Type Systems for Programming Languages

Benjamin C. Pierce
bcpierce@cis.upenn.edu

Working draft of January 15, 2000

This is preliminary draft of a book in progress. Comments, suggestions, and corrections are welcome.

Contents

Preface **8**

1 Introduction **13**
1.1 What is a Type System? . . . 13
1.2 A Brief History of Type . . . 14
1.3 Applications of Type Systems . . . 16
1.4 Related Reading . . . 16

2 Mathematical Preliminaries **18**
2.1 Sets and Relations . . . 18
2.2 Induction . . . 18
2.3 Term Rewriting . . . 19

3 Untyped Arithmetic Expressions **20**
3.1 Basics . . . 20
Syntax . . . 21
Evaluation . . . 21
3.2 Formalities . . . 21
Syntax . . . 21
Evaluation . . . 25
3.3 Properties . . . 27
3.4 Implementation . . . 27
Syntax . . . 27
Evaluation . . . 28
3.5 Summary . . . 28
3.6 Further Reading . . . 30

4 The Untyped Lambda-Calculus **31**
4.1 Basics . . . 32
Syntax . . . 33
Operational Semantics . . . 34
4.2 Programming in the Lambda-Calculus . . . 34

4.3 Is the Lambda-Calculus a Programming Language? 39
4.4 Formalities 39
Syntax 39
Substitution 40
Operational Semantics 43
Summary 43
4.5 Further Reading 44

5 Implementing the Lambda-Calculus 45
5.1 Nameless Representation of Terms 45
Syntax 46
Shifting and Substitution 48
Evaluation 49
5.2 A Concrete Realization 49
Syntax 50
Shifting and Substitution 50
Evaluation 51
5.3 Ordinary vs. Nameless Representations 51

6 Typed Arithmetic Expressions 52
6.1 Syntax 52
6.2 The Typing Relation 52
6.3 Properties of Typing and Reduction 53
Typing Derivations 53
Typechecking 54
Safety = Preservation + Progress 54
6.4 Implementation 55
6.5 Summary 56

7 Simply Typed Lambda-Calculus 58
7.1 Syntax 58
7.2 The Typing Relation 59
7.3 Summary 61
7.4 Properties of Typing and Reduction 62
Typechecking 62
Typing and Substitution 64
Type Soundness 64
7.5 Implementation 65
7.6 Further Reading 66

8 Extensions **67**
8.1 Base Types . . . 67
8.2 Unit type . . . 67
8.3 Let bindings . . . 68
8.4 Records and Tuples . . . 68
8.5 Variants . . . 72
8.6 General recursion . . . 72
8.7 Lists . . . 73
8.8 Lazy records and let-bindings . . . 75

9 References **76**
9.1 Further Reading . . . 79

10 Exceptions **80**
10.1 Errors . . . 80
10.2 Exceptions . . . 80

11 Type Equivalence **81**

12 Definitions **83**
12.1 Type Definitions . . . 83
12.2 Term Definitions . . . 85

13 Subtyping **86**
13.1 The Subtype Relation . . . 87
Variance . . . 88
Summary . . . 89
13.2 Metatheory of Subtyping . . . 91
Algorithmic Subtyping . . . 91
Minimal Typing . . . 92
13.3 Implementation . . . 96
13.4 Meets and Joins . . . 97
13.5 Primitive Subtyping . . . 99
13.6 The Bottom Type . . . 99
13.7 Other stuff . . . 99

14 Imperative Objects **100**
14.1 Objects . . . 100
14.2 Object Generators . . . 102
14.3 Subtyping . . . 103
14.4 Basic classes . . . 104
14.5 Extending the Internal State . . . 105
14.6 Classes with “Self” . . . 106

15 Recursive Types **109**
15.1 Examples . . . 109
Lists . . . 109
Hungry Functions . . . 109
Recursive Values from Recursive Types . . . 110
Untyped Lambda-Calculus, Redux . . . 110
Recursive Objects . . . 112
15.2 Equi-recursive Types . . . 112
ML Implementation . . . 114
15.3 Iso-recursive Types . . . 115
15.4 Subtyping and Recursive Types . . . 116

16 Case Study: Featherweight Java **117**

17 Type Reconstruction **118**
17.1 Substitution . . . 118
17.2 Universal vs. Existential Type Variables . . . 119
17.3 Constraint-Based Typing . . . 121
17.4 Unification . . . 125
17.5 Principal Typings . . . 128
17.6 Further Reading . . . 129

18 Universal Types **130**
18.1 Motivation . . . 130
18.2 Varieties of Polymorphism . . . 131
18.3 Definitions . . . 132
18.4 Examples . . . 135
Warm-ups . . . 135
Polymorphic Lists . . . 136
Impredicative Encodings . . . 136
18.5 Metatheory . . . 139
Soundness . . . 139
Strong Normalization . . . 139
Erasure and Typeability . . . 140
Type Reconstruction . . . 141
18.6 Implementation . . . 141
Nameless Representation of Types . . . 141
ML Code . . . 141
18.7 Further Reading . . . 143