Introduction to Computing

Explorations in Language, Logic, and Machines

Fall 2009

David Evans
University of Virginia

For the latest version of this book and supplementary materials, visit:

http://computingbook.org

Printed: 19 August 2009

creative commons
Attribution-Noncommercial-Share Alike 3.0 United States License

Contents

1 Computing **1**

1.1 Processes, Procedures, and Computers . . . 2

1.2 Measuring Computing Power . . . 3

1.2.1 Information . . . 4

1.2.2 Representing Data . . . 9

1.2.3 Growth of Computing Power . . . 13

1.3 Science, Engineering, and Liberal Art . . . 14

1.4 Summary and Roadmap . . . 17

Part I Defining Procedures

2 Language **23**

2.1 Surface Forms and Meanings . . . 23

2.2 Language Construction . . . 24

2.3 Recursive Transition Networks . . . 26

2.4 Replacement Grammars . . . 31

2.5 Summary . . . 39

3 Programming **41**

3.1 Problems with Natural Languages . . . 42

3.2 Programming Languages . . . 43

3.3 Scheme . . . 45

3.4 Expressions . . . 46

3.4.1 Primitives . . . 46

3.4.2 Application Expressions . . . 47

3.5 Definitions . . . 51

3.6 Procedures . . . 52

3.6.1 Making Procedures . . . 53

3.6.2 Substitution Model of Evaluation . . . 53

3.7 Decisions . . . 56

3.8 Evaluation Rules . . . 58

3.9 Summary . . . 60

4 Problems and Procedures **61**

4.1 Solving Problems . . . 61

4.2 Composing Procedures . . . 62

4.2.1 Procedures as Inputs and Outputs . . . 63

4.3 Recursive Problem Solving . . . 65

4.4 Evaluating Recursive Applications . . . 74

4.5 Developing Complex Programs . . . 77
4.5.1 Printing . . . 78
4.5.2 Tracing . . . 79
4.6 Summary . . . 83

5 Data **85**
5.1 Types . . . 85
5.2 Pairs . . . 87
5.2.1 Making Pairs . . . 90
5.2.2 Triples to Octuples . . . 91
5.3 Lists . . . 92
5.4 List Procedures . . . 94
5.4.1 Procedures that Examine Lists . . . 95
5.4.2 Generic Accumulators . . . 96
5.4.3 Procedures that Construct Lists . . . 99
5.5 Lists of Lists . . . 103
5.6 Data Abstraction . . . 105
5.7 Summary of Part I . . . 116

Part II Analyzing Procedures

6 Machines **121**
6.1 History of Computing Machines . . . 122
6.2 Mechanizing Logic . . . 124
6.2.1 Implementing Logic . . . 125
6.2.2 Composing Operations . . . 127
6.2.3 Arithmetic . . . 130
6.3 Modeling Computing . . . 133
6.3.1 Turing Machines . . . 136
6.4 Summary . . . 141

7 Cost **143**
7.1 Empirical Measurements . . . 143
7.2 Orders of Growth . . . 148
7.2.1 Big *O* . . . 148
7.2.2 Omega . . . 152
7.2.3 Theta . . . 153
7.3 Analyzing Procedures . . . 155
7.3.1 Input Size . . . 155
7.3.2 Running Time . . . 156
7.3.3 Worst Case Input . . . 157
7.4 Growth Rates . . . 158
7.4.1 No Growth: Constant Time . . . 159
7.4.2 Linear Growth . . . 159
7.4.3 Quadratic Growth . . . 165
7.4.4 Exponential Growth . . . 167
7.4.5 Faster than Exponential Growth . . . 169
7.4.6 Non-terminating Procedures . . . 169
7.5 Summary . . . 170

8 Sorting and Searching **173**
8.1 Sorting . . . 173
8.1.1 Best-First Sort . . . 173
8.1.2 Insertion Sort . . . 178
8.1.3 Quicker Sorting . . . 179
8.1.4 Binary Trees . . . 182
8.1.5 Quicksort . . . 187
8.2 Searching . . . 189
8.2.1 Unstructured Search . . . 189
8.2.2 Binary Search . . . 190
8.2.3 Indexed Search . . . 191
8.3 Summary . . . 201

Part III Improving Expressiveness

9 Mutation **205**
9.1 Assignment . . . 205
9.2 Impact of Mutation . . . 207
9.2.1 Names, Places, Frames, and Environments . . . 208
9.2.2 Evaluation Rules with State . . . 209
9.3 Mutable Pairs and Lists . . . 213
9.4 Imperative Programming . . . 215
9.4.1 List Mutators . . . 215
9.4.2 Imperative Control Structures . . . 218
9.5 Summary . . . 221

10 Objects **223**
10.1 Packaging Procedures and State . . . 224
10.1.1 Encapsulation . . . 224
10.1.2 Messages . . . 225
10.1.3 Object Terminology . . . 228
10.2 Inheritance . . . 229
10.2.1 Implementing Subclasses . . . 231
10.2.2 Overriding Methods . . . 234
10.3 Object-Oriented Programming . . . 237
10.4 Summary . . . 240

11 Interpreters **241**
11.1 Python . . . 242
11.1.1 Python Programs . . . 244
11.1.2 Data Types . . . 247
11.1.3 Applications and Invocations . . . 250
11.1.4 Control Statements . . . 251
11.2 Parser . . . 252
11.3 Evaluator . . . 255
11.3.1 Primitives . . . 255
11.3.2 If Expressions . . . 257
11.3.3 Definitions and Names . . . 258
11.3.4 Procedures . . . 259

11.3.5 Application 260
11.3.6 Finishing the Interpreter 261
11.4 Lazy Evaluation 262
11.4.1 Lazy Interpreter 262
11.4.2 Lazy Programming 265
11.5 Summary 267

Part IV The Limits of Computing

12 Computability 271
12.1 Mechanizing Reasoning 271
12.1.1 Gödel’s Incompleteness Theorem 274
12.2 The Halting Problem 275
12.3 Universality 278
12.4 Proving Non-Computability 280
12.5 Summary 286

13 Intractability *not yet available*

List of Figures

1.1 Using three bits to distinguish eight possible values. 6

2.1 Simple recursive transition network. 27
2.2 RTN with a cycle. 28
2.3 RTN for Exercise 2.6. 28
2.4 Recursive transition network with subnetworks. 29
2.5 Alternate *Noun* subnetwork. 29
2.6 RTN generating “Alice runs”. 31
2.7 Derivation of **37** from *Number*. 34
2.8 System power relationships. 36
2.9 Converting the *Number* productions to an RTN. 37
2.10 Converting the *MoreDigits* productions to an RTN. 38
2.11 Converting the *Digit* productions to an RTN. 38

3.1 Running a Scheme program. 46

4.1 A procedure maps inputs to an output. 62
4.2 Composition. 63
4.3 Circular Composition. 65
4.4 Recursive Composition. 66
4.5 Cornering the Queen. 82

5.1 Pegboard Puzzle. 106

6.1 Computing *and* with wine. 126
6.2 Computing logical *or* and *not* with wine 128
6.3 Computing *and3* by composing two *and* functions. 129
6.4 Sample input devices. 134

6.5 Sample output devices. 134
6.6 Turing Machine model. 136
6.7 Rules for checking balanced parentheses Turing Machine. 139
6.8 Checking parentheses Turing Machine. 139

7.1 Evaluation of *fibo* procedure. 147
7.2 Visualization of the sets $O(f)$, $\Omega(f)$, and $\Theta(f)$. 149
7.3 Orders of Growth. 150

8.1 Unbalanced trees. 187

9.1 Sample environments. 209
9.2 Environment created to evaluate (*bigger* 3 4). 211
9.3 Environment after evaluating (**define** *inc* (*make-adder* 1)). . . . 211
9.4 Environment for evaluating the body of (*inc* 149). 212
9.5 Mutable pair created by evaluating (*set-mcdr! pair pair*). 213
9.6 MutableList created by evaluating (*mlist* 1 2 3). 214

10.1 Environment produced by evaluating: 225
10.2 Inheritance Hierarchy. 230
10.3 Counter class hierarchy. 235

12.1 Incomplete and inconsistent axiomatic systems. 273
12.2 Universal Turing Machine. 279
12.3 Two-state Busy Beaver Machine. 285

List of Explorations

1.1 Guessing Numbers 7
1.2 Twenty Questions 8
2.1 Power of Language Systems 36
4.1 Square Roots 72
4.2 Recipes for π 80
4.3 Recursive Definitions and Games 81
5.1 Pascal's Triangle 104
5.2 Pegboard Puzzle 106
7.1 Multiplying Like Rabbits 145
8.1 Searching the Web 199
12.1 Virus Detection 281
12.2 Busy Beavers 284

Index

O, 148
Ω, 152
Θ, 153
π, 80
Gödel, 274
Entscheidungsproblem, 140
Elements, 271
Organon, 271

abacus, 122
abstraction, **16**, 44, 51, 117
accumulate, 69
accumulators, 96
Ada, Countess of Lovelace, 41, 123, 141
algorithm, **2**, 61
aliasing, 217, **217**
alphabet, **26**
Analytical Engine, 41
any-uple, 92
Apollo Guidance Computer, 13
append, 101
application, 54
apply, 228
Aristotle, 271
assignment, 205, **205**, 245
asymptotic operators, 148
axiomatic system, **271**

Babbage, Charles, 123, 148
Backus, John, 31
Backus-Naur Form, 31
base case, **34**, **66**
begin, 206
best-first-sort, 173
bigger, 86
binary numbers, **9**
binary question, **4**
binary search, 190
binary tree, **6**, 182, 190
binomial expansion, 104
bit, **4**
Bletchley Park, 140
Bonacci, Filius, 145
Boole, George, 124
Boolean, 47, 85, 124
Boolean logic, **124**
brute force, **106**
busy beaver problem, 284

car, 87
Carroll, Lewis, 241
cdr, 87
Church, Alonzo, 140
class, **229**, 258
coffee, 2
Colbert, Stephen, 26
compiler, **44**
complete, **272**
compose, 64
composition, 62
computability, **271**
computable, 271, **276**
computer, **3**
computing machines, 122
cond, 226
conditional expression, 226
cons, 87
consistent, **272**
constant time, **159**
constructors, **229**
Corner the Queen, 82
countable, **9**
counter, 207, 224
counting, 122

Dahl, Ole Johan, 238
data abstraction, **105**
datatype, **85**
Davis, Miles, 41
debugging, **77**
defensive programming, **98**, 108
definition, 51, 206, 228
depth, **6**, **183**
derivation, **32**
diagonalization, **9**
dictionary, 249
digital abstraction, **126**
Digital Equipment Corporation, 238
Dijkstra, Edsger, 85, 241
discrete, 9
display, 78
divide-and-conquer, **62**
domain names, 35
DrScheme, 45
dynamic dispatch, **236**
dynamic programming, **147**

Einstein, Albert, 41, 83
else, 227
encapsulation, 224, **224**
Engelbart, Douglas, 238
ENIAC, 237
Enigma, 140

environment, **208**
Epimenides paradox, 274
eq?, 107, 218
equal?, 175
Erdös, Paul, 2
Euclid, 70, 271
Euler, Leonhard, 277
evaluation, **46**
evaluation stack, 76
evaluator, **241**, 255
exponential, 167
expression, **46**
expt, 81

factorial, 67, **67**, 103
fcompose, 64
Feynman, Richard, 81
Fibonacci, 145, 149
filtering, 100
Fisher, George, 15
flattening lists, 103
format, 108
Forrester, Jay, 237
Fortran, 31
frame, **208**
Franklin, Benjamin, 41
function, **47**
functional programming, 215

games, 81
Gardner, Martin, 82
Gauss, Karl, 68
global environment, 208
Goldbach's Conjecture, 277
Goldbach, Christian, 277
Google, 1
grammar, 31
growth rates, 158

halting problem, 275
Heron, 72
higher-order procedure, **63**
Hilbert, David, 140, 271
Hopper, Grace, 41, 44, 78

immutable, **213**
imperative programming, 215, **215**
inc, 64
incompleteness, 274
indexed search, 191
information, **4**
information processes, **2**
inheritance, **224**
inherits, **231**
instance variables, **228**
interpreter, **44**, **241**
intractability, 167
intsto, 102
invoke, **229**
Isaacs, Rufus, 82

Jefferson, Thomas, 23, 26

Kay, Alan, 238, 239
khipu, 122
King, Martin Luther, 41

lambda, 53
Lambda calculus, 140
language, **23**
lazy evaluation, 262, **262**
Leibniz, Gottfried, 80, 121, 123
length, 95
let expression, 176
Lin, Maya, 41
linearly, **159**
LISP, 45
list, **93**, 92–94
List (Python), 247
list procedures, 94
list-append, 101, 217
list-flatten, 103
list-get-element, 97
list-length, 95
list-map, 99
list-product, 96
list-reverse, 101
list-search, 189
list-sum, 96
Lockhart, Paul, 2
logarithm, **6**
logarithmic growth, 181

Mādhava, 80
machine code, 44
magnetic-core memory, 237
map, 99
McCarthy, John, 45
mcons, 213
means of abstraction, 25, 43

means of combination, 25
measuring input size, 155
messages, 225
methods, **228**
Methods (Python), 250
MIT, 237, 238
mlist, 214
mlist-append, 217
modulo, 70
morpheme, 25
mutable lists, 214
mutable pair, 213
mutators, **205**

name, 51
natural language, **23**
natural languages, 42
newline, 78
Newton, Isaac, 41, 239
Newton, Issac, 14, 72, 123
Nim, 81
noncomputable, 271, **276**
null, **93**
null?, 93
Number, 85
numbers, 47
Nygaard, Kristen, 238

object, **228**
object-oriented programming, **223**
Objects (Python), 250
Olson, Ken, 238
override, **234**

Pólya, George, 61
Paine, Thomas, 157
Pair, 87–91
pair, **87**
parse tree, **32**
parser, **241**
parsing, 252
Pascal's Triangle, 104
Pascal, Blaise, 43, 104, 122
Pascaline, 122
pegboard puzzle, 171, 189
pixel, **11**
pizza, 1
place, **208**
postulate, 272
power set, **168**
precedence, **245**
primitive, 25
primitive expressions, 46
primitive procedures, 47, 48
primitives, 255
Principia Mathematica, **274**
printf, 78
printing, 78
problem, **61**
Procedure, 86
procedure, **2**, 52, 53
programmability, 123
programming language, **43**
proof, **272**
proof by construction, **37**
proposition, **272**
Python, 220

quadratically, **165**

random, 177
recursive definition, 19, 27, 65–77, 116
recursive descent, 254, **254**
recursive grammar, 33
recursive transition network, **26**
reducible, 280
reduction, **280**
relaxation, **200**
repeat-until, **220**
reverse, 101
Roebling, John, 41
rules of evaluation, 58
Russell's paradox, **273**
Russell, Bertrand, 265, 273

scalar, **85**
Scheme, 45
searching, 189
set, 205
set-mcar, 213
set-mcdr, 213
Shakespeare, William, 41
side effects, **79**, 206
Simula, 238
Sketchpad, 238
Smalltalk, 239
sorted binary tree, **182**
sorting, 173–189
special form, **56**
square, 54

square root, 72
stack, **29**
state, 52
Steele, Guy, 45
String, 191
string, **26**, 248
subclass, **230**
substitution, 207
superclass, **230**
surface forms, 23
Survivor, 81
Sussman, Gerald, 45
Sutherland, Ivan, 238
syllogisms, **271**
Symbol, 107
symbol, 226

tagged list, **107**
tail recursive, **71**
thunk, **262**, 263
token, **243**
tokenizer, **243**
Toy Story, 1
tracing, 79
transitive, **174**
tree, 182
truth table, **125**
truthiness, 26
Turing Machine, 136, 157
Turing, Alan, 136, 140
types, 85

universal computers, **3**
universal computing machine, **137**
universal programming language, **117**, **280**
Universal Turing Machine, **279**
universality, 117
URL, 195

virus, 281

web, 195
web crawler, **199**
web-get, 195
well-balanced, **184**
while loop, **219**
Whirlwind, 237
worst case, **157**
Wulf, William, 15