Ben-Gurion University of the Negev
Faculty of Natural Science
Department of Computer Science

Principles of Programming Languages

Mira Balaban

Lecture Notes

May 6, 2017

Many thanks to Tamar Pinhas, Ami Hauptman, Eran Tomer, Barak Bar-Orion, Azzam Maraee, Yaron Gonen, Ehud Barnea, Rotem Mairon, Igal Khitron, Rani Etinger, Ran Anner, Tal Achimeir, Michael Elhadad, Michael Frank for their great help in preparing these notes and the associated code.

Contents

Introduction **1**

1 The Elements of Programming **4**

1.1 The Elements of Programming 4

1.1.1 Expressions (SICP 1.1.1) 5

1.1.2 Abstraction and Reference: Variables and Values (SICP 1.1.2) 7

1.1.3 Evaluation of Scheme Forms (SICP 1.1.3) 8

1.1.4 User Defined Procedures (compound procedures) 9

1.1.5 Conditional Expressions 14

1.2 Procedure Design Using Contracts 18

1.2.1 The Design by Contract (DbC) approach: 20

1.3 Hierarchical Data types: Pairs and Lists 25

1.3.1 The Pair Type 25

1.3.2 Symbol values 27

1.3.3 The List Type 30

1.4 Procedures and the Processes they Generate (SICP 1.2) 38

1.4.1 Linear Recursion and Iteration (SICP 1.2.1) 39

1.4.2 Tree Recursion (SICP 1.2.2) 45

1.4.3 Orders of Growth (SICP 1.2.3) 47

1.5 Higher-Order Procedures 52

1.5.1 Procedures as Parameters (SICP 1.3.1) 53

1.5.2 Constructing procedure arguments at run-time 57

1.5.3 Defining Local Variables – Using the `let` Abbreviation 58

1.5.4 Procedures as Returned Values (SICP 1.3.4) 62

1.5.5 Procedures that Store Delayed (Future) Computations 67

1.6 Polymorphic Procedures 70

2 Theoretical Introduction of Programming Languages: Syntax, Semantics, Types **76**

2.1 Syntax: Concrete and Abstract 76

2.1.1 Concrete Syntax: 77
2.1.2 Abstract Syntax 79
2.2 Operational Semantics: The Substitution Model 83
2.2.1 The Substitution Model – Applicative Order Evaluation: 87
2.2.2 The Substitution Model – Normal Order Evaluation: 94
2.2.3 Comparison: The applicative order and the normal order of evaluations: 96
2.2.4 Local Recursive Procedures 97
2.2.5 Partial evaluation using Function Currying 104
2.2.6 Revisiting: Primitive procedures, User-defined procedures, Derived expressions (Syntactic sugar), Special operators 108
2.2.7 Formalizing tail recursion – analysis of expressions that create iterative processes 112
2.3 Types 114
2.3.1 Atomic Types 115
2.3.2 Composite Types 116
2.3.3 A Type Language for Scheme 121

3 Abstraction on Data 124
3.1 Data Abstraction: Abstract Data Types 125
3.1.1 Example: Binary Trees – Management of Hierarchical Information . . 126
3.1.2 Example: Rational Number Arithmetic (SICP 2.1.1) 137
3.1.3 Example: Church Numbers 146
3.1.4 What is Meant by Data? (SICP 2.1.3) 148
3.2 The Sequence Interface (includes examples from SICP 2.2.3) 154
3.2.1 Mapping over Lists 155
3.2.2 More sequence operations: The Sequence interface as an abstraction barrier 158
3.2.3 Partial evaluation of sequence operations 163
3.2.4 Nested mappings 165

4 Delayed Evaluation on Data and on Control (Behavior) 170
4.1 Lazy Lists (ML Sequences, Scheme Streams) 170
4.1.1 The Lazy List (ML Sequence, Scheme Stream) Abstract Data Type . 172
4.1.2 Integer lazy lists 177
4.1.3 Elementary Lazy List Processing 178
4.1.4 High Order Lazy-List Functions 180
4.2 Continuation Passing Style (CPS) Programming 183
4.2.1 Recursive to Iterative CPS Transformations 187
4.2.2 Controlling Multiple Alternative Future Computations: Errors (Exceptions), Search and Backtracking 192

5 Type Correctness 201
5.1 What is Type Checking/Inference? 202
5.2 The Type Language 205
5.3 Type Substitutions and Unifiers 206
5.4 Static Type Inference for Scheme 209
5.4.1 Typing Statements 209
5.4.2 Static type inference for a restricted Scheme language 212
5.4.3 Adding definitions: 224
5.4.4 Adding control: 227
5.4.5 Adding recursion: 229
5.4.6 Type Correctness with the Pair and List Types 232
5.5 Type Checking and Inference using Type Constraints Approach 235
5.5.1 Creating and solving type constraints 236
5.5.2 Implementing Type Inference Using the Type Constraints Approach . 242

6 Evaluators for Functional Programming 248
6.1 Abstract Syntax Parser (ASP) (SICP 4.1.2) 251
6.1.1 The parser procedures: 253
6.1.2 Derived expressions 259
6.2 A Meta-Circular Evaluator for the Substitution Model – Applicative-Eval Operational Semantics 263
6.2.1 Data Structures package 264
6.2.2 Core Package: Evaluation Rules 271
6.3 The Environment Based Operational Semantics 278
6.3.1 Data Structures 280
6.3.2 The Environment Model Evaluation Algorithm 283
6.3.3 Static (Lexical) and Dynamic Scoping Evaluation Policies 292
6.4 A Meta-Circular Evaluator for the Environment Based Operational Semantics 298
6.4.1 Core Package: Evaluation Rules 298
6.4.2 Data Structures Package 302
6.5 A Meta-Circular Compiler for Functional Programming (SICP 4.1.7) 310
6.5.1 The Analyzer 317

7 Logic Programming - in a Nutshell 329
7.1 Relational Logic Programming 331
7.1.1 Syntax Basics 331
7.1.2 Facts 332
7.1.3 Rules 336
7.1.4 Syntax 342
7.1.5 Operational Semantics 343
7.1.6 Relational logic programs and SQL operations 354

7.2 Logic Programming . . . 355
7.2.1 Syntax . . . 355
7.2.2 Operational semantics . . . 357
7.2.3 Data Structures . . . 360
7.3 Meta-tools: Backtracking optimization (cuts); Unify; Meta-circular interpreters 365
7.3.1 Backtracking optimization – The `cut` operator . . . 365
7.3.2 Unify . . . 370
7.3.3 Meta-circular interpreters for Logic Programming . . . 373
7.4 Prolog . . . 380
7.4.1 Arithmetics . . . 381
7.4.2 Negation in Logic Programming . . . 383

8 Imperative Programming 385
8.1 State Based Modeling . . . 389
8.1.1 Object modeling . . . 390
8.1.2 Procedural implementation of mutable data structures . . . 395
8.1.3 Sameness and sharing – value and identity equality . . . 400
8.2 Operational Semantics: Extending the Environment Model with Local State Management . . . 403
8.3 Extending the Environment Model Interpreter and Compiler . . . 406
8.3.1 Imperative evaluator . . . 406
8.3.2 Imperative compiler . . . 408
8.4 Type checking/Inference with assignments: . . . 409
8.5 Costs of Introducing Mutation . . . 411
8.5.1 Losing Referential Transparency . . . 412
8.5.2 Dependency on Mutation Ordering . . . 413

9 Conclusion: Techniques, Principles and Language Comparison 414

Click here to download full PDF material