Contents

I Models of computation **5**

1 Introduction **7**

2 Mathematical preliminaries **9**

2.0.1 Sets . . . 9

2.0.2 Relations and Functions . . . 10

2.0.3 Principle of Mathematical Induction . . . 11

3 A functional model of computation **17**

3.1 The primitive expressions . . . 18

3.2 Substitution of functions . . . 20

3.2.1 Substitution using `let` . . . 21

3.3 Definition of functions using conditionals . . . 23

3.4 Functions as inductively defined computational processes . . . 24

3.5 Recursive processes . . . 26

3.6 Analysis of correctness and efficiency . . . 28

3.6.1 Correctness . . . 28

3.6.2 Efficiency . . . 28

3.6.3 Efficiency, Why and How? . . . 29

3.6.4 In the long run: Asymptotic analysis and Orders of growth . . . 30

3.7 More examples of recursive algorithms . . . 31

3.8 Scope rules . . . 41

3.9 Tail-recursion and iterative processes . . . 43

3.9.1 Correctness of an iterative process . . . 45

3.10 More examples of iterative processes . . . 46

4 The Imperative model of computation **53**

4.1 The primitives for the imperative model . . . 53

4.1.1 Variables and the *assignment* instruction . . . 54

4.1.2 Assertions . . . 56

4.1.3 The *if then else* instruction . . . 57

4.1.4 The *while do* instruction . . . 61

4.1.5 Functions and procedures in the imperative model 63

5 Step-wise refinement and Procedural Abstraction 67

5.1 Step-wise refinement 67

5.1.1 Executable specifications and rapid-prototyping 68

5.1.2 Examples of step-wise refinement 69

5.2 Procedural abstraction using higher-order functions 83

5.2.1 Functions as input parameters 83

5.2.2 Polymorphic functions 86

5.2.3 Constructing functions using `lambda` (λ) 89

5.2.4 Functions as returned values 91