Structure and Interpretation of Computer Programs

second edition

Harold Abelson and Gerald Jay Sussman with Julie Sussman

foreword by Alan J. Perlis

The MIT Press Cambridge, Massachusetts London, England

The McGraw-Hill Companies, Inc. New York St. Louis San Francisco Montreal Toronto This book is one of a series of texts written by faculty of the Electrical Engineering and Computer Science Department at the Massachusetts Institute of Technology. It was edited and produced by The MIT Press under a joint production-distribution arrangement with The McGraw-Hill Companies, Inc.

Ordering Information:

North America Text orders should be addressed to: The McGraw-Hill Companies Order Services P.O. Box 545 Blacklick, OH 43004-0545 For toll-free customer service, call 1-800-338-3987

All other orders should be addressed to: The MIT Press 55 Hayward Street Cambridge, MA 02142 or at the toll-free number 1-800-356-0343

Outside North America

All orders should be addressed to The MIT Press or its local distributor.

©1996 by The Massachusetts Institute of Technology Second edition

All rights reserved. No part of this book may be reproduced in any form or by any electronic or mechanical means (including photocopying, recording, or information storage and retrieval) without permission in writing from the publisher.

This book was set by the authors using the LTEX typesetting system and was printed and bound in the United States of America.

Library of Congress Cataloging-in-Publication Data Abelson, Harold

Structure and interpretation of computer programs / Harold Abelson and Gerald Jay Sussman, with Julie Sussman.—2nd ed.

p. cm.—(Electrical engineering and computer science series)

Includes bibliographical references and index.

ISBN 0-262-01153-0; ISBN-13 978-0-262-01153-2 (MIT Press hc) ISBN 0-262-51087-1; ISBN-13 978-0-262-51087-5 (MIT Press pbk) ISBN 0-07-000484-6; ISBN-13 978-0-07-000484-9 (McGraw Hill hc)

1. Electronic digital computers—Programming. 2. LISP (Computer
program language) I. Sussman, Gerald Jay. II. Sussman, Julie.III. Title. IV. Series: MIT electrical engineering and computer
science series.QA76.6.A2551996005.13'3—dc2096-17756

20 19 18 17 16 15

This book is dedicated, in respect and admiration, to the spirit that lives in the computer.

"I think that it's extraordinarily important that we in computer science keep fun in computing. When it started out, it was an awful lot of fun. Of course, the paying customers got shafted every now and then, and after a while we began to take their complaints seriously. We began to feel as if we really were responsible for the successful, error-free perfect use of these machines. I don't think we are. I think we're responsible for stretching them, setting them off in new directions, and keeping fun in the house. I hope the field of computer science never loses its sense of fun. Above all, I hope we don't become missionaries. Don't feel as if you're Bible salesmen. The world has too many of those already. What you know about computing other people will learn. Don't feel as if the key to successful computing is only in your hands. What's in your hands, I think and hope, is intelligence: the ability to see the machine as more than when you were first led up to it, that you can make it more."

Alan J. Perlis (April 1, 1922–February 7, 1990)

Contents

1

Contents					
Foreword					
Preface to the Second Edition					
Preface to the First Edition					
Acknowledgments					
Building Abstractions with Procedures 1					
 1.1 The Elements of Programming 1.1 Expressions 1.1.2 Naming and the Environment 1.1.3 Evaluating Combinations 1.1.4 Compound Procedures 1.1.5 The Substitution Model for Procedure Application 1.1.6 Conditional Expressions and Predicates 1.1.7 Example: Square Roots by Newton's Method 1.1.8 Procedures as Black-Box Abstractions 	4 5 7 9 11 13 17 21 26				
 Procedures and the Processes They Generate Linear Recursion and Iteration Tree Recursion Orders of Growth A Exponentiation Greatest Common Divisors Example: Testing for Primality 	31 32 37 42 44 48 50				
 1.3 Formulating Abstractions with Higher-Order Procedures 1.3.1 Procedures as Arguments 1.3.2 Constructing Procedures Using Lambda 1.3.3 Procedures as General Methods 1.3.4 Procedures as Returned Values 	56 57 62 66 72				

2	Build	ing Abstractions with Data	79
	2.1	Introduction to Data Abstraction	83
	2.1.1	Example: Arithmetic Operations for Rational Numbers	83
	2.1.2	Abstraction Barriers	87
	2.1.3	What Is Meant by Data?	90
	2.1.4	Extended Exercise: Interval Arithmetic	93
	2.2	Hierarchical Data and the Closure Property	97
	2.2.1	Representing Sequences	99
	2.2.2	Hierarchical Structures	107
	2.2.3	Sequences as Conventional Interfaces	113
	2.2.4	Example: A Picture Language	126
	2.3	Symbolic Data	142
	2.3.1	Quotation	142
	2.3.2	Example: Symbolic Differentiation	145
	2.3.3	Example: Representing Sets	151
	2.3.4	Example: Huffman Encoding Trees	161
	2.4	Multiple Representations for Abstract Data	169
	2.4.1	Representations for Complex Numbers	171
	2.4.2	Tagged data	175
	2.4.3	Data-Directed Programming and Additivity	179
	2.5	Systems with Generic Operations	187
	2.5.1	Generic Arithmetic Operations	189
	2.5.2	Combining Data of Different Types	193
	2.5.3	Example: Symbolic Algebra	202
3	Modu	ularity, Objects, and State	217
	3.1	Assignment and Local State	218
	3.1.1	Local State Variables	219
	3.1.2	The Benefits of Introducing Assignment	225
	3.1.3	The Costs of Introducing Assignment	229
	3.2	The Environment Model of Evaluation	236
	3.2.1	The Rules for Evaluation	238
	3.2.2	Applying Simple Procedures	241
	3.2.3	Frames as the Repository of Local State	244
	3.2.4	Internal Definitions	249
	3.3	Modeling with Mutable Data	251
	3.3.1	Mutable List Structure	252

	3.3.2	Representing Queues	261
	3.3.3	Representing Tables	266
	3.3.4	A Simulator for Digital Circuits	273
	3.3.5	Propagation of Constraints	285
	3.4	Concurrency: Time Is of the Essence	297
	3.4.1	The Nature of Time in Concurrent Systems	298
	3.4.2	Mechanisms for Controlling Concurrency	303
	3.5 3.5.1 3.5.2 3.5.3 3.5.4 3.5.5	Streams Streams Are Delayed Lists Infinite Streams Exploiting the Stream Paradigm Streams and Delayed Evaluation Modularity of Functional Programs and Modularity of Objects	 316 317 326 334 346 352
4	Meta	linguistic Abstraction	359
	4.1	The Metacircular Evaluator	362
	4.1.1	The Core of the Evaluator	364
	4.1.2	Representing Expressions	368
	4.1.3	Evaluator Data Structures	376
	4.1.4	Running the Evaluator as a Program	381
	4.1.5	Data as Programs	384
	4.1.6	Internal Definitions	388
	4.1.7	Separating Syntactic Analysis from Execution	393
	4.2	Variations on a Scheme—Lazy Evaluation	398
	4.2.1	Normal Order and Applicative Order	399
	4.2.2	An Interpreter with Lazy Evaluation	401
	4.2.3	Streams as Lazy Lists	409
	4.3	Variations on a Scheme—Nondeterministic Computing	412
	4.3.1	Amb and Search	414
	4.3.2	Examples of Nondeterministic Programs	418
	4.3.3	Implementing the Amb Evaluator	426
	4.4	Logic Programming	438
	4.4.1	Deductive Information Retrieval	441
	4.4.2	How the Query System Works	453
	4.4.3	Is Logic Programming Mathematical Logic?	462
	4.4.4	Implementing the Query System	468

Contents

5	Computing with Register Machines		
	5.1	Designing Register Machines	492
	5.1.1	A Language for Describing Register Machines	494
	5.1.2	Abstraction in Machine Design	499
	5.1.3	Subroutines	502
	5.1.4	Using a Stack to Implement Recursion	506
	5.1.5	Instruction Summary	512
	5.2	A Register-Machine Simulator	513
	5.2.1	The Machine Model	515
	5.2.2	The Assembler	520
	5.2.3	Generating Execution Procedures for Instructions	523
	5.2.4	Monitoring Machine Performance	530
	5.3	Storage Allocation and Garbage Collection	533
	5.3.1	Memory as Vectors	534
	5.3.2	Maintaining the Illusion of Infinite Memory	540
	5.4	The Explicit-Control Evaluator	547
	5.4.1	The Core of the Explicit-Control Evaluator	549
	5.4.2	Sequence Evaluation and Tail Recursion	555
	5.4.3	Conditionals, Assignments, and Definitions	558
	5.4.4	Running the Evaluator	560
	5.5	Compilation	566
	5.5.1	Structure of the Compiler	569
	5.5.2	Compiling Expressions	574
	5.5.3	Compiling Combinations	581
	5.5.4	Combining Instruction Sequences	587
	5.5.5	An Example of Compiled Code	591
	5.5.6	Lexical Addressing	600
	5.5.7	Interfacing Compiled Code to the Evaluator	603
	Refer	ences	611
	List of Exercises		
	Index		621

<u>x</u>_____