

---

# Contents

---

<b>Foreword</b> . . . . .	xi
<b>Preface</b> . . . . .	xiii
<b>Chapter 1. From Hardware to Software</b> . . . . .	1
1.1. Computers: a low-level view . . . . .	1
1.1.1. Information processing . . . . .	1
1.1.2. Memories . . . . .	2
1.1.3. CPUs . . . . .	3
1.1.4. Peripheral devices . . . . .	7
1.2. Computers: a high-level view . . . . .	8
1.2.1. Modeling computations . . . . .	9
1.2.2. High-level languages . . . . .	9
1.2.3. From source code to executable programs . . . . .	10
<b>Chapter 2. Introduction to Semantics of Programming Languages</b> . . . . .	15
2.1. Environment, memory and state . . . . .	16
2.1.1. Evaluation environment . . . . .	16
2.1.2. Memory . . . . .	18
2.1.3. State . . . . .	20
2.2. Evaluation of expressions . . . . .	21
2.2.1. Syntax . . . . .	21
2.2.2. Values . . . . .	22
2.2.3. Evaluation semantics . . . . .	24
2.3. Definition and assignment . . . . .	26
2.3.1. Defining an identifier . . . . .	26
2.3.2. Assignment . . . . .	29
2.4. Exercises . . . . .	31

---

<b>Chapter 3. Semantics of Functional Features</b> . . . . .	35
3.1. Syntactic aspects . . . . .	35
3.1.1. Syntax of a functional kernel . . . . .	35
3.1.2. Abstract syntax tree . . . . .	36
3.1.3. Reasoning by induction over expressions . . . . .	39
3.1.4. Declaration of variables, bound and free variables . . . . .	39
3.2. Execution semantics: evaluation functions . . . . .	42
3.2.1. Evaluation errors . . . . .	42
3.2.2. Values . . . . .	43
3.2.3. Interpretation of operators . . . . .	45
3.2.4. Closures . . . . .	46
3.2.5. Evaluation of expressions . . . . .	47
3.3. Execution semantics: operational semantics . . . . .	54
3.3.1. Simple expressions . . . . .	55
3.3.2. Call-by-value . . . . .	56
3.3.3. Recursive and mutually recursive functions . . . . .	60
3.3.4. Call-by-name . . . . .	61
3.3.5. Call-by-value versus call-by-name . . . . .	62
3.4. Evaluation functions versus evaluation relations . . . . .	64
3.4.1. Status of the evaluation function . . . . .	64
3.4.2. Induction over evaluation trees . . . . .	65
3.5. Semantic properties . . . . .	69
3.5.1. Equivalent expressions . . . . .	69
3.5.2. Equivalent environments . . . . .	71
3.6. Exercises . . . . .	71
<b>Chapter 4. Semantics of Imperative Features</b> . . . . .	77
4.1. Syntax of a kernel of an imperative language . . . . .	77
4.2. Evaluation of expressions . . . . .	81
4.3. Evaluation of definitions . . . . .	86
4.4. Operational semantics . . . . .	89
4.4.1. Big-step semantics . . . . .	89
4.4.2. Small-step semantics . . . . .	93
4.4.3. Expressiveness of operational semantics . . . . .	95
4.5. Semantic properties . . . . .	96
4.5.1. Equivalent programs . . . . .	96
4.5.2. Program termination . . . . .	98
4.5.3. Determinism of program execution . . . . .	100
4.5.4. Big steps versus small steps . . . . .	103
4.6. Procedures . . . . .	109
4.6.1. Blocks . . . . .	109
4.6.2. Procedures . . . . .	112
4.7. Other approaches . . . . .	118

---

4.7.1. Denotational semantics . . . . .	118
4.7.2. Axiomatic semantics, Hoare logic . . . . .	129
4.8. Exercises . . . . .	134
<b>Chapter 5. Types . . . . .</b>	<b>137</b>
5.1. Type checking: when and how? . . . . .	139
5.1.1. When to verify types? . . . . .	139
5.1.2. How to verify types? . . . . .	140
5.2. Informal typing of a program $\mathbf{Exp}_2$ . . . . .	141
5.2.1. A first example . . . . .	141
5.2.2. Typing a conditional expression . . . . .	142
5.2.3. Typing without type constraints . . . . .	142
5.2.4. Polymorphism . . . . .	143
5.3. Typing rules in $\mathbf{Exp}_2$ . . . . .	143
5.3.1. Types, type schemes and typing environments . . . . .	143
5.3.2. Generalization, substitution and instantiation . . . . .	146
5.3.3. Typing rules and typing trees . . . . .	151
5.4. Type inference algorithm in $\mathbf{Exp}_2$ . . . . .	154
5.4.1. Principal type . . . . .	154
5.4.2. Sets of constraints and unification . . . . .	155
5.4.3. Type inference algorithm . . . . .	159
5.5. Properties . . . . .	167
5.5.1. Properties of typechecking . . . . .	167
5.5.2. Properties of the inference algorithm . . . . .	167
5.6. Typechecking of imperative constructs . . . . .	168
5.6.1. Type algebra . . . . .	168
5.6.2. Typing rules . . . . .	169
5.6.3. Typing polymorphic definitions . . . . .	171
5.7. Subtyping and overloading . . . . .	172
5.7.1. Subtyping . . . . .	173
5.7.2. Overloading . . . . .	175
<b>Chapter 6. Data Types . . . . .</b>	<b>179</b>
6.1. Basic types . . . . .	179
6.1.1. Booleans . . . . .	179
6.1.2. Integers . . . . .	181
6.1.3. Characters . . . . .	186
6.1.4. Floating point numbers . . . . .	187
6.2. Arrays . . . . .	191
6.3. Strings . . . . .	194
6.4. Type definitions . . . . .	194
6.4.1. Type abbreviations . . . . .	195
6.4.2. Records . . . . .	196

6.4.3. Enumerated types . . . . .	200
6.4.4. Sum types . . . . .	202
6.5. Generalized conditional . . . . .	205
6.5.1. C style <code>switch/case</code> . . . . .	205
6.5.2. Pattern matching . . . . .	208
6.6. Equality . . . . .	216
6.6.1. Physical equality . . . . .	217
6.6.2. Structural equality . . . . .	218
6.6.3. Equality between functions . . . . .	220
<b>Chapter 7. Pointers and Memory Management . . . . .</b>	<b>223</b>
7.1. Addresses and pointers . . . . .	223
7.2. Endianness . . . . .	225
7.3. Pointers and arrays . . . . .	225
7.4. Passing parameters by address . . . . .	226
7.5. References . . . . .	229
7.5.1. References in C++ . . . . .	229
7.5.2. References in Java . . . . .	233
7.6. Memory management . . . . .	234
7.6.1. Memory allocation . . . . .	234
7.6.2. Freeing memory . . . . .	237
7.6.3. Automatic memory management . . . . .	239
<b>Chapter 8. Exceptions . . . . .</b>	<b>243</b>
8.1. Errors: notification and propagation . . . . .	243
8.1.1. Global variable . . . . .	245
8.1.2. Record definition . . . . .	245
8.1.3. Passing by address . . . . .	245
8.1.4. Introducing exceptions . . . . .	246
8.2. A simple formalization: ML-style exceptions . . . . .	247
8.2.1. Abstract syntax . . . . .	247
8.2.2. Values . . . . .	248
8.2.3. Type algebra . . . . .	248
8.2.4. Operational semantics . . . . .	248
8.2.5. Typing . . . . .	250
8.3. Exceptions in other languages . . . . .	250
8.3.1. Exceptions in OCaml . . . . .	251
8.3.2. Exceptions in Python . . . . .	251
8.3.3. Exceptions in Java . . . . .	253
8.3.4. Exceptions in C++ . . . . .	254

<b>Conclusion</b> .....	257
<b>Appendix: Solutions to the Exercises</b> .....	259
<b>List of Notations</b> .....	287
<b>Index of Programs</b> .....	289
<b>References</b> .....	293
<b>Index</b> .....	295