9	Bibliography	240
10	Answers to Selected Exercises	246
11	Index	269

0

Introduction

This book is a presentation of LISP which is both practical and theoretical. For the practical, the syntax of the language, the programming styles, and the semantics of computation are carefully developed in four of the first five chapters. For the theoretical, the algebra of interpreters, the lambda calculus as a foundation for LISP, and the algebraic significance of LISP's approach to artificial intelligence, are the topics of the last three chapters. As the title LISP, Lore, and Logic suggests, it goes beyond a simple description of the language and its theory. The additional material consists of historical comments and quotations, photographs of many of those involved in LISP's creation, and interesting examples of LISP's applications.

The level of difficulty is that of an undergraduate calculus course. The material has been designed to appeal to a variety of readers, from the bright freshman to the practicing professional and from computer scientists and mathematicians to chemists, engineers, and philosophers. We assume some experience in computing and at least a sophomore's background in mathematics. The course in which this material was developed was a theoretically oriented LISP programming course – it will serve well as the primary text in such courses. This little book may also be used as a secondary text in courses covering artificial intelligence with LISP, programming languages, and the theory of computation.

The text is organized into eight chapters – each corresponding to a view of LISP. Chapter 1, A First Look at LISP, presents a few words on LISP's origins, followed by a broad and lightly detailed overview of LISP and LISP computing. Chapter 2, Elementary Programming, is a moderately complete presentation of LISP programming in terms of fundamental operations and strategies. In these programs, we are still dealing with only the most basic data structures and functions. Chapter 3, Deeper into Essential Structure, goes back to pick up essential syntactic and semantic details not fully developed in the first two. At this point, we have a good general view of programming. In fact, with a few additional exercises and the first two sections from Chapter 5, there would be

a complete baby course in LISP. We now take a short break, giving the reader time to digest Chapters 2 and 3.

In Chapter 4, Computational Philosophy, topics of a more general nature are introduced. The most abstract topic is a sketch of mathematical models of computation. The most practical is a good solid presentation of types of recursion, their computational properties, and efficient strategies for their programming. Chapter 4 concludes with an entertaining section on computational folklore.

Chapter 5, LISP Functions for Powerful Programming, returns to programming to pick up the remaining important ideas. These include debugging tools, applicative operators, macros, structures, function closures, and surgical functions. As applications, there are colorful and nontrivial LISP programs for parallel computing, translating between English and Inuit, altering LISP's listener, processing molecules in organic chemistry, self-processing, and list processing without CONS. We have now completed most of our development of programming.

DIVISION OF TOPICS:

The division of the text between programming, theory, and the algebra of computation is illustrated in this Venn diagram.

The last three chapters develop the theory and tie it to the language. Chapter 6, Interpreters: From Algebra to LISP, describes the workings of interpreters in general and for LISP in particular. This chapter concludes with a glimpse at

compiled functions. Chapter 7, Mathematical Foundations of LISP, is a simplified introduction to the lambda calculus and its relation to LISP. The lambda calculus is interesting as a foundation for LISP and as a bizarre paradigm for computing in general. Finally, Chapter 8, Automatic Reasoning, Algebraic Intelligence, touches on logic, automatic theorem proving, PROLOG, and the algebraic basis for machine intelligence. This chapter makes the important point that classical machine intelligence is based on fundamental connections between syntax and semantics which exist in algebraic languages. PROLOG is introduced as a sibling of LISP.

There are several paths which the teacher or the independent reader may take through this text. The best are illustrated by paths through the accompanying flow chart. Boxes represent programming chapters, circles represent theoretical chapters.

As you get into this text, LISP manuals may be useful. Good manuals listed in the bibliography include Steele[1984] – the bible of Common LISP; Gold Hill Computers[1984] – most of this text’s code was developed on their G(olden)C(ommon)LISP for personal computers; Moon[1974]; Pitman[1981]; and Teitelman[1978]; and Weinreb&Moon[1981]. All of the LISP code in this text has been machine verified (directly from the text files) in the Common

LISP of Gold Hill Computers (GCLISP 1.01). Further, implementation sensitive parts of the code have been verified on Symbolics' Common LISP and Texas Instruments' Explorer Common LISP.

Readers with comments, suggestions, history, photographs, corrections, etc., are encouraged to write to the author. Be sure to include your name and return address.