

# Table of Contents

<b>To the Reader</b>	<b>xiii</b>
<b>1 The Basics of Interpretation</b>	<b>1</b>
1.1 Evaluation . . . . .	2
1.2 Basic Evaluator . . . . .	3
1.3 Evaluating Atoms . . . . .	4
1.4 Evaluating Forms . . . . .	6
1.4.1 Quoting . . . . .	7
1.4.2 Alternatives . . . . .	8
1.4.3 Sequence . . . . .	9
1.4.4 Assignment . . . . .	11
1.4.5 Abstraction . . . . .	11
1.4.6 Functional Application . . . . .	11
1.5 Representing the Environment . . . . .	12
1.6 Representing Functions . . . . .	15
1.6.1 Dynamic and Lexical Binding . . . . .	19
1.6.2 Deep or Shallow Implementation . . . . .	23
1.7 Global Environment . . . . .	25
1.8 Starting the Interpreter . . . . .	27
1.9 Conclusions . . . . .	28
1.10 Exercises . . . . .	28
<b>2 Lisp, 1, 2, . . . <math>\omega</math></b>	<b>31</b>
2.1 Lisp <sub>1</sub> . . . . .	32
2.2 Lisp <sub>2</sub> . . . . .	32
2.2.1 Evaluating a Function Term . . . . .	35
2.2.2 Duality of the Two Worlds . . . . .	36
2.2.3 Using Lisp <sub>2</sub> . . . . .	38
2.2.4 Enriching the Function Environment . . . . .	38
2.3 Other Extensions . . . . .	39
2.4 Comparing Lisp <sub>1</sub> and Lisp <sub>2</sub> . . . . .	40
2.5 Name Spaces . . . . .	43
2.5.1 Dynamic Variables . . . . .	44
2.5.2 Dynamic Variables in COMMON LISP . . . . .	48
2.5.3 Dynamic Variables without a Special Form . . . . .	50
2.5.4 Conclusions about Name Spaces . . . . .	52

2.6	Recursion . . . . .	53
2.6.1	Simple Recursion . . . . .	54
2.6.2	Mutual Recursion . . . . .	55
2.6.3	Local Recursion in Lisp <sub>2</sub> . . . . .	56
2.6.4	Local Recursion in Lisp <sub>1</sub> . . . . .	57
2.6.5	Creating Uninitialized Bindings . . . . .	60
2.6.6	Recursion without Assignment . . . . .	62
2.7	Conclusions . . . . .	67
2.8	Exercises . . . . .	68
<b>3</b>	<b>Escape &amp; Return: Continuations</b>	<b>71</b>
3.1	Forms for Handling Continuations . . . . .	74
3.1.1	The Pair <b>catch/throw</b> . . . . .	74
3.1.2	The Pair <b>block/return-from</b> . . . . .	76
3.1.3	Escapes with a Dynamic Extent . . . . .	77
3.1.4	Comparing <b>catch</b> and <b>block</b> . . . . .	79
3.1.5	Escapes with Indefinite Extent . . . . .	81
3.1.6	Protection . . . . .	84
3.2	Actors in a Computation . . . . .	87
3.2.1	A Brief Review about Objects . . . . .	87
3.2.2	The Interpreter for Continuations . . . . .	89
3.2.3	Quoting . . . . .	90
3.2.4	Alternatives . . . . .	90
3.2.5	Sequence . . . . .	90
3.2.6	Variable Environment . . . . .	91
3.2.7	Functions . . . . .	92
3.3	Initializing the Interpreter . . . . .	94
3.4	Implementing Control Forms . . . . .	95
3.4.1	Implementation of <b>call/cc</b> . . . . .	95
3.4.2	Implementation of <b>catch</b> . . . . .	96
3.4.3	Implementation of <b>block</b> . . . . .	97
3.4.4	Implementation of <b>unwind-protect</b> . . . . .	99
3.5	Comparing <b>call/cc</b> to <b>catch</b> . . . . .	100
3.6	Programming by Continuations . . . . .	102
3.6.1	Multiple Values . . . . .	103
3.6.2	Tail Recursion . . . . .	104
3.7	Partial Continuations . . . . .	106
3.8	Conclusions . . . . .	107
3.9	Exercises . . . . .	108
<b>4</b>	<b>Assignment and Side Effects</b>	<b>111</b>
4.1	Assignment . . . . .	111
4.1.1	Boxes . . . . .	114
4.1.2	Assignment of Free Variables . . . . .	116
4.1.3	Assignment of a Predefined Variable . . . . .	121
4.2	Side Effects . . . . .	121
4.2.1	Equality . . . . .	122

## TABLE OF CONTENTS

vii

4.2.2	Equality between Functions . . . . .	125
4.3	Implementation . . . . .	127
4.3.1	Conditional . . . . .	128
4.3.2	Sequence . . . . .	129
4.3.3	Environment . . . . .	129
4.3.4	Reference to a Variable . . . . .	130
4.3.5	Assignment . . . . .	130
4.3.6	Functional Application . . . . .	131
4.3.7	Abstraction . . . . .	131
4.3.8	Memory . . . . .	132
4.3.9	Representing Values . . . . .	133
4.3.10	A Comparison to Object Programming . . . . .	135
4.3.11	Initial Environment . . . . .	135
4.3.12	Dotted Pairs . . . . .	137
4.3.13	Comparisons . . . . .	137
4.3.14	Starting the Interpreter . . . . .	138
4.4	Input/Output and Memory . . . . .	139
4.5	Semantics of Quotations . . . . .	140
4.6	Conclusions . . . . .	145
4.7	Exercises . . . . .	145
<b>5</b>	<b>Denotational Semantics</b> . . . . .	<b>147</b>
5.1	A Brief Review of $\lambda$ -Calculus . . . . .	149
5.2	Semantics of Scheme . . . . .	151
5.2.1	References to a Variable . . . . .	153
5.2.2	Sequence . . . . .	154
5.2.3	Conditional . . . . .	155
5.2.4	Assignment . . . . .	157
5.2.5	Abstraction . . . . .	157
5.2.6	Functional Application . . . . .	158
5.2.7	call/cc . . . . .	158
5.2.8	Tentative Conclusions . . . . .	159
5.3	Semantics of $\lambda$ -calcul . . . . .	159
5.4	Functions with Variable Arity . . . . .	161
5.5	Evaluation Order for Applications . . . . .	164
5.6	Dynamic Binding . . . . .	167
5.7	Global Environment . . . . .	170
5.7.1	Global Environment in Scheme . . . . .	170
5.7.2	Automatically Extendable Environment . . . . .	172
5.7.3	Hyperstatic Environment . . . . .	173
5.8	Beneath This Chapter . . . . .	174
5.9	$\lambda$ -calculus and Scheme . . . . .	175
5.9.1	Passing Continuations . . . . .	177
5.9.2	Dynamic Environment . . . . .	180
5.10	Conclusions . . . . .	180
5.11	Exercises . . . . .	181

<b>6</b>	<b>Fast Interpretation</b>	<b>183</b>
6.1	A Fast Interpreter . . . . .	183
6.1.1	Migration of Denotations . . . . .	184
6.1.2	Activation Record . . . . .	184
6.1.3	The Interpreter: the Beginning . . . . .	187
6.1.4	Classifying Variables . . . . .	191
6.1.5	Starting the Interpreter . . . . .	195
6.1.6	Functions with Variable Arity . . . . .	196
6.1.7	Reducible Forms . . . . .	197
6.1.8	Integrating Primitives . . . . .	199
6.1.9	Variations on Environments . . . . .	202
6.1.10	Conclusions: Interpreter with Migrated Computations . . . . .	205
6.2	Rejecting the Environment . . . . .	206
6.2.1	References to Variables . . . . .	208
6.2.2	Alternatives . . . . .	209
6.2.3	Sequence . . . . .	209
6.2.4	Abstraction . . . . .	209
6.2.5	Applications . . . . .	210
6.2.6	Conclusions: Interpreter with Environment in a Register . . . . .	211
6.3	Diluting Continuations . . . . .	211
6.3.1	Closures . . . . .	211
6.3.2	The Pretreater . . . . .	212
6.3.3	Quoting . . . . .	212
6.3.4	References . . . . .	212
6.3.5	Conditional . . . . .	213
6.3.6	Assignment . . . . .	213
6.3.7	Sequence . . . . .	214
6.3.8	Abstraction . . . . .	214
6.3.9	Application . . . . .	215
6.3.10	Reducible Forms . . . . .	216
6.3.11	Calling Primitives . . . . .	218
6.3.12	Starting the Interpreter . . . . .	218
6.3.13	The Function <code>call/cc</code> . . . . .	219
6.3.14	The Function <code>apply</code> . . . . .	219
6.3.15	Conclusions: Interpreter without Continuations . . . . .	220
6.4	Conclusions . . . . .	221
6.5	Exercises . . . . .	221
<b>7</b>	<b>Compilation</b>	<b>223</b>
7.1	Compiling into Bytes . . . . .	225
7.1.1	Introducing the Register <code>*val*</code> . . . . .	225
7.1.2	Inventing the Stack . . . . .	226
7.1.3	Customizing Instructions . . . . .	228
7.1.4	Calling Protocol for Functions . . . . .	230
7.2	Language and Target Machine . . . . .	231
7.3	Disassembly . . . . .	235
7.4	Coding Instructions . . . . .	235

## TABLE OF CONTENTS

ix

7.5	Instructions . . . . .	238
7.5.1	Local Variables . . . . .	239
7.5.2	Global Variables . . . . .	240
7.5.3	Jumps . . . . .	242
7.5.4	Invocations . . . . .	243
7.5.5	Miscellaneous . . . . .	244
7.5.6	Starting the Compiler-Interpreter . . . . .	245
7.5.7	Catching Our Breath . . . . .	247
7.6	Continuations . . . . .	247
7.7	Escapes . . . . .	249
7.8	Dynamic Variables . . . . .	252
7.9	Exceptions . . . . .	255
7.10	Compiling Separately . . . . .	260
7.10.1	Compiling a File . . . . .	260
7.10.2	Building an Application . . . . .	262
7.10.3	Executing an Application . . . . .	266
7.11	Conclusions . . . . .	267
7.12	Exercises . . . . .	268
<b>8</b>	<b>Evaluation &amp; Reflection</b>	<b>271</b>
8.1	Programs and Values . . . . .	271
8.2	<code>eval</code> as a Special Form . . . . .	277
8.3	Creating Global Variables . . . . .	279
8.4	<code>eval</code> as a Function . . . . .	280
8.5	The Cost of <code>eval</code> . . . . .	281
8.6	Interpreted <code>eval</code> . . . . .	282
8.6.1	Can Representations Be Interchanged? . . . . .	282
8.6.2	Global Environment . . . . .	283
8.7	Reifying Environments . . . . .	286
8.7.1	Special Form <code>export</code> . . . . .	286
8.7.2	The Function <code>eval/b</code> . . . . .	289
8.7.3	Enriching Environments . . . . .	290
8.7.4	Reifying a Closed Environment . . . . .	292
8.7.5	Special Form <code>import</code> . . . . .	296
8.7.6	Simplified Access to Environments . . . . .	301
8.8	Reflective Interpreter . . . . .	302
8.9	Conclusions . . . . .	308
8.10	Exercises . . . . .	309
<b>9</b>	<b>Macros: Their Use &amp; Abuse</b>	<b>311</b>
9.1	Preparation for Macros . . . . .	312
9.1.1	Multiple Worlds . . . . .	313
9.1.2	Unique World . . . . .	313
9.2	Macro Expansion . . . . .	314
9.2.1	Exogenous Mode . . . . .	314
9.2.2	Endogenous Mode . . . . .	316
9.3	Calling Macros . . . . .	317

9.4	Expanders . . . . .	318
9.5	Acceptability of an Expanded Macro . . . . .	320
9.6	Defining Macros . . . . .	321
9.6.1	Multiple Worlds . . . . .	322
9.6.2	Unique World . . . . .	325
9.6.3	Simultaneous Evaluation . . . . .	331
9.6.4	Redefining Macros . . . . .	331
9.6.5	Comparisons . . . . .	332
9.7	Scope of Macros . . . . .	333
9.8	Evaluation and Expansion . . . . .	336
9.9	Using Macros . . . . .	338
9.9.1	Other Characteristics . . . . .	339
9.9.2	Code Walking . . . . .	340
9.10	Unexpected Captures . . . . .	341
9.11	A Macro System . . . . .	344
9.11.1	Objectification—Making Objects . . . . .	344
9.11.2	Special Forms . . . . .	350
9.11.3	Evaluation Levels . . . . .	351
9.11.4	The Macros . . . . .	352
9.11.5	Limits . . . . .	355
9.12	Conclusions . . . . .	356
9.13	Exercises . . . . .	356
<b>10</b>	<b>Compiling into C</b> . . . . .	<b>359</b>
10.1	Objectification . . . . .	360
10.2	Code Walking . . . . .	360
10.3	Introducing Boxes . . . . .	362
10.4	Eliminating Nested Functions . . . . .	363
10.5	Collecting Quotations and Functions . . . . .	367
10.6	Collecting Temporary Variables . . . . .	370
10.7	Taking a Pause . . . . .	371
10.8	Generating C . . . . .	372
10.8.1	Global Environment . . . . .	373
10.8.2	Quotations . . . . .	375
10.8.3	Declaring Data . . . . .	378
10.8.4	Compiling Expressions . . . . .	379
10.8.5	Compiling Functional Applications . . . . .	382
10.8.6	Predefined Environment . . . . .	384
10.8.7	Compiling Functions . . . . .	385
10.8.8	Initializing the Program . . . . .	387
10.9	Representing Data . . . . .	390
10.9.1	Declaring Values . . . . .	393
10.9.2	Global Variables . . . . .	395
10.9.3	Defining Functions . . . . .	396
10.10	Execution Library . . . . .	397
10.10.1	Allocation . . . . .	397
10.10.2	Functions on Pairs . . . . .	398

## TABLE OF CONTENTS

xi

10.10.3 Invocation . . . . .	399
10.11 <b>call/cc: To Have and Have Not</b> . . . . .	402
10.11.1 The Function <b>call/ep</b> . . . . .	403
10.11.2 The Function <b>call/cc</b> . . . . .	404
10.12 Interface with C . . . . .	413
10.13 Conclusions . . . . .	414
10.14 Exercises . . . . .	414
<b>11 Essence of an Object System</b>	<b>417</b>
11.1 Foundations . . . . .	419
11.2 Representing Objects . . . . .	420
11.3 Defining Classes . . . . .	422
11.4 Other Problems . . . . .	425
11.5 Representing Classes . . . . .	426
11.6 Accompanying Functions . . . . .	429
11.6.1 Predicates . . . . .	430
11.6.2 Allocator without Initialization . . . . .	431
11.6.3 Allocator with Initialization . . . . .	433
11.6.4 Accessing Fields . . . . .	435
11.6.5 Accessors for Reading Fields . . . . .	436
11.6.6 Accessors for Writing Fields . . . . .	437
11.6.7 Accessors for Length of Fields . . . . .	438
11.7 Creating Classes . . . . .	439
11.8 Predefined Accompanying Functions . . . . .	440
11.9 Generic Functions . . . . .	441
11.10 Method . . . . .	446
11.11 Conclusions . . . . .	448
11.12 Exercises . . . . .	448
<b>Answers to Exercises</b>	<b>451</b>
<b>Bibliography</b>	<b>481</b>
<b>Index</b>	<b>495</b>