

# Contents

*Preface*    **xiii**

**1**    *Introduction*    **1**

- 1.1    Types in Computer Science    1
- 1.2    What Type Systems Are Good For    4
- 1.3    Type Systems and Language Design    9
- 1.4    Capsule History    10
- 1.5    Related Reading    12

**2**    *Mathematical Preliminaries*    **15**

- 2.1    Sets, Relations, and Functions    15
- 2.2    Ordered Sets    16
- 2.3    Sequences    18
- 2.4    Induction    19
- 2.5    Background Reading    20

**I**    **Untyped Systems**    **21**

**3**    *Untyped Arithmetic Expressions*    **23**

- 3.1    Introduction    23
- 3.2    Syntax    26
- 3.3    Induction on Terms    29
- 3.4    Semantic Styles    32
- 3.5    Evaluation    34
- 3.6    Notes    43

<b>4</b>	<b><i>An ML Implementation of Arithmetic Expressions</i></b>	<b>45</b>
4.1	Syntax	46
4.2	Evaluation	47
4.3	The Rest of the Story	49
<b>5</b>	<b><i>The Untyped Lambda-Calculus</i></b>	<b>51</b>
5.1	Basics	52
5.2	Programming in the Lambda-Calculus	58
5.3	Formalities	68
5.4	Notes	73
<b>6</b>	<b><i>Nameless Representation of Terms</i></b>	<b>75</b>
6.1	Terms and Contexts	76
6.2	Shifting and Substitution	78
6.3	Evaluation	80
<b>7</b>	<b><i>An ML Implementation of the Lambda-Calculus</i></b>	<b>83</b>
7.1	Terms and Contexts	83
7.2	Shifting and Substitution	85
7.3	Evaluation	87
7.4	Notes	88
<b>II</b>	<b>Simple Types</b>	<b>89</b>
<b>8</b>	<b><i>Typed Arithmetic Expressions</i></b>	<b>91</b>
8.1	Types	91
8.2	The Typing Relation	92
8.3	Safety = Progress + Preservation	95
<b>9</b>	<b><i>Simply Typed Lambda-Calculus</i></b>	<b>99</b>
9.1	Function Types	99
9.2	The Typing Relation	100
9.3	Properties of Typing	104
9.4	The Curry-Howard Correspondence	108
9.5	Erasure and Typability	109
9.6	Curry-Style vs. Church-Style	111
9.7	Notes	111
<b>10</b>	<b><i>An ML Implementation of Simple Types</i></b>	<b>113</b>
10.1	Contexts	113
10.2	Terms and Types	115
10.3	Typechecking	115

<b>11</b>	<b><i>Simple Extensions</i></b>	<b>117</b>
11.1	Base Types	117
11.2	The Unit Type	118
11.3	Derived Forms: Sequencing and Wildcards	119
11.4	Ascription	121
11.5	Let Bindings	124
11.6	Pairs	126
11.7	Tuples	128
11.8	Records	129
11.9	Sums	132
11.10	Variants	136
11.11	General Recursion	142
11.12	Lists	146
<b>12</b>	<b><i>Normalization</i></b>	<b>149</b>
12.1	Normalization for Simple Types	149
12.2	Notes	152
<b>13</b>	<b><i>References</i></b>	<b>153</b>
13.1	Introduction	153
13.2	Typing	159
13.3	Evaluation	159
13.4	Store Typings	162
13.5	Safety	165
13.6	Notes	170
<b>14</b>	<b><i>Exceptions</i></b>	<b>171</b>
14.1	Raising Exceptions	172
14.2	Handling Exceptions	173
14.3	Exceptions Carrying Values	175
<b>III</b>	<b>Subtyping</b>	<b>179</b>
<b>15</b>	<b><i>Subtyping</i></b>	<b>181</b>
15.1	Subsumption	181
15.2	The Subtype Relation	182
15.3	Properties of Subtyping and Typing	188
15.4	The Top and Bottom Types	191
15.5	Subtyping and Other Features	193
15.6	Coercion Semantics for Subtyping	200
15.7	Intersection and Union Types	206
15.8	Notes	207

<b>16</b>	<b><i>Metatheory of Subtyping</i></b>	<b>209</b>
16.1	Algorithmic Subtyping	210
16.2	Algorithmic Typing	213
16.3	Joins and Meets	218
16.4	Algorithmic Typing and the Bottom Type	220
<b>17</b>	<b><i>An ML Implementation of Subtyping</i></b>	<b>221</b>
17.1	Syntax	221
17.2	Subtyping	221
17.3	Typing	222
<b>18</b>	<b><i>Case Study: Imperative Objects</i></b>	<b>225</b>
18.1	What Is Object-Oriented Programming?	225
18.2	Objects	228
18.3	Object Generators	229
18.4	Subtyping	229
18.5	Grouping Instance Variables	230
18.6	Simple Classes	231
18.7	Adding Instance Variables	233
18.8	Calling Superclass Methods	234
18.9	Classes with Self	234
18.10	Open Recursion through Self	235
18.11	Open Recursion and Evaluation Order	237
18.12	A More Efficient Implementation	241
18.13	Recap	244
18.14	Notes	245
<b>19</b>	<b><i>Case Study: Featherweight Java</i></b>	<b>247</b>
19.1	Introduction	247
19.2	Overview	249
19.3	Nominal and Structural Type Systems	251
19.4	Definitions	254
19.5	Properties	261
19.6	Encodings vs. Primitive Objects	262
19.7	Notes	263

<b>IV</b>	<b>Recursive Types</b>	<b>265</b>
20	<i>Recursive Types</i>	267
20.1	Examples	268
20.2	Formalities	275
20.3	Subtyping	279
20.4	Notes	279
21	<i>Metatheory of Recursive Types</i>	281
21.1	Induction and Coinduction	282
21.2	Finite and Infinite Types	284
21.3	Subtyping	286
21.4	A Digression on Transitivity	288
21.5	Membership Checking	290
21.6	More Efficient Algorithms	295
21.7	Regular Trees	298
21.8	$\mu$ -Types	299
21.9	Counting Subexpressions	304
21.10	Digression: An Exponential Algorithm	309
21.11	Subtyping Iso-Recursive Types	311
21.12	Notes	312
<b>V</b>	<b>Polymorphism</b>	<b>315</b>
22	<i>Type Reconstruction</i>	317
22.1	Type Variables and Substitutions	317
22.2	Two Views of Type Variables	319
22.3	Constraint-Based Typing	321
22.4	Unification	326
22.5	Principal Types	329
22.6	Implicit Type Annotations	330
22.7	Let-Polymorphism	331
22.8	Notes	336
23	<i>Universal Types</i>	339
23.1	Motivation	339
23.2	Varieties of Polymorphism	340
23.3	System F	341
23.4	Examples	344
23.5	Basic Properties	353
23.6	Erasure, Typability, and Type Reconstruction	354

23.7	Erasure and Evaluation Order	357
23.8	Fragments of System F	358
23.9	Parametricity	359
23.10	Impredicativity	360
23.11	Notes	361
<b>24</b>	<b><i>Existential Types</i></b>	<b>363</b>
24.1	Motivation	363
24.2	Data Abstraction with Existentials	368
24.3	Encoding Existentials	377
24.4	Notes	379
<b>25</b>	<b><i>An ML Implementation of System F</i></b>	<b>381</b>
25.1	Nameless Representation of Types	381
25.2	Type Shifting and Substitution	382
25.3	Terms	383
25.4	Evaluation	385
25.5	Typing	386
<b>26</b>	<b><i>Bounded Quantification</i></b>	<b>389</b>
26.1	Motivation	389
26.2	Definitions	391
26.3	Examples	396
26.4	Safety	400
26.5	Bounded Existential Types	406
26.6	Notes	408
<b>27</b>	<b><i>Case Study: Imperative Objects, Redux</i></b>	<b>411</b>
<b>28</b>	<b><i>Metatheory of Bounded Quantification</i></b>	<b>417</b>
28.1	Exposure	417
28.2	Minimal Typing	418
28.3	Subtyping in Kernel $F_{<}$	421
28.4	Subtyping in Full $F_{<}$	424
28.5	Undecidability of Full $F_{<}$	427
28.6	Joins and Meets	432
28.7	Bounded Existentials	435
28.8	Bounded Quantification and the Bottom Type	436

<b>VI</b>	<b>Higher-Order Systems</b>	<b>437</b>
<b>29</b>	<b><i>Type Operators and Kinding</i></b>	<b>439</b>
29.1	Intuitions	440
29.2	Definitions	445
<b>30</b>	<b><i>Higher-Order Polymorphism</i></b>	<b>449</b>
30.1	Definitions	449
30.2	Example	450
30.3	Properties	453
30.4	Fragments of $F_\omega$	461
30.5	Going Further: Dependent Types	462
<b>31</b>	<b><i>Higher-Order Subtyping</i></b>	<b>467</b>
31.1	Intuitions	467
31.2	Definitions	469
31.3	Properties	472
31.4	Notes	472
<b>32</b>	<b><i>Case Study: Purely Functional Objects</i></b>	<b>475</b>
32.1	Simple Objects	475
32.2	Subtyping	476
32.3	Bounded Quantification	477
32.4	Interface Types	479
32.5	Sending Messages to Objects	480
32.6	Simple Classes	481
32.7	Polymorphic Update	482
32.8	Adding Instance Variables	485
32.9	Classes with “Self”	486
32.10	Notes	488
	<b>Appendices</b>	<b>491</b>
<b>A</b>	<b><i>Solutions to Selected Exercises</i></b>	<b>493</b>
<b>B</b>	<b><i>Notational Conventions</i></b>	<b>565</b>
B.1	Metavariable Names	565
B.2	Rule Naming Conventions	565
B.3	Naming and Subscripting Conventions	566
	<b>References</b>	<b>567</b>
	<b>Index</b>	<b>605</b>