Mathematics and Computation
7'th edition
Volume 2

Klaus Grue

June 1, 2001

Mathematics and Computation, 7'th edition, Volume 2 of 3
Klaus Grue, DIKU, Universitetsparken 1, DK-2100 Copenhagen, Denmark
grue@diku.dk, http://www.diku.dk/~grue/
©1994–2001 Klaus Grue

ISBN 87-981270-3-9

Contents

1 Syntax **101**

1.1 The Map system . . . 101
1.2 How Map treats formulas . . . 101
1.3 Index . . . 102
1.4 How I read formulas . . . 102
1.5 Computation . . . 103
1.6 Equal signs . . . 104
1.7 Priority . . . 104
1.8 Arity and fixity . . . 105
1.9 Syntax trees . . . 106
1.10 Stepwise formation of term trees . . . 108
1.11 Definitions . . . 109
1.12 Macro definitions . . . 109
1.13 Parentheses . . . 110
1.14 Associativity . . . 111
1.15 Equal priority . . . 112
1.16 Unary minus . . . 112
1.17 Successor and predecessor . . . 113
1.18 Overview of equal signs . . . 114
1.19 Optimising definitions . . . 114
1.20 Terms and statements . . . 115
1.21 Notational freedom . . . 115
1.22 Answers . . . 116

2 Computation **119**

2.1 Reduction . . . 119
2.2 Reduction rules . . . 120
2.3 Truth, falsehood and selection . . . 122
2.4 Numbers . . . 123
2.5 Predicates . . . 123
2.6 Recursive definitions . . . 124
2.7 Bottom [⊥]˙ . . . 126
2.8 Strictness of [x + y]˙ . . . 127
2.9 Strictness of [if(a, b, c)]˙ . . . 128

2.10 Order of computation 129
2.11 Strictness of [x = y]˙ 129
2.12 [x ≡ y]˙ is a directive 130
2.13 Division 131
2.14 Exceptions 131
2.15 Exception catching 133
2.16 Operations on truth values 134
2.17 Answers 135

3 Decimal fractions **137**
3.1 Overview 137
3.2 Infinity 138
3.3 Exact and floating fractions 139
3.4 Predicates 141
3.5 Rounding 141
3.6 Precision 144
3.7 Exact arithmetic operations 144
3.8 Division 146
3.9 Integer division 147
3.10 Power 148
3.11 Mantissa and exponent 149
3.12 Answers 150

4 Truth values **153**
4.1 Logical 'and' 153
4.2 [x ∧ y]˙ 154
4.3 Logical "or" 155
4.4 [x ∨ y]˙ 156
4.5 Negation 157
4.6 Implication and biimplication 158
4.7 Priority and associativity 158
4.8 Associativity of relations 159
4.9 Associativity of implication 160
4.10 Reduction order in forming syntax trees 161
4.11 Commas 162
4.12 Term reduction priority 163
4.13 Answers 164

5 Pairs **165**
5.1 Coordinates 165
5.2 Equality of pairs 166
5.3 Head and tail 166
5.4 Strictness of head and tail 167
5.5 Laziness of [x :: y]˙ 168
5.6 Pairhood 168
5.7 Lists 169

5.8 Representation of lists 171
5.9 Pairs and recursion 172
5.10 Head and tail of lists 173
5.11 Infinite lists 174
5.12 Operations and predicates on pairs 176
5.13 Answers 177

6 Algebra **181**
6.1 Algebraic rules 181
6.2 Replacement 182
6.3 Reverse replacement 185
6.4 Algebraic proofs 185
6.5 Algebraic systems 186
6.6 $[\,\mathsf{Mac}\,]^{\cdot}$ rules 188
6.7 $[\,\mathsf{Mac}\,]^{\cdot}$ lemmas 189
6.8 $[\,\mathsf{Mac}\,]^{\cdot}$ proofs 190
6.9 Use of abbreviations 192
6.10 The $[\,\mathsf{Mac}\,]^{\cdot}$ contradiction 193
6.11 Algebra versus computation 193
6.12 Proofs and syntax trees 194
6.13 Reflexivity 195
6.14 Definitions 196
6.15 Use of lemmas 198
6.16 Computation 199
6.17 Answers 200

7 Variables **203**
7.1 Substitution 203
7.2 How to read substitutions 204
7.3 Only variables can be substituted 205
7.4 Substitution in syntax trees 205
7.5 Algebraic rules for substitution 205
7.6 Substitution and numerals 206
7.7 Substitution and variables 207
7.8 Binding 207
7.9 Free variables 208
7.10 Bound versus binding variables 209
7.11 Renaming of bound variables 209
7.12 No renaming of free variables 211
7.13 Variable clashes 211
7.14 Nested substitution 211
7.15 Binding and nested substitution 212
7.16 Binding diagrams and syntax trees 213
7.17 Substitution inside out 215
7.18 Substitution outside in 215
7.19 Distribution 217

7.20 Automatic renaming 219
7.21 The meaning of free variables 219
7.22 Substitution of equals 219
7.23 Grand substitution 220
7.24 Answers 220

8 Functions 301

8.1 Introduction 301
8.2 Referential transparency 301
8.3 Constructs and referential transparency 303
8.4 Constructs and substitution 304
8.5 $[\lambda x.y]^{\cdot}$ and $[x\,'\,y]^{\cdot}$ 304
8.6 Functions versus constructs 305
8.7 Equality of functions 308
8.8 Use of grand substitution 309
8.9 The alpha- and beta-rule 310
8.10 Recursive functions 311
8.11 Functions that take functions as input 312
8.12 Functions that produce functions as output 312
8.13 Repeated application 313
8.14 Functions that take functions as input and output 314
8.15 Reduction of lambda-terms 314
8.16 Renaming during computations 315
8.17 A more complicated example of renaming 316
8.18 The fixed point operator 317
8.19 Fixed points versus recursion 318
8.20 Answers 319

9 Representation 321

9.1 Bits 321
9.2 Words 322
9.3 Integers 323
9.4 Representation 324
9.5 Maps 325
9.6 Simple pairs 327
9.7 Simple lists 328
9.8 Truth values and exceptions 328
9.9 Weak and strong representations 329
9.10 Decimal fractions 332
9.11 Lists 334
9.12 Summary of representation 334
9.13 The weak representation 334
9.14 Answers 336

10 Sets **339**
10.1 Introduction . . . 339
10.2 Some sets . . . 340
10.3 Classical maps . . . 340
10.4 Relation to the weak representation . . . 341
10.5 Sets of truth values . . . 342
10.6 Sets of decimal fractions . . . 342
10.7 The cartesian product . . . 343
10.8 The empty list . . . 344
10.9 Sets of lists . . . 344
10.10 Infinite lists are non-classical . . . 345
10.11 Answers . . . 346

11 Derivations **349**
11.1 Introduction . . . 349
11.2 Derivation rules . . . 349
11.3 Instantiation statements . . . 350
11.4 Derivation proofs . . . 352
11.5 Labelled proof lines . . . 352
11.6 Indexed variables . . . 353
11.7 Instantiation and binding constructs . . . 354
11.8 Substitution . . . 355
11.9 Terms that represent formulas . . . 355
11.10 Type inference . . . 356
11.11 Numeral types . . . 357
11.12 Floating point type inference . . . 358
11.13 Lemmas with premisses . . . 359
11.14 Use of lemmas . . . 360
11.15 Circular proofs . . . 360
11.16 Subtypes . . . 361
11.17 Sets and representations . . . 361
11.18 The class of sets . . . 362
11.19 Replacement . . . 363
11.20 Definition . . . 363
11.21 Replacement with two premisses . . . 364
11.22 Repetition . . . 364
11.23 Selection . . . 365
11.24 Computation . . . 365
11.25 Derivation systems . . . 366
11.26 Answers . . . 366

12 Deduction **369**
12.1 Introduction . . . 369
12.2 Implication . . . 369
12.3 Implication versus inference . . . 370
12.4 Deduction . . . 372

12.5 Deduction in two levels 372
12.6 References into two blocks 373
12.7 References in and out of blocks 374
12.8 Modus Ponens 375
12.9 Proofs that end inside a block 375
12.10 Hypotheses that do not begin a block 376
12.11 Variable fixation 376
12.12 Answers 377

13 Proof techniques **379**
13.1 Brute force 379
13.2 Backchaining 380
13.3 Backward application 383
13.4 Backward application of deduction 384
13.5 Algebra 387
13.6 Answers 388

14 Logic **391**
14.1 Proof by cases 391
14.2 Tautologies 392
14.3 Answers 395

15 Natural numbers **397**
15.1 The Peano axioms 397
15.2 Induction 398
15.3 Using induction 399
15.4 Analysis of the induction proof 403
15.5 Understanding induction 405
15.6 Answers 406

16 Counterproofs **407**
16.1 Introduction 407
16.2 A useful rule 409
16.3 Counterexamples 410
16.4 Counterexamples to terms 411
16.5 Counterexamples to implications 411
16.6 Answers 412

17 Information **415**
17.1 Properties 415
17.2 Maps 416
17.3 Information contents 417
17.4 Information comparison 417
17.5 Separability 419
17.6 Relation to implication 419
17.7 Properties of $[\, x \preceq y \,]^{\cdot}$ 420

17.8 Monotonicity . . . 421
17.9 Diagrams of [$x \preceq y$] . . . 422
17.10 Finite approximations . . . 423
17.11 Information contents of functions . . . 425
17.12 Minimal fixed points . . . 426
17.13 Uses of minimality . . . 427
17.14 Minimality of definitions . . . 428
17.15 [$x \equiv y$] is a directive . . . 429
17.16 Answers . . . 431

18 Quantifiers **435**
18.1 Universal quantification . . . 435
18.2 The [Gen] rule . . . 436
18.3 The [ElimAll] rule . . . 437
18.4 Existential quantification . . . 437
18.5 The [IntroExists] rule . . . 438
18.6 Local definitions . . . 439
18.7 Choice . . . 439
18.8 The [ElimExists] rules . . . 442
18.9 Type rules for quantification . . . 443
18.10 Equality rules for quantification . . . 443
18.11 Fundamental constructs . . . 443
18.12 Answers . . . 444

A Reference Manual **501**
A.1 Error reporting . . . 501
A.2 Priority . . . 502
A.3 Associativity and term reduction . . . 503
A.4 all x in y colon z [$\forall x \in y : z$] . . . 504
A.5 bottom [$\perp$] . . . 505
A.6 case x comma y comma z end [case(x, y, z)] . . . 506
A.7 choose x in y colon z [$\varepsilon x \in y : z$] . . . 507
A.8 classical [ℓ] . . . 508
A.9 construct x [**construct** x] . . . 509
A.10 empty list [$\langle\rangle$] . . . 510
A.11 exception [$\bullet$] . . . 511
A.12 exists x in y colon z [$\exists x \in y : z$] . . . 512
A.13 false [F] . . . 514
A.14 f four of x end [$f_4(x)$] . . . 515
A.15 f one of x end [$f_1(x)$] . . . 516
A.16 f three of x end [$f_3(x)$] . . . 517
A.17 f two [f_2] . . . 518
A.18 if x then y else z end [if(x, y, z)] . . . 519
A.19 infinity [∞] . . . 521
A.20 lambda x dot y [$\lambda x.y$] . . . 522
A.21 minimum x comma y end [min(x, y)] . . . 523

A.22 minus x [−x] 524
A.23 nil map [N] 525
A.24 not x [¬x] 526
A.25 parenthesis x end [(x)] 527
A.26 plus x [+x] 528
A.27 precision x [#x] 529
A.28 substitute x where y is z end [⟨$\mathcal{A}$ | x:=$\mathcal{B}$⟩] 530
A.29 the class of sets [**Set**] 532
A.30 the set of decimal fractions [**D**] 533
A.31 the set of decimal fractions of precision x end [$\mathbf{D}_{\mathsf{x}}$] 535
A.32 the set of exceptions [**X**] 537
A.33 the set of false maps [**F**] 539
A.34 the set of infinities of precision x end [$\mathbf{D}^{\infty}_{\mathsf{x}}$] 541
A.35 the set of integers [**Z**] 543
A.36 the set of minus infinities of precision x end [$\mathbf{D}^{-\infty}_{\mathsf{x}}$] 545
A.37 the set of natural numbers [**N**] 547
A.38 the set of negative integers [$\mathbf{Z}^{-}$] 549
A.39 the set of positive integers [$\mathbf{Z}^{+}$] 551
A.40 the set of the empty list [**E**] 553
A.41 the set of true maps [**T**] 554
A.42 the set of truth values [**B**] 556
A.43 true [T] 558
A.44 tuple x end [⟨x⟩] 559
A.45 variable x [**variable** x] 560
A.46 x and y [x ∧ y] 561
A.47 x apply y [x ' y] 563
A.48 x associates as y [x $\check{\rightarrow}$ y] 564
A.49 x atom [x atom] 565
A.50 x belongs to y [x ∈ y] 566
A.51 x cartesian y [x × y] 568
A.52 x colon y [x : y] 569
A.53 x comma y [x, y] 570
A.54 x computational equal y [x = y] 571
A.55 x computational unequal y [x ≠ y] 573
A.56 x concludes y [x ⊳ y] 575
A.57 x deduces y [x → y] 576
A.58 x defined equal y [x ≐ y] 577
A.59 x divide y [x/y] 578
A.60 x equal y [x ≡ y] 580
A.61 x exceptional [x?] 582
A.62 x faculty [x!] 583
A.63 x greater priority y [x $\check{>}$ y] 584
A.64 x head [x head] 585
A.65 x if and only if y [x ⇔ y] 586
A.66 x implied by y [x ⇐ y] 588
A.67 x implies y [x ⇒ y] 590

A.68 x infers y [$x \vdash y$]˙ . . . 591
A.69 x integer divide y [$x // y$]˙ . . . 592
A.70 x lemma y colon z . . . 594
A.71 x listset [x^*]˙ . . . 595
A.72 x macro equal y [$x \ddot{=} y$]˙ . . . 596
A.73 x minus y [$x - y$]˙ . . . 597
A.74 x modulo y [$x \% y$]˙ . . . 599
A.75 x modus ponens y [$x \trianglerighteq y$]˙ . . . 601
A.76 x optimised equal y [$x \doteq y$]˙ . . . 602
A.77 x or y [$x \vee y$]˙ . . . 603
A.78 x pair [x pair]˙ . . . 605
A.79 x pair y [$x :: y$]˙ . . . 606
A.80 x plus y [$x + y$]˙ . . . 607
A.81 x power y end [x^y]˙ . . . 609
A.82 x predecessor [x^-]˙ . . . 611
A.83 x proof of y colon z . . . 612
A.84 x reduces to y [$x \stackrel{+}{\rightarrow} y$]˙ . . . 613
A.85 x round y [x@y]˙ . . . 614
A.86 x rule y colon z . . . 616
A.87 x same priority y [$x \stackrel{\smile}{=} y$]˙ . . . 617
A.88 x semicolon y [x; y]˙ . . . 618
A.89 x simple head [x Head]˙ . . . 619
A.90 x simple pair y [$x \therefore y$]˙ . . . 620
A.91 x simple tail [x Tail]˙ . . . 621
A.92 x strongly greater than y [$x > y$]˙ . . . 622
A.93 x strongly less than y [$x < y$]˙ . . . 624
A.94 x successor [x^+]˙ . . . 626
A.95 x tail [x tail]˙ . . . 627
A.96 x term reduces to y [$x \stackrel{\circ}{\rightarrow} y$]˙ . . . 628
A.97 x times y [$x \cdot y$]˙ . . . 630
A.98 x weakly greater than y [$x \geq y$]˙ . . . 632
A.99 x weakly less information y [$x \preceq y$]˙ . . . 634
A.100 x weakly less than y [$x \leq y$]˙ . . . 636

B Map theory **639**
B.1 Introduction . . . 639
B.2 Priority . . . 639
B.3 Associativity . . . 639
B.4 Fundamental constructs . . . 640
B.5 Definitions . . . 640
B.6 Axioms and inferences . . . 641

C Index **643**

D Bibliography **677**

E Summary of syntax **679**
E.1 Associativity and term reduction 679
E.2 Priority 680

Chapter 8

Functions

8.1 Introduction

$[\ f_3(\mathsf{x}) \doteq 2 \cdot \mathsf{x} + 4\ ]$ defines $[\ f_3(\mathsf{x})\ ]^{\cdot}$ as a new construct and assigns the meaning to $[\ f_3(\mathsf{x})\ ]^{\cdot}$ that it reduces to $[\ 2 \cdot \mathsf{x} + 4\ ]^{\cdot}$. In computer science jargon, $[\ f_3(\mathsf{x})\ ]^{\cdot}$ produces an output when it receives an input:

$$[\ f_3(\underset{\text{input}}{\underset{\uparrow}{3}}) \equiv \underset{\text{output}}{\underset{\uparrow}{10}}\ ]^{\cdot}.$$

In this chapter I introduce the notion of a *function*. A function is a "mathematical something" that produces an output when it receives an input just like a construct does. To explain the relation between functions and constructs I have to talk about referential transparency first.

8.2 Referential transparency

Which one of these two statements is true?

- $[\ 2 + 3\ ]^{\cdot}$ is a term.
- $[\ 2 + 3\ ]^{\cdot}$ is five.

Well, the first one is true if you look at $[\ 2 + 3\ ]^{\cdot}$ as three dark shapes on a white background, namely a two-digit, a plus-sign, and a three-digit. The second one is true if you read through the dark shapes and see the meaning behind the shapes. The meaning of $[\ 2 + 3\ ]^{\cdot}$ is "five".

I don't know what a "five-meaning" is, but I suppose it is some sort of event in your or my brain. At least I know that at the time of writing, a meaning is not something one can print a picture of in a book. But I know how to provoke

---

function

a five-meaning in you mind. Unless you read this book for the pleasure of seeing dark shapes on a white background, I just have to put four dark shapes after one another, namely an f-letter, an i-letter, a v-letter, and an e-letter.

If I want to communicate a five-meaning to you and you are willing to receive it, it would be very convenient if the communication could be done along a direct link from my to your brain. As that is not possible at the time of writing, I have to encode the meaning as dark shapes on a white background, and you have to read those shapes and build a five-meaning in your brain.

I will say that the five-meaning is the *semantics* of [2+3]˙. On the contrary, the *syntax* of [2 + 3]˙ comprises three dark shapes on a white background, namely a two-digit, a plus-sign, and a three-digit.

I suppose you don't read this book for the pleasure of seeing dark shapes on a white background. Rather, I suppose you go for the intension of the book. The silent agreement between you and me that you should look through the dark shapes to find the meaning behind them is known as *referential transparency*.

I am aware that you have to do quite a job to read this book. In addition to letting your eyes move across the pages you have to extract the meaning, i.e. to understand what you read. In the end, no-one can help you understand anything. Only you yourself can build a meaning in your own brain. All I can do as a writer is to make the task for you as easy as I can.

In the text, you have to interpret everything outside square brackets semantically. In principle, you have to interpret everything inside square brackets syntactically. So, strictly speaking,

> [2 + 3]˙ is a term

is correct whereas

> [2 + 3]˙ is five

is wrong. The correct formulation of the latter would be

> the value of [2 + 3]˙ is five

or

> the meaning of [2 + 3]˙ is five

However, if I had insisted strictly on syntactic interpretation of everything inside square brackets, this book would have been much heavier than it is. So when I put a term in square brackets it is up to you, dear reader, to find out whether I talk about the term or the meaning of the term.

---

semantics
syntax
referential transparency

8.3 Constructs and referential transparency

Which one of these three statements is true?

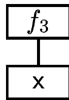
- $[\, f_3(\mathsf{x}) \,]^{\cdot}$ is a term.
- $[\, f_3(\mathsf{x}) \,]^{\cdot}$ is a construct.
- $[\, f_3(\mathsf{x}) \,]^{\cdot}$ is $[\, 2 \cdot \mathsf{x} + 4 \,]^{\cdot}$.

I assume you can see right away by now that, strictly speaking, only the first one is true. The correct formulation of the third statement would be

> the value of $[\, f_3(\mathsf{x}) \,]^{\cdot}$ equals the value of $[\, 2 \cdot \mathsf{x} + 4 \,]^{\cdot}$

The second statement is a bit more complicated. That statement talks about the "f-three-construct" that consists of an f-letter, a three-digit, a left parenthesis, a place to put an argument (an x-letter in this case), and a right parenthesis. In other words I assume you read $[\, f_3(\mathsf{x}) \,]^{\cdot}$ extensionally but disregards $[\, \mathsf{x} \,]^{\cdot}$.

The syntax tree of $[\, f_3(\mathsf{x}) \,]^{\cdot}$ reads:



As you can see, the f-three-construct is a unary operator that happens to be the principal operator of the term $[\, f_3(\mathsf{x}) \,]^{\cdot}$. Therefore, strictly speaking, the second statement could be formulated:

> The principal operator of $[\, f_3(\mathsf{x}) \,]^{\cdot}$ is a construct.

I started Section 8.1 with the following sentence:

> $[\, f_3(\mathsf{x}) \doteq 2 \cdot \mathsf{x} + 4 \,]^{\cdot}$ defines $[\, f_3(\mathsf{x}) \,]^{\cdot}$ as a new construct and $\cdots$

As you can see now, a more correct formulation would be

> the meaning of $[\, f_3(\mathsf{x}) \doteq 2 \cdot \mathsf{x} + 4 \,]^{\cdot}$ defines the principal operator of $[\, f_3(\mathsf{x}) \,]^{\cdot}$ as a new construct and $\cdots$

Clumsy, isn't it?

From now on I will say that

- $[\, f_3(\mathsf{x}) \,]^{\cdot}$ is a term.
- $[\, f_3(\mathsf{x}) \,]^{\cdot}$ is a construct.
- $[\, f_3(\mathsf{x}) \,]^{\cdot}$ is $[\, 2 \cdot \mathsf{x} + 4 \,]^{\cdot}$.

are all correct in each their way, and I will leave it to you, dear reader, to figure out the meaning of each statement.

8.4 Constructs and substitution

As mentioned in Section 8.1, [$f_3(\mathsf{x}) \doteq 2 \cdot \mathsf{x} + 4$] defines [$f_3(\mathsf{x})$] as a new construct and assigns the meaning to [$f_3(\mathsf{x})$] that it reduces to [$2 \cdot \mathsf{x} + 4$]. For all terms [$\mathcal{S}$] you have

$$[\, f_3(\mathcal{S}) \equiv 2 \cdot \mathcal{S} + 4 \,].$$

Stated another way you have

$$[\, f_3(\mathcal{S}) \equiv \langle 2 \cdot \mathsf{x} + 4 \mid \mathsf{x}{:=}\mathcal{S} \rangle \,].$$

In particular you have

$$\begin{array}{lll} [\, f_3(3) & \equiv & \langle 2 \cdot \mathsf{x} + 4 \mid \mathsf{x}{:=}3 \rangle \\ & \equiv & 2 \cdot 3 + 4 \\ & \equiv & 10 \,] \end{array}$$

In general, a definition of form

$$[\, f(\mathsf{x}) \doteq \mathcal{A} \,]$$

gives rise to a rule of form

$$[\, f(\mathcal{S}) \equiv \langle \mathcal{A} \mid \mathsf{x}{:=}\mathcal{S} \rangle \,].$$

8.5 [λx.y] and [x ' y]

In this chapter, I introduce two new constructs, namely

[$\boxed{\lambda \mathsf{x.y}}$], and

[$\boxed{\mathsf{x}\,'\,\mathsf{y}}$].

The *lambda-construct* [λx.y] is a binary prefix construct. Furthermore, [λx.y] is a binding construct whose first argument must be a variable. If [x] is a variable and [$\mathcal{A}$] is a term, then the occurrence of [x] right after lambda in [$\lambda \mathsf{x}.\mathcal{A}$] is a binding occurrence which binds all occurrences of [x] that are free in [$\mathcal{A}$]. Here you have an example of a binding diagram:

$$[\, \lambda \mathsf{x}.\, 2 \cdot \mathsf{x} + 4 \,].$$

[λx.y]	lambda x dot y
[x ' y]	x apply y
	lambda-construct

$[\,\mathsf{x}\,'\,\mathsf{y}\,]^{\cdot}$ is a binary infix construct which is not binding. In $[\,\mathsf{x}\,'\,\mathsf{y}\,]^{\cdot}$, $[\,\mathsf{x}\,]^{\cdot}$ and $[\,\mathsf{y}\,]^{\cdot}$ can be arbitrary terms. Here you have the priority and associativity rules of $[\,\lambda\mathsf{x.y}\,]^{\cdot}$ and $[\,\mathsf{x}\,'\,\mathsf{y}\,]^{\cdot}$:

$$[\,\mathsf{x}\,'\,\mathsf{y}\,'\,\mathsf{z} \breve{\rightarrow} (\mathsf{x}\,'\,\mathsf{y})\,'\,\mathsf{z}\,], \text{ and}$$

$$\left[\mathsf{x}\,'\,\mathsf{y} \mathrel{\breve{>}} \mathsf{x@y} \mathrel{\breve{>}} \mathsf{x}\left\{\begin{array}{l}\mathsf{y}\\ \mathsf{z}\end{array}\right. \mathrel{\breve{>}} \lambda\mathsf{x.y} \mathrel{\breve{>}} \mathsf{x} \equiv \mathsf{y}\right].$$

The constructs $[\,\lambda\mathsf{x.y}\,]^{\cdot}$ and $[\,\mathsf{x}\,'\,\mathsf{y}\,]^{\cdot}$ satisfy the following axiom for all variables $[\,\mathsf{x}\,]^{\cdot}$ and all terms $[\,\mathcal{A}\,]^{\cdot}$ and $[\,\mathcal{S}\,]^{\cdot}$:

$[\,$**Mac rule ApplyLambda** $:\ (\lambda\mathsf{x}.\mathcal{A})\,'\,\mathcal{S} \equiv \langle \mathcal{A} \mid \mathsf{x}{:=}\mathcal{S}\rangle\,]$

Here you have some examples:

$$\begin{array}{lcl}[\,(\lambda\mathsf{x}.\ 2\cdot\mathsf{x}+4)\,'\,3 & \equiv & \langle 2\cdot\mathsf{x}+4 \mid \mathsf{x}{:=}3\rangle\\ & \equiv & 2\cdot 3+4\\ & \equiv & 10\,]^{\cdot}\end{array}$$

$$\begin{array}{lcl}[\,(\lambda\mathsf{x}.\ 2\cdot\mathsf{x}+4)\,'\,\mathsf{x} & \equiv & \langle 2\cdot\mathsf{x}+4 \mid \mathsf{x}{:=}\mathsf{x}\rangle\\ & \equiv & 2\cdot\mathsf{x}+4\,]^{\cdot}\end{array}$$

8.6 Functions versus constructs

Now recall that $[\,f_3(\mathsf{x}) \doteq 2\cdot\mathsf{x}+4\,]^{\cdot}$ introduces $[\,f_3(\mathsf{x})\,]^{\cdot}$ as a new construct which satisfies

$$[\,f_3(\mathcal{S}) \equiv \langle 2\cdot\mathsf{x}+4 \mid \mathsf{x}{:=}\mathcal{S}\rangle\,]^{\cdot}.$$

If you compare this with

$$[\,(\lambda\mathsf{x}.\ 2\cdot\mathsf{x}+4)\,'\,\mathcal{S} \equiv \langle 2\cdot\mathsf{x}+4 \mid \mathsf{x}{:=}\mathcal{S}\rangle\,]^{\cdot}$$

you can see that

$$[\,f_3(\mathcal{S})\,]^{\cdot}$$

and

$$[\,(\lambda\mathsf{x}.\ 2\cdot\mathsf{x}+4)\,'\,\mathcal{S}\,]^{\cdot}$$

are equal to each other.

I now make a new definition:

$$\left[\boxed{f_2} \doteq \lambda\mathsf{x}.\ 2\cdot\mathsf{x}+4\right].$$

$[\,f_2\,]^{\cdot}$ f two

As you can see, $[\, f_2 \,]^{\cdot}$ is a null-ary construct whereas $[\, f_3(\mathsf{x}) \,]^{\cdot}$ is a unary construct. Furthermore, you have

$$
\begin{array}{lcl}
[\, f_2 \,'\, \mathcal{S} & \equiv & (\lambda \mathsf{x}.\, 2 \cdot \mathsf{x} + 4) \,'\, \mathcal{S} \\
 & \equiv & \langle 2 \cdot \mathsf{x} + 4 \mid \mathsf{x}{:=}\mathcal{S} \rangle \\
 & \equiv & 2 \cdot \mathcal{S} + 4 \,]^{\cdot}
\end{array}
$$

In other words:

$$[\, f_2 \,'\, \mathcal{S} \equiv f_3(\mathcal{S}) \,]^{\cdot}.$$

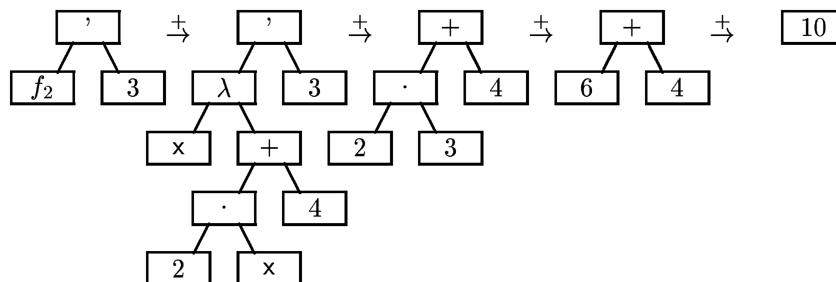
As you can see, $[\, f_2 \,]^{\cdot}$ and $[\, f_3(\mathsf{x}) \,]^{\cdot}$ are very similar. Here you have the definition and main property of $[\, f_2 \,]^{\cdot}$ together with the definition and main property of $[\, f_3(\mathsf{x}) \,]^{\cdot}$:

$$
\begin{array}{llll}
[\, f_2 \doteq & \lambda \mathsf{x}.\, 2 \cdot \mathsf{x} + 4 \,]^{\cdot} & [\, f_2 \,'\, \mathsf{x} \equiv & 2 \cdot \mathsf{x} + 4 \,]^{\cdot} \\
[\, f_3(\mathsf{x}) \doteq & 2 \cdot \mathsf{x} + 4 \,]^{\cdot} & [\, f_3(\mathsf{x}) \equiv & 2 \cdot \mathsf{x} + 4 \,]^{\cdot}
\end{array}
$$

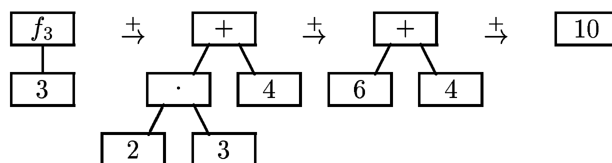
In particular:

$$
\begin{array}{lcl}
[\, f_2 \,'\, 3 & \equiv & 10 \,]^{\cdot} \\
[\, f_3(3) & \equiv & 10 \,]^{\cdot}
\end{array}
$$

Using syntax trees, computation of $[\, f_2 \,'\, 3 \,]^{\cdot}$ proceeds thus:



In contrast, computation of $[\, f_3(3) \,]^{\cdot}$ proceeds thus:



In computer jargon, $[\, f_2 \,]^{\cdot}$ produces an output when it receives an input:

$$[\, f_2 \,'\, 3 \equiv 10 \,]^{\cdot}.$$

input output

Now recall that a *function* is a “mathematical something” that produces an output when it receives an input. $[\, f_2 \,]^{\cdot}$ above is such a mathematical something. Or, said more directly, $[\, f_2 \,]^{\cdot}$ is a function.

Here you have some examples of use of the word “function”:

- $[\, f_2 \,]^{\cdot}$ is the function that multiplies its argument by two and adds four.
- $[\, \lambda \mathsf{x}.\ 2 \cdot \mathsf{x} + 4 \,]^{\cdot}$ is the function that multiplies its argument by two and adds four.
- $[\, \lambda \mathsf{x}.\ 3 \cdot \mathsf{x} + 5 \,]^{\cdot}$ is the function that multiplies its argument by three and adds five.
- If $[\, \mathsf{x} \,]^{\cdot}$ is a variables and $[\, \mathcal{A} \,]^{\cdot}$ is a term then $[\, \lambda \mathsf{x}.\mathcal{A} \,]^{\cdot}$ is a function.
- If $[\, \mathsf{f} \,]^{\cdot}$ is a function and $[\, \mathsf{x} \,]^{\cdot}$ is a mathematical object, then $[\, \mathsf{f}\text{'}\mathsf{x} \,]^{\cdot}$ denotes the output $[\, \mathsf{f} \,]^{\cdot}$ produces when it receives $[\, \mathsf{x} \,]^{\cdot}$ as input.

Here you have four statements about the syntax and semantics of $[\, 2 + 3 \,]^{\cdot}$ and $[\, \lambda \mathsf{x}.\ 2 \cdot \mathsf{x} + 4 \,]^{\cdot}$:

- $[\, 2 + 3 \,]^{\cdot}$ is a term.
- $[\, 2 + 3 \,]^{\cdot}$ is a number.
- $[\, \lambda \mathsf{x}.\ 2 \cdot \mathsf{x} + 4 \,]^{\cdot}$ is a term.
- $[\, \lambda \mathsf{x}.\ 2 \cdot \mathsf{x} + 4 \,]^{\cdot}$ is a function.

Strictly speaking, only the first and third statement is correct. The strictly correct formulation of the other two statements are:

- The value of $[\, 2 + 3 \,]^{\cdot}$ is a number.
- The value of $[\, \lambda \mathsf{x}.\ 2 \cdot \mathsf{x} + 4 \,]^{\cdot}$ is a function.

The value of $[\, 2 + 3 \,]^{\cdot}$ is five and five is a number, so the value of $[\, 2 + 3 \,]^{\cdot}$ is a number. The value of $[\, \lambda \mathsf{x}.\ 2 \cdot \mathsf{x} + 4 \,]^{\cdot}$ is the function that multiplies its argument by two and adds four. In particular, the value of $[\, \lambda \mathsf{x}.\ 2 \cdot \mathsf{x} + 4 \,]^{\cdot}$ is a function.

The example above emphasizes that a function is a value. The construct $[\, f_3(\mathsf{x}) \,]^{\cdot}$ and the function $[\, f_2 \,]^{\cdot}$ are similar in that

$$[\, f_3(\mathsf{x}) \equiv 2 \cdot \mathsf{x} + 4 \,]^{\cdot}$$

and

$$[\, f_2\text{'}\mathsf{x} \equiv 2 \cdot \mathsf{x} + 4 \,]^{\cdot}.$$

However, when I talk about the construct $[\, f_3(\mathsf{x}) \,]^{\cdot}$ I talk about some dark shapes on a white background, and when I talk about the function $[\, f_2 \,]^{\cdot}$ I talk about the meaning behind the shapes.

Here you have some statements about $[\, f_2 \,]^{\cdot}$ and $[\, f_3(\mathsf{x}) \,]^{\cdot}$:

---

function

(1) $[\ f_2\ ]^{\cdot}$ is a construct.

(2) $[\ f_2\ ]^{\cdot}$ is a function.

(3) $[\ f_3(\mathsf{x})\ ]^{\cdot}$ is a construct.

(4) $[\ f_3(\mathsf{x})\ ]^{\cdot}$ is an integer if $[\ \mathsf{x}\ ]^{\cdot}$ is an integer.

(5) $[\ f_2\ \text{'}\ \mathsf{x}\ ]^{\cdot}$ is an integer if $[\ \mathsf{x}\ ]^{\cdot}$ is an integer.

Statement (1) and (3) are correct in the sense that the principal operators of the two terms are constructs. $[\ f_2\ ]^{\cdot}$ is a null-ary construct and $[\ f_3(\mathsf{x})\ ]^{\cdot}$ is a unary construct. Statement (2) is correct in the sense that the value of $[\ f_2\ ]^{\cdot}$ is a function. Statement (4) is correct in the sense that the value of $[\ f_3(\mathsf{x})\ ]^{\cdot}$ is an integer if $[\ \mathsf{x}\ ]^{\cdot}$ is an integer. Statement (5) is correct in the same sense as statement (4).

The point I try to make is that being a function is something semantic, i.e. something that has to do with the meaning or value of a term. In contrast, being a construct is something syntactic.

$[\ f_2\ ]^{\cdot}$ is both a construct and a function, depending on viewpoint. From a syntactic viewpoint, $[\ f_2\ ]^{\cdot}$ is a construct that consists of an f-letter and a 2-digit. From a semantic viewpoint, $[\ f_2\ ]^{\cdot}$ is the function that multiplies its argument by two and adds four.

From a syntactic viewpoint, $[\ f_3(\mathsf{x})\ ]^{\cdot}$ is a construct. From a semantic viewpoint, $[\ f_3(\mathsf{x})\ ]^{\cdot}$ is a value that depends on the value of $[\ \mathsf{x}\ ]^{\cdot}$. $[\ f_3\ ]^{\circ}$ is not a function. Actually, $[\ f_3\ ]^{\circ}$ is syntactically invalid.

Finally, $[\ \lambda\mathsf{x}.\mathcal{A}\ ]^{\cdot}$ is both a construct and a function. From a syntactic point of view, $[\ \lambda\mathsf{x}.\mathcal{A}\ ]^{\cdot}$ is a binary construct. From a semantic point of view, $[\ \lambda\mathsf{x}.\mathcal{A}\ ]^{\cdot}$ is a function. In other words, $[\ \lambda\mathsf{x}.\mathcal{A}\ ]^{\cdot}$ is a construct that takes a variable and a term as arguments and produces a function as its value.

In Section 8.11 you will see that the difference between $[\ f_2\ ]^{\cdot}$ and $[\ f_3(\mathsf{x})\ ]^{\cdot}$ is more than hot air. In that section you will see real differences in what one can do with $[\ f_2\ ]^{\cdot}$ and $[\ f_3(\mathsf{x})\ ]^{\cdot}$.

8.7 Equality of functions

If $[\ \mathsf{f}\ ]^{\cdot}$ and $[\ \mathsf{g}\ ]^{\cdot}$ are functions, then

$$[\ \mathsf{f} \equiv \mathsf{g}\ ]^{\cdot}$$

holds if and only if

$$[\ \mathsf{f}\ \text{'}\ \mathsf{x} \equiv \mathsf{g}\ \text{'}\ \mathsf{x}\ ]^{\cdot}$$

holds (i.e. $[\ \mathsf{f}\ \text{'}\ \mathsf{x} \equiv \mathsf{g}\ \text{'}\ \mathsf{x}\ ]^{\cdot}$ holds for all values of $[\ \mathsf{x}\ ]^{\cdot}$). Here you have an example:

[**Mac lemma L8.7.1** : $(\lambda\mathsf{u}.\ \mathsf{u}+2)\ \text{'}\ \mathsf{x} \equiv (\lambda\mathsf{v}.\ \mathsf{v}+2)\ \text{'}\ \mathsf{x}$]

[**Mac proof of** L8.7.1:

Algebra ▷	$(\lambda u.\ u + 2)\ '\ x$	;
Replace ▷ ApplyLambda ▷	$\langle u + 2 \mid u{:=}x\rangle$	;
Replace ▷ SubXPlusY ▷	$\langle u \mid u{:=}x\rangle + \langle 2 \mid u{:=}x\rangle$	;
Replace ▷ SubXX ▷	$x + \langle 2 \mid u{:=}x\rangle$	;
Replace ▷ SubNumeral ▷	$x + 2$	;
Reverse ▷ SubNumeral ▷	$x + \langle 2 \mid v{:=}x\rangle$	;
Reverse ▷ SubXX ▷	$\langle v \mid v{:=}x\rangle + \langle 2 \mid v{:=}x\rangle$	;
Reverse ▷ SubXPlusY ▷	$\langle v + 2 \mid v{:=}x\rangle$	;
Reverse ▷ ApplyLambda ▷	$(\lambda v.\ v + 2)\ '\ x$	]

From $[\ (\lambda u.\ u + 2)\ '\ x \equiv (\lambda v.\ v + 2)\ '\ x\ ]^{\cdot}$ you have $[\ \lambda u.\ u + 2 \equiv \lambda v.\ v + 2\ ]^{\cdot}$.

$[\ \lambda u.\ u + 2\ ]^{\cdot}$ is the function that adds two to its argument and $[\ \lambda v.\ v + 2\ ]^{\cdot}$ is also the function that adds two to its argument, so it is not surprising that $[\ \lambda u.\ u + 2 \equiv \lambda v.\ v + 2\ ]^{\cdot}$ holds. The binding diagram of $[\ \lambda u.\ u + 2\ ]^{\cdot}$ reads

$$[\ \lambda u.\ u + 2\ ]^{\cdot}$$

and the binding diagram of $[\ \lambda v.\ v + 2\ ]^{\cdot}$ reads

$$[\ \lambda v.\ v + 2\ ]^{\cdot}$$

The anonymous binding diagrams of both $[\ \lambda u.\ u + 2\ ]^{\cdot}$ and $[\ \lambda v.\ v + 2\ ]^{\cdot}$ read

$$[\ \lambda *.\ * + 2\ ]^{\cdot}$$

so $[\ \lambda u.\ u + 2\ ]^{\cdot}$ and $[\ \lambda v.\ v + 2\ ]^{\cdot}$ are structurally equal (i.e. identical except for naming of bound variables).

In general, if two terms are structurally equal, then the terms are equal. The opposite is not true. As an example, $[\ \lambda u.\ \langle u\rangle\ \text{head}\ ]^{\cdot}$ and $[\ \lambda u.u\ ]^{\cdot}$ are structurally different but

$$[\ \langle u\rangle\ \text{head} \equiv u\ ]^{\cdot}$$

gives

$$[\ \lambda u.\ \langle u\rangle\ \text{head} \equiv \lambda u.u\ ]^{\cdot}$$

by substitution of equals.

8.8 Use of grand substitution

The proof of L8.7.1 is rather cumbersome. Here you have a proof that uses $[\ \text{Substitution}\ ]^{\cdot}$:

[**Mac proof of** L8.7.1:
Algebra ▷ $(\lambda u.u + 2)$ ' x ;
Replace ▷ ApplyLambda ▷ $\langle u + 2 \mid u{:=}x \rangle$;
Substitution ▷ $\langle v + 2 \mid v{:=}x \rangle$;
Reverse ▷ ApplyLambda ▷ $(\lambda v.v + 2)$ ' x]

The third proof line is correct because [$\langle u + 2 \mid u{:=}x \rangle$]˙ and [$\langle v + 2 \mid v{:=}x \rangle$]˙ both become [$x + 2$]˙ when performing the substitutions.

8.9 The alpha- and beta-rule

Alonzo Church (American mathematician 1903–1995) once decided to investigate functions systematically. He based his work on older work in which "the function that adds two to its argument" was denoted

[$\hat{x}x{+}2$]°.

Unfortunately, the typesetter who turned Church's manuscript into print was unable to put a hat over a variable, so he put it before the variable:

[$\hat{}xx{+}2$]°.

Another typesetter saw this and considered it a fault. He guessed that the hat [ˆ]° was an ill-printed lambda, so he turned the print into

[$\lambda xx{+}2$]°.

This is where the lambda comes from. Today, Church's theory of functions is known as *lambda calculus*.

In lambda calculus, there are two rules which are known as the *alpha-rule* and the *beta-rule*, respectively. The alpha-rule reads:

If two terms are structurally equal, then they are equal.

The beta-rule reads

[$(\lambda x.\mathcal{A})$ ' $\mathcal{S} \equiv \langle \mathcal{A} \mid x{:=}\mathcal{S} \rangle$]˙.

In the literature you may find many other formulations of these two rules, and you may find versions of lambda calculus that have more rules than these two.

In this book I will use [ApplyLambda]˙ for the beta-rule and [Rename]˙ for the alpha-rule:

[**Mac rule ApplyLambda** : $(\lambda x.\mathcal{A})$ ' $\mathcal{S} \equiv \langle \mathcal{A} \mid x{:=}\mathcal{S} \rangle$]

lambda calculus
alpha-rule
beta-rule

[**Mac rule Rename** : **if** structurally equal$(\mathcal{A}, \mathcal{B})$ **then** $\mathcal{A} \equiv \mathcal{B}$]

I can use [Rename] to give yet another proof of [L8.7.1]. The proof simply says: [$(\lambda u.\ u + 2)$ ' x] and [$(\lambda v.\ v + 2)$ ' x] are structurally equal; hence they are equal. Stated formally this reads:

[**Mac lemma L8.7.1** : $(\lambda u.\ u + 2)$ ' x $\equiv$ $(\lambda v.\ v + 2)$ ' x]

[**Mac proof of** L8.7.1**:**

Algebra $\triangleright$	$(\lambda u.\ u + 2)$ ' x	;
Rename $\triangleright$	$(\lambda v.\ v + 2)$ ' x	]

8.10 Recursive functions

Recall the definition of [n!]:

$$\left[n! \doteq n = 0 \begin{cases} 1 \\ n \cdot (n-1)! \end{cases} \right].$$

The definition defines [n!] as a unary suffix construct. Now consider the following definition:

$$\left[\boxed{\text{fac}} \doteq \lambda n.\ n = 0 \begin{cases} 1 \\ n \cdot \text{fac}\ '\ (n-1) \end{cases} \right].$$

Here you have some examples:

[**Mac lemma L8.10.1** : fac ' $0 \equiv 1$]

[**Mac proof of** L8.10.1**:**

Algebra $\triangleright$	fac ' 0	;
Definition $\triangleright$	$(\lambda n.\ \text{if}(n = 0, 1, n \cdot \text{fac}\ '\ (n-1)))$ ' 0	;
Replace $\triangleright$ ApplyLambda $\triangleright$	$\langle \text{if}(n = 0, 1, n \cdot \text{fac}\ '\ (n-1)) \mid n{:=}0 \rangle$	;
Substitution $\triangleright$	$\text{if}(0 = 0, 1, 0 \cdot \text{fac}\ '\ (0-1))$	;
Replace $\triangleright$ $0 = 0 \equiv \mathsf{T}$ $\triangleright$	$\text{if}(\mathsf{T}, 1, 0 \cdot \text{fac}\ '\ (0-1))$	;
Replace $\triangleright$ $\text{if}(\mathsf{T}, x, y) \equiv x$ $\triangleright$	1	]

[**Mac lemma L8.10.2** : fac ' $1 \equiv 1$]

[**Mac proof of** L8.10.2**:**

Algebra $\triangleright$	fac ' 1	;
Definition $\triangleright$	$(\lambda n.\ \text{if}(n = 0, 1, n \cdot \text{fac}\ '\ (n-1)))$ ' 1	;
Replace $\triangleright$ ApplyLambda $\triangleright$	$\langle \text{if}(n = 0, 1, n \cdot \text{fac}\ '\ (n-1)) \mid n{:=}1 \rangle$	;
Substitution $\triangleright$	$\text{if}(1 = 0, 1, 1 \cdot \text{fac}\ '\ (1-1))$	;
Replace $\triangleright$ $1 = 0 \equiv \mathsf{F}$ $\triangleright$	$\text{if}(\mathsf{F}, 1, 1 \cdot \text{fac}\ '\ (1-1))$	;
Replace $\triangleright$ $\text{if}(\mathsf{F}, x, y) \equiv y$ $\triangleright$	$1 \cdot \text{fac}\ '\ (1-1)$	;
Replace $\triangleright$ $1 - 1 \equiv 0$ $\triangleright$	$1 \cdot \text{fac}\ '\ 0$	;
Replace $\triangleright$ L8.10.1 $\triangleright$	$1 \cdot 1$	;
Replace $\triangleright$ $1 \cdot 1 \equiv 1$ $\triangleright$	1	]

[fac] faculty

You can go on like this and prove [fac'2 ≡ 2]˙, [fac'3 ≡ 6]˙, [fac'4 ≡ 1·2·3·4]˙ and so on.

As you can see, the definition of [fac]˙ above defines [fac]˙ as a null-ary construct whose value is the faculty function. The definition of [fac]˙ is recursive in the sense that [fac]˙ occurs in the right hand side of the definition of [fac]˙.

8.11 Functions that take functions as input

[**Mac lemma L8.11.1** : $(\lambda \mathsf{f}.\ \mathsf{f}\ '\ 3)\ '\ f_2 \equiv 10$]

[**Mac proof of** L8.11.1:

Algebra ▷	$(\lambda \mathsf{f}.\ \mathsf{f}\ '\ 3)\ '\ f_2$	;
Replace ▷ ApplyLambda ▷	$\langle \mathsf{f}\ '\ 3 \mid \mathsf{f}{:=}f_2 \rangle$	;
Substitution ▷	$f_2\ '\ 3$	;
Definition ▷	$(\lambda \mathsf{x}.\ 2 \cdot \mathsf{x} + 4)\ '\ 3$	;
Replace ▷ ApplyLambda ▷	$\langle 2 \cdot \mathsf{x} + 4 \mid \mathsf{x}{:=}3 \rangle$	;
Substitution ▷	$2 \cdot 3 + 4$	;
Replace ▷ $2 \cdot 3 + 4 \equiv 10$ ▷	10	]

As you can see, [$\lambda \mathsf{f}.\ \mathsf{f}\ '\ 3$]˙ is a function that produces the output [10]˙ when it receives the function [f_2]˙ as input.

In general, if [$\lambda \mathsf{f}.\ \mathsf{f}\ '\ 3$]˙ receives a function as input, then it applies the function to [3]˙ and outputs the result of that.

This is the first time you see a real difference between [f_2]˙ and [$f_3(\mathsf{x})$]˙: You can pass the function [f_2]˙ as input to another function. You cannot pass the f-three construct as input to a function. If you try to compute

[$(\lambda \mathsf{f}.\ \mathsf{f}\ '\ 3)\ '\ f_3(\mathsf{x})$]˙

you do not get [$f_3(3)$]˙. Rather, you get [$f_3(\mathsf{x})\ '\ 3$]˙, whatever that means. If you remove the circle superscript in

[$(\lambda \mathsf{f}.\ \mathsf{f}\ '\ 3)\ '\ f_3$]°

then Map would protest and say that the syntax is invalid.

8.12 Functions that produce functions as output

[**Mac lemma L8.12.1** : $(\lambda \mathsf{u}.\lambda \mathsf{x}.\ \mathsf{u} \cdot \mathsf{x} + 4)\ '\ 2 \equiv f_2$]

[**Mac proof of** L8.12.1:

Algebra ▷	$(\lambda \mathsf{u}.\lambda \mathsf{x}.\ \mathsf{u} \cdot \mathsf{x} + 4)\ '\ 2$	;
Replace ▷ ApplyLambda ▷	$\langle \lambda \mathsf{x}.\ \mathsf{u} \cdot \mathsf{x} + 4 \mid \mathsf{u}{:=}2 \rangle$	;
Substitution ▷	$\lambda \mathsf{x}.\ 2 \cdot \mathsf{x} + 4$	;
Definition ▷	f_2	]

As you can see, [λu.λv. u $\cdot$ x + 4] is a function that produces the function [f_2] as output when it receives [2] as input.

Again, you can see a difference between [f_2] and [f_3(x)]: A function like [λu.λx. u $\cdot$ x + 4] can produce [f_2] as output, but no function can produce the f-three construct as output.

In the proof above I have cheated a bit in the second step because I have used [Substitution] to perform a substitution. The problem is that I have not revealed the rules for substituting into a lambda-construct. Here you have the substitution rules for [λx.y] and [x ' y]:

[**Mac rule SubXApplyY** : $\langle \mathcal{A}\ ' \ \mathcal{B} \mid x{:=}\mathcal{S}\rangle \equiv \langle \mathcal{A} \mid x{:=}\mathcal{S}\rangle \ ' \ \langle \mathcal{B} \mid x{:=}\mathcal{S}\rangle$]

[**Mac rule SubLambdaXX** : $\langle \lambda x.\mathcal{A} \mid x{:=}\mathcal{S}\rangle \equiv \lambda x.\mathcal{A}$]

Furthermore, if [x] and [y] are distinct variables and [$\mathcal{S}$] is free for [x] in [λy.$\mathcal{A}$] then [$\langle \lambda y.\mathcal{A} \mid x{:=}\mathcal{S}\rangle \equiv \lambda y.\ \langle \mathcal{A} \mid x{:=}\mathcal{S}\rangle$]:

[**Mac rule SubLambdaXYFreeFor** :
if distinct(x, y) $\wedge$ freefor(λx.$\mathcal{A}$, y, $\mathcal{S}$) **then** $\langle \lambda y.\mathcal{A} \mid x{:=}\mathcal{S}\rangle \equiv \lambda y.\ \langle \mathcal{A} \mid x{:=}\mathcal{S}\rangle$]

A purer algebraic formulation reads

[**Mac rule SubLambdaXY** :
if distinct(x, y) **then** $\langle \lambda y.\mathcal{A} \mid x{:=}\langle \mathcal{S} \mid y{:=}\mathsf{T}\rangle\rangle \equiv \lambda y.\ \langle \mathcal{A} \mid x{:=}\langle \mathcal{S} \mid y{:=}\mathsf{T}\rangle\rangle$]

8.13 Repeated application

[**Mac lemma L8.13.1** : (λu.λx. u $\cdot$ x + 4) ' 2 ' 3 $\equiv$ 10]

[**Mac proof of** L8.13.1**:**

Algebra $\rhd$	(λu.λx. u $\cdot$ x + 4) ' 2 ' 3	;
Replace $\rhd$ ApplyLambda $\rhd$	$\langle \lambda$x. u $\cdot$ x + 4 $\mid$ u:=2$\rangle$ ' 3	;
Substitution $\rhd$	(λx. 2 $\cdot$ x + 4) ' 3	;
Replace $\rhd$ ApplyLambda $\rhd$	$\langle$2 $\cdot$ x + 4 $\mid$ x:=3$\rangle$	;
Substitution $\rhd$	2 $\cdot$ 3 + 4	;
Replace $\rhd$ 2 $\cdot$ 3 + 4 $\equiv$ 10 $\rhd$	10	]

As you saw in the previous section, [λu.λx. u $\cdot$ x + 4] is a function that outputs [f_2] when it inputs [2]. In the example above, [λu.λx. u $\cdot$ x + 4] inputs [2] and outputs [f_2]. After that, [f_2] inputs [3] and outputs [10].

Application of a function to several arguments, one at a time, occurs frequently. This is why I (and most others for the matter) make application left associative:

[f ' x ' y $\breve{\rightarrow}$ (f ' x) ' y].

8.14 Functions that take functions as input and output

Definition 8.14.1 [[twice] $\doteq$ λf.λx. f ' (f ' x)].

Definition 8.14.2 [$g_3 \doteq \lambda$x. x + 3].

Definition 8.14.3 [$g_{33} \doteq \lambda$x. x + 3 + 3]

[**Mac lemma L8.14.4** : twice ' $g_3 \equiv g_{33}$]

[**Mac proof of** L8.14.4**:**

Algebra ▷	twice ' g_3	;
Definition ▷	(λf.λx. f ' (f ' x)) ' g_3	;
Replace ▷ ApplyLambda ▷	⟨λx. f ' (f ' x) | f:=g_3⟩	;
Substitution ▷	λx. g_3 ' (g_3 ' x)	;
Definition ▷	λx. g_3 ' ((λx. x + 3) ' x)	;
Replace ▷ ApplyLambda ▷	λx. g_3 ' ⟨x + 3 | x:=x⟩	;
Substitution ▷	λx. g_3 ' (x + 3)	;
Definition ▷	λx. (λx. x + 3) ' (x + 3)	;
Replace ▷ ApplyLambda ▷	λx. ⟨x + 3 | x:=x + 3⟩	;
Substitution ▷	λx. x + 3 + 3	;
Definition ▷	g_{33}	]

As you can see, [twice]˙ is a function that produces the function [g_{33}]˙ as output when it receives the function [g_3]˙ as input.

Exercise 8.14.5 Compute [twice ' f_2 ' 3]˙.

Exercise 8.14.6 (a) Give a counterexample to [x + 3 + 3 ≡ x + 6]˙, i.e. find a value for [x]˙ for which [x + 3 + 3 ≡ x + 6]˙ fails.

(b) Let [$g_6 \doteq \lambda$x. x + 6]˙. Does [$g_{33} \equiv g_6$]˙ hold or fail?

(c) Does [twice ' $g_3 \equiv g_6$]˙ hold or fail?

8.15 Reduction of lambda-terms

If [x]˙ is a variable, if [$\mathcal{A}$]˙ and [$\mathcal{S}$]˙ are terms, and if [$\mathcal{A}'$]˙ is the result of performing the substitution [⟨$\mathcal{A}$ | x:=$\mathcal{S}$⟩]˙, then Map considers

$$\left[(\lambda x.\mathcal{A}) \text{ ' } \mathcal{S} \stackrel{+}{\to} \mathcal{A}' \right]^{\cdot}$$

as a reduction rule. As an example, Map regards

$$\left[(\lambda x.\ x + 3) \text{ ' } 2 \stackrel{+}{\to} 2 + 3 \right]^{\cdot}$$

[twice]˙ twice

as a reduction. The literature calls such a reduction a *beta-reduction*.

Here you can see how Map computes $(\lambda u.\lambda x.\ u \cdot x + 4)\ '\ 2\ '\ 3$:

$$
\begin{aligned}
(\lambda u.\lambda x.\ u \cdot x + 4)\ '\ 2\ '\ 3 &\overset{+}{\to} (\lambda x.\ 2 \cdot x + 4)\ '\ 3 \\
&\overset{+}{\to} 2 \cdot 3 + 4 \\
&\overset{+}{\to} 6 + 4 \\
&\overset{+}{\to} 10
\end{aligned}
$$

As you may recall, any reduction sequence can be turned into an algebraic proof. See the proof of Lemma [L8.13.1] for an algebraic proof that corresponds to the above reduction sequence.

Here you can see how Map computes $(\lambda u.\lambda x.\ u \cdot x + 4)\ '\ 2$:

$$(\lambda u.\lambda x.\ u \cdot x + 4)\ '\ 2 \overset{+}{\to} \lambda x.\ 2 \cdot x + 4.$$

As you can see, the result of the computation is $\lambda x.\ 2 \cdot x + 4$. This is an example of a computation whose result is a function. If you have a feeling that the computation is unfinished because it results in a function, then you have to adjust you feelings. The value of $\lambda u.\lambda x.\ u \cdot x + 4)\ '\ 2$ is $\lambda x.\ 2 \cdot x + 4$, and no further reduction is possible.

8.16 Renaming during computations

If you need to compute the value of a term, you may need to rename bound variables during the computation. I will give you a simple and a complicated example of that. Here you have the simple one:

$$
\begin{aligned}
(\lambda x.\lambda y.\ x + y)\ '\ (y + 1) &\overset{+}{\to} (\lambda x.\lambda z.\ x + z)\ '\ (y + 1) \\
&\overset{+}{\to} \lambda z.\ y + 1 + z
\end{aligned}
$$

In the example above, I rename the bound variable y into a *fresh variable* z. A variable is "fresh" if it does not occur in the term before the renaming.

Without the renaming, the computation proceeds thus:

$$(\lambda x.\lambda y.\ x + y)\ '\ (y + 1) \overset{+}{\to} \lambda y.\ y + 1 + y.$$

The computation above yields the result $\lambda y.\ y + 1 + y$ which is wrong. The beta-reduction

$$(\lambda x.\mathcal{A})\ '\ \mathcal{S} \overset{+}{\to} \mathcal{A}'$$

is only valid when $\mathcal{S}$ is free for x in $\mathcal{A}$. If $\mathcal{S}$ is not free for x in $\mathcal{A}$ you have to rename bound variables before you can perform the beta-reduction.

beta-reduction
fresh variable

The literature refers to renaming of bound variables during a computation as an *alpha-reduction* or *alpha-renaming*.

Strictly speaking, alpha-renaming is more general than alpha-reduction. An alpha-renaming is an alpha-reduction if the number of different variable names in the term increases. An alpha-renaming that results in the same or a smaller number of different variable names is not an alpha-reduction. As an example,

$$\left[\, (\lambda x.x) \text{'}\, 2 + (\lambda x.x) \text{'}\, 3 \stackrel{+}{\rightarrow} (\lambda x.x) \text{'}\, 2 + (\lambda y.y) \text{'}\, 3 \,\right]^{\cdot}$$

is an alpha-reduction because the number of different variable names increases from one to two.

8.17 A more complicated example of renaming

Recall that a term is "simple" if different variables in it have different names. As an example, $[\, (\lambda x.x) \text{'}\, 2 + (\lambda y.y) \text{'}\, 3 \,]^{\cdot}$ is simple and $[\, (\lambda x.x) \text{'}\, 2 + (\lambda x.x) \text{'}\, 3 \,]^{\cdot}$ is not. As another example,

$$[\, (\lambda x.\ x \text{'}\, x) \text{'}\, (\lambda y.\lambda z.\ y \text{'}\, z) \,]^{\cdot}$$

is simple, so you may find it surprising that you have to rename bound variables to compute its value. Here you have the computation:

$$\begin{array}{ll}
[\, (\lambda x.\ x \text{'}\, x) \text{'}\, (\lambda y.\lambda z.\ y \text{'}\, z) & \stackrel{+}{\rightarrow} \\
(\lambda y.\lambda z.\ y \text{'}\, z) \text{'}\, (\lambda y.\lambda z.\ y \text{'}\, z) & \stackrel{+}{\rightarrow} \\
\lambda z.\ (\lambda y.\lambda z.\ y \text{'}\, z) \text{'}\, z & \stackrel{+}{\rightarrow} \\
\lambda z.\ (\lambda y.\lambda u.\ y \text{'}\, u) \text{'}\, z & \stackrel{+}{\rightarrow} \\
\lambda z.\lambda u.\ z \text{'}\, u \,]^{\cdot} &
\end{array}$$

The computation above has four steps, namely a beta-, beta-, alpha-, and beta-reduction, respectively. Without the alpha-reduction you get a faulty computation:

$$\begin{array}{ll}
[\, (\lambda x.\ x \text{'}\, x) \text{'}\, (\lambda y.\lambda z.\ y \text{'}\, z) & \stackrel{+}{\rightarrow} \\
(\lambda y.\lambda z.\ y \text{'}\, z) \text{'}\, (\lambda y.\lambda z.\ y \text{'}\, z) & \stackrel{+}{\rightarrow} \\
\lambda z.\ (\lambda y.\lambda z.\ y \text{'}\, z) \text{'}\, z & \stackrel{+}{\rightarrow} \\
\lambda z.\lambda z.\ z \text{'}\, z \,]^{\cdot} &
\end{array}$$

The following exercise is nice in that it has a simple formulation and a simple result which is completely obvious once you are through some computations. I have heard one person say that any computer science student should solve the exercise once in a lifetime. I have heard another say that it is the exercise that turns boys into men (leaving possible effects on girls unspecified). Solving the exercise merely requires about $[\, 10 \,]^{\cdot}$ reduction steps but, admittedly, there are some possibilities of loosing track of parentheses and names of bound variables.

---

alpha-reduction
alpha-renaming

Exercise 8.17.1 Compute [twice ’ twice]˙.

If you have forgot:

$$[\ \mathsf{twice} \doteq \lambda \mathsf{f}.\lambda \mathsf{x}.\ \mathsf{f}\ \text{'}\ (\mathsf{f}\ \text{'}\ \mathsf{x})\ ].$$

8.18 The fixed point operator

Definition 8.18.1 $\left[\ \boxed{\mathsf{Y}} \doteq \lambda \mathsf{f}.\ (\lambda \mathsf{x}.\ \mathsf{f}\ \text{'}\ (\mathsf{x}\ \text{'}\ \mathsf{x}))\ \text{'}\ (\lambda \mathsf{x}.\ \mathsf{f}\ \text{'}\ (\mathsf{x}\ \text{'}\ \mathsf{x}))\ \right].$

[**Mac lemma L8.18.2** : Y ’ f ≡ f ’ (Y ’ f)]

[**Mac proof of** L8.18.2**:**

Algebra ▷	Y ’ f	;
Definition ▷	(λf. (λx. f ’(x’x)) ’ (λx. f ’(x’x))) ’ f	;
Replace ▷ ApplyLambda ▷	⟨(λx. f ’(x’x)) ’ (λx. f ’(x’x)) | f:=f⟩	;
Substitution ▷	(λx. f ’(x’x)) ’ (λx. f ’(x’x))	;
Rename ▷	(λy. f ’(y’y)) ’ (λx. f ’(x’x))	;
Replace ▷ ApplyLambda ▷	⟨f ’(y’y) | y:=λx. f ’(x’x)⟩	;
Substitution ▷	f ’ ((λx. f ’(x’x)) ’ (λx. f ’(x’x)))	;
Substitution ▷	f ’ ⟨(λx. f ’(x’x)) ’ (λx. f ’(x’x)) | f:=f⟩	;
Reverse ▷ ApplyLambda ▷	f ’ ((λf. (λx. f ’(x’x)) ’ (λx. f ’(x’x))) ’ f)	;
Definition ▷	f ’ (Y ’ f)	]

If [f]˙ is a function then the lemma above says that the equation

[f ’ x ≡ x]˙

has at least one solution, namely

[x ≡ Y ’ f]˙.

I think it is a surprising result that the equation [f ’ x ≡ x]˙ has a solution for **any** function [f]˙. To illustrate the result, I will consider a function [f]˙ for which [f ’ x ≡ x]˙ has no solutions in classical mathematics, namely [λx. x + 1]˙. According to the result above, the equation

[(λx. x + 1) ’ x ≡ x]˙

has at least one solution. In other words, the equation

[x + 1 ≡ x]˙

has at least one solution. If you search a bit, you can find infinitely many solutions. Here you have four ones:

[⊥ + 1 ≡ ⊥]˙,

[Y]˙ fixpoint

$[\, \bullet + 1 \equiv \bullet \,]^{\cdot}$,

$[\, 1\emptyset\mathrm{F} + 1 \equiv 1\emptyset\mathrm{F} \,]^{\cdot}$, and

$[\, 2\emptyset\mathrm{F} + 1 \equiv 2\emptyset\mathrm{F} \,]^{\cdot}$.

Now that you have seen that

$[\, \mathsf{x} + 1 \equiv \mathsf{x} \,]^{\cdot}$

has lots of solutions, it may be interesting to ask: Which one of the solutions does $[\, \mathsf{Y}\, '\, \lambda \mathsf{x}.\ \mathsf{x} + 1 \,]^{\cdot}$ provide? If you try to compute $[\, \mathsf{Y}\, '\, \lambda \mathsf{x}.\ \mathsf{x} + 1 \,]^{\cdot}$ you will soon see that the computation never yields an answer. In other words:

$[\, \mathsf{Y}\, '\, \lambda \mathsf{x}.\ \mathsf{x} + 1 \equiv \bot \,]^{\cdot}$.

As you can see, $[\, \mathsf{x}{+}1 \equiv \mathsf{x} \,]^{\cdot}$ has lots of solutions and $[\, \mathsf{Y}'\lambda \mathsf{x}.\ \mathsf{x}{+}1 \,]^{\cdot}$ is a particular one of them. In Chapter 17 I state a general rule about which particular one $[\, \mathsf{Y} \,]^{\cdot}$ yields.

The function $[\, \mathsf{Y} \,]^{\cdot}$ is known as the "fixed point operator". To explain why, I will talk a little about geometry.

If you take a plane and rotate it e.g. $[\, 45 \,]^{\cdot}$ degrees, then the points in the plane will move to new positions. However, exactly one point will stay where it is, namely the center of the rotation. For that reason, the center of the rotation is called a *fixed point*. If you take the mirror image of a plane around a mirror axis, then all points that are away from the axis will move to the other side of the axis whereas all points on the axis will stay where they are, so all points on the axis are fixed points. In general, if $[\, \mathsf{f} \,]^{\cdot}$ is a function from points to points in a plane, then any point $[\, \mathsf{x} \,]^{\cdot}$ for which $[\, \mathsf{f}\, '\, \mathsf{x} \equiv \mathsf{x} \,]^{\cdot}$ is a fixed point.

The operator $[\, \mathsf{Y} \,]^{\cdot}$ is able to solve equations of form $[\, \mathsf{f}'\mathsf{x} \equiv \mathsf{x} \,]^{\cdot}$, and therefore it is called the "fixed point operator" even if $[\, \mathsf{f} \,]^{\cdot}$ is not a function from points to points.

8.19 Fixed points versus recursion

Definition 8.19.1 $\left[\ \mathrm{Fac} \doteq \lambda \mathsf{f}.\lambda \mathsf{n}.\ \mathsf{n} = 0 \begin{cases} 1 \\ \mathsf{n} \cdot \mathsf{f}\, '\, (\mathsf{n} - 1) \end{cases} \right]^{\cdot}$

[**Mac lemma L8.19.2** : Fac ' fac ≡ fac]

[**Mac proof of** L8.19.2**:**

Algebra ⊳	Fac ' fac	;
Definition ⊳	$(\lambda \mathsf{f}.\lambda \mathsf{n}.\ \mathrm{if}(\mathsf{n}{=}0, 1, \mathsf{n} \cdot \mathsf{f}\, '\, (\mathsf{n} - 1)))\, '\, \mathrm{fac}$	;
Replace ⊳ ApplyLambda ⊳	$\langle \lambda \mathsf{n}.\ \mathrm{if}(\mathsf{n}{=}0, 1, \mathsf{n} \cdot \mathsf{f}\, '\, (\mathsf{n} - 1)) \mid \mathsf{f}{:=}\mathrm{fac} \rangle$	;
Substitution ⊳	$\lambda \mathsf{n}.\ \mathrm{if}(\mathsf{n}{=}0, 1, \mathsf{n} \cdot \mathrm{fac}\, '\, (\mathsf{n} - 1))$	;
Definition ⊳	fac	]

fixed point

As you can see, $[\text{fac}]^{\cdot}$ is a solution to the equation

$$[\,\text{Fac}\,'\,\mathsf{x} \equiv \mathsf{x}\,]^{\cdot}.$$

In other words, $[\text{fac}]^{\cdot}$ is a fixed point of $[\text{Fac}]^{\cdot}$. Actually, $[\text{fac}]^{\cdot}$ is exactly the fixed point that $[\mathsf{Y}]^{\cdot}$ yields:

$$[\,\text{fac} \equiv \mathsf{Y}\,'\,\text{Fac}\,]^{\cdot}.$$

I now define

$$\left[\text{fac}_2 \doteq \mathsf{Y}\,'\,\lambda\mathsf{f}.\lambda\mathsf{n}.\ \mathsf{n} = 0 \begin{cases} 1 \\ \mathsf{n} \cdot \mathsf{f}\,'\,(\mathsf{n} - 1) \end{cases}\right]^{\cdot}.$$

You have $[\,\text{fac}_2 \equiv \mathsf{Y}\,'\,\text{Fac} \equiv \text{fac}\,]^{\cdot}$, so $[\,\text{fac}_2\,]^{\cdot}$ is the same function as $[\text{fac}]^{\cdot}$. In other words, the following two definitions define the same function:

$$\left[\text{fac} \doteq \lambda\mathsf{n}.\ \mathsf{n} = 0 \begin{cases} 1 \\ \mathsf{n} \cdot \text{fac}\,'\,(\mathsf{n} - 1) \end{cases}\right]^{\cdot}, \text{ and}$$

$$\left[\text{fac}_2 \doteq \mathsf{Y}\,'\,\lambda\mathsf{f}.\lambda\mathsf{n}.\ \mathsf{n} = 0 \begin{cases} 1 \\ \mathsf{n} \cdot \mathsf{f}\,'\,(\mathsf{n} - 1) \end{cases}\right]^{\cdot}.$$

The first definition is recursive because the defined concept occurs on the right hand side. The second definition is non-recursive but defines the same function.

If you don't like recursive definitions (many mathematicians don't), then you can always do without them because you can reformulate the definitions using $[\mathsf{Y}]^{\cdot}$.

8.20 Answers

Answer 8.14.5

$$\begin{aligned}[\,\text{twice}\,'\,f_2\,'\,3 &\equiv f_2\,'\,(f_2\,'\,3) \\ &\equiv f_2\,'\,(2 \cdot 3 + 4) \\ &\equiv f_2\,'\,10 \\ &\equiv 2 \cdot 10 + 4 \\ &\equiv 24\,]^{\cdot}\end{aligned}$$

Answer 8.14.6 (a) $[\,\mathsf{x} \equiv 1\emptyset\mathrm{F}\,]^{\cdot}$ is a counterexample since $[\,1\emptyset\mathrm{F} + 6 \equiv 2\emptyset\mathrm{F}\,]^{\cdot}$ whereas $[\,1\emptyset\mathrm{F} + 3 + 3 \equiv 1\emptyset\mathrm{F}\,]^{\cdot}$. (b) $[\,g_{33} \equiv g_6\,]^{\cdot}$ fails because $[\,g_{33}\,'\,1\emptyset\mathrm{F} \equiv g_6\,'\,1\emptyset\mathrm{F}\,]^{\cdot}$ fails. Two functions $[\mathsf{f}]^{\cdot}$ and $[\mathsf{g}]^{\cdot}$ are different if $[\,\mathsf{f}\,'\,\mathsf{x} \equiv \mathsf{g}\,'\,\mathsf{x}\,]^{\cdot}$ fails for one or more values of $[\mathsf{x}]^{\cdot}$, so it is enough to give one example of an $[\mathsf{x}]^{\cdot}$ for which $[\,\mathsf{f}\,'\,\mathsf{x} \equiv \mathsf{g}\,'\,\mathsf{x}\,]^{\cdot}$ fails to see that $[\,\mathsf{f} \equiv \mathsf{g}\,]^{\cdot}$ fails. (c) $[\,\text{twice}\,'\,g_3 \equiv g_{33}\,]^{\cdot}$ holds and $[\,g_{33} \equiv g_6\,]^{\cdot}$ fails so $[\,\text{twice}\,'\,g_3 \equiv g_6\,]^{\cdot}$ fails.

Answer 8.17.1 I state the reduction in the form af an algebraic proof of $[\,\text{twice}\,'\,\text{twice} \equiv \lambda\mathsf{g}.\lambda\mathsf{x}.\ \mathsf{g}\,'\,(\mathsf{g}\,'\,(\mathsf{g}\,'\,(\mathsf{g}\,'\,\mathsf{x})))\,]^{\cdot}$:

[**Mac lemma L8.20.1** : twice ' twice ≡ λg.λx. g ' (g ' (g ' (g ' x)))]

[**Mac proof of** L8.20.1**:**

Algebra ▷	twice ' twice	;
Definition ▷	(λf.λx. f '(f 'x)) ' twice	;
Replace ▷ ApplyLambda ▷	⟨λx. f '(f 'x) | f:=twice⟩	;
Substitution ▷	λx. twice ' (twice ' x)	;
Rename ▷	λg. twice ' (twice ' g)	;
Definition ▷	λg. twice ' ((λf.λx. f '(f 'x)) ' g)	;
Replace ▷ ApplyLambda ▷	λg. twice ' ⟨λx. f '(f 'x) | f:=g⟩	;
Substitution ▷	λg. twice ' (λx. g'(g'x))	;
Rename ▷	λg. twice ' (λy. g'(g'y))	;
Definition ▷	λg. (λf.λx. f '(f 'x)) ' (λy. g'(g'y))	;
Replace ▷ ApplyLambda ▷	λg. ⟨λx. f '(f 'x) | f:=λy. g'(g'y)⟩	;
Substitution ▷	λg.λx. (λy. g'(g'y)) ' ((λy. g'(g'y)) ' x)	;
Replace ▷ ApplyLambda ▷	λg.λx. (λy. g'(g'y)) ' ⟨g'(g'y) | y:=x⟩	;
Substitution ▷	λg.λx. (λy. g'(g'y)) ' (g'(g'x))	;
Replace ▷ ApplyLambda ▷	λg.λx. ⟨g'(g'y) | y:=g'(g'x)⟩	;
Substitution ▷	λg.λx. g ' (g ' (g ' (g ' x)))	]

Chapter 9

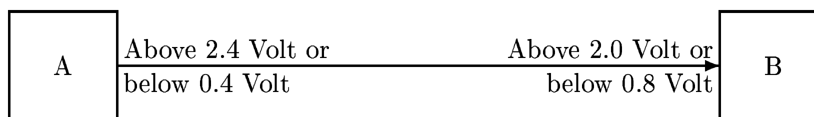
Representation

9.1 Bits

The simplest kind of data a computer can work with is the truth value. As mentioned earlier, there are two truth values, [T]˙ and [F]˙.

When dealing with computer hardware, it is customary to use the symbols [1]˙ and [0]˙ instead of [T]˙ and [F]˙, respectively. The symbols [1]˙ and [0]˙ are known as *binary terms*, and it is customary to abbreviate "binary term" as *bit*.

In a physical computer, bits are represented by physical quantities. As an example, a computer that uses the transistor-transistor logic (TTL) conventions represents bits by voltages: A TTL device that inputs a bit regards all voltages below [0.8]˙ Volt as binary [0]˙ and all voltages above [2.0]˙ Volt as binary [1]˙. A TTL device that outputs a bit represents a binary [0]˙ by a voltage below [0.4]˙ Volt and a binary [1]˙ by a voltage above [2.4]˙ Volt. If you connect an output from one TTL device [*A*]° to an input of another TTL device [*B*]°, and if [*A*]° outputs either a binary [0]˙ or a binary [1]˙, then [*A*]° will either output a voltage below [0.4]˙ Volt or a voltage above [2.4]˙ Volt. The device [*B*]° will safely detect these voltages as binary [0]˙ and [1]˙, respectively.



The TTL conventions are old fashioned, but I mention them here for illustration because they are so simple.

---

binary term
bit

9.2 Words

It is very limited what a computer can do with just one bit. As an example, consider the letters $[\, a, b, c, \ldots \,]^{\circ}$ of the alphabet. There are $[\, 26 \,]^{\cdot}$ letters in the English alphabet and only two bit values $[\, 0 \,]^{\cdot}$ and $[\, 1 \,]^{\cdot}$, so there is no way to represent letters by a single bit.

The American Standard Code for Information Interchange (ASCII) specifies a way in which any letter of the English alphabet can be represented by a list of exactly seven bits. Here is a sample of the ASCII representation:

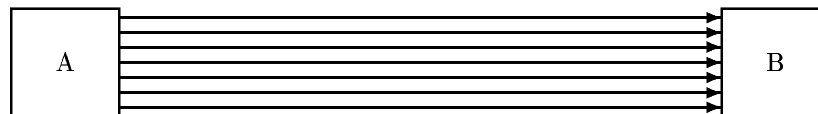
character	code
a	$\langle 1,1,0,0,0,0,1 \rangle$
b	$\langle 1,1,0,0,0,1,0 \rangle$
c	$\langle 1,1,0,0,0,1,1 \rangle$
$\vdots$	$\vdots$
z	$\langle 1,1,1,1,0,1,0 \rangle$

ASCII also specifies how to represent capital letters and various other characters. Here is a sample:

character	code
@	$\langle 1,0,0,0,0,0,0 \rangle$
A	$\langle 1,0,0,0,0,0,1 \rangle$
B	$\langle 1,0,0,0,0,1,0 \rangle$
C	$\langle 1,0,0,0,0,1,1 \rangle$
$\vdots$	$\vdots$
Z	$\langle 1,0,1,1,0,1,0 \rangle$
0	$\langle 0,1,1,0,0,0,0 \rangle$
1	$\langle 0,1,1,0,0,0,1 \rangle$
2	$\langle 0,1,1,0,0,1,0 \rangle$
$\vdots$	$\vdots$
9	$\langle 0,1,1,1,0,0,1 \rangle$

Again, ASCII is old fashioned, but I use it here for illustration because it is so simple. In computer jargon, a finite list of bits is known as a *word*. As an example, ASCII specifies that the letter $[\, a \,]^{\circ}$ is represented by the seven bit word $[\, \langle 1,1,0,0,0,0,1 \rangle \,]^{\cdot}$.

If one TTL device $[\, A \,]^{\circ}$ needs to transmit a character to another TTL device $[\, B \,]^{\circ}$, then it can do so using seven signal lines that carry one bit each:



In computer jargon, a collection of signal wires is known as a *bus*. As an example, in the illustration above there is a seven bit bus from $[\, A \,]^{\circ}$ to $[\, B \,]^{\circ}$, and that seven bit bus carries a seven bit word from $[\, A \,]^{\circ}$ to $[\, B \,]^{\circ}$.

word
bus

9.3 Integers

Bits and words are popular in hardware design because they are easy to work with in hardware. However, programmers also want to work with data structures that are more difficult to work with in hardware. One such structure is the integer.

An [n]-bit word can have [2^n] different values and there are infinitely many integers, so there is no way to represent integers by [n]-bit words.

Hardware designers solve that problem by putting a restriction on the size of integers. As an example, there are [65536] integers from [-32768] to [32767] (inclusive), and [$65536 = 2^{16}$], so it is possible to represent integers in the range from [-32768] to [32767] by [16]-bit words. Integers in the range from [-32768] to [32767] are often called [16]-bit signed integers or just [16]-bit integers. The hardware designer may then design a computer that can compute with [16]-bit integers. Such a machine is typically called a [16]-bit machine.

That solution is unacceptable for programmers. Programmers occasionally need numbers larger than [16]-bit integers. The programmer may then buy hardware that uses more bits per integer, e.g. [32]- and [64]-bit machines, but the fundamental problem remains: An [n]-bit machine can merely compute with numbers of a limited size.

Fortunately, if the hardware of a computer can compute with [n]-bit integers for some [n], then it is possible to program it to compute with integers of any size. When doing so, a large integer may take up more memory than a small one. An integer with one million digits will typically take up around a half to two megabytes depending on implementation. Computer integers that are larger than the integers of the underlying hardware are sometimes referred to as *big numbers* or *bignums*.

If a computer works with big numbers, then it may run out of memory if the numbers become too large. A typical present day computer would just report a memory overflow and halt the computation. However, Alan Turing (English mathematician) proposed a computer architecture which is now known as the *Turing machine*. The architecture of the Turing machine is such that there is no limit on how much memory can be attached to it (as opposed e.g. to the four gigabyte limit of [32]-bit machines and the sixteen billion gigabyte limit of [64]-bit machines). Furthermore, the Turing machine is such that one may add more memory to it while it is working. If a Turing machine runs out of memory, then it just pauses and waits for the user to add more memory.

A Turing machine has what is known as a *potentially infinite* memory. This means that at any time, its memory is finite, but there is no limit on how large it can grow during a computation. From the point of view of the Turing machine, the memory is infinite since it is always extended to meet the needs

---

big number
bignum
Turing machine
potentially infinite

of the machine.

Turing machines are popular in mathematical computer science because one does not have to care about memory overflows. Theories that take memory overflow into account are so complicated and so machine dependent that they have no applications. When studying an algorithm in mathematical computer science, it is tacitly assumed that the algorithm runs on a Turing machine or similar where memory overflow is no issue. Afterwards, when the algorithm is used in practice, it is typical to conduct experiments (known as *tests*) to ensure that the algorithm can run within the given amount of memory on the given hardware.

Theories about exactly when a memory overflow occurs are complicated, but there is a branch of mathematical computer science known as complexity theory that deals with upper and lower bounds of memory requirements and computational speed. Such upper and lower bounds are often enough to decide whether or not a given algorithm can run on a given piece of hardware.

In this book I assume that algorithms run on Turing-like machines and that integers can have any size.

9.4 Representation

A computer can deal with many kinds of data such as numbers, characters, images, and so on. Regardless of this diversity, however, it is customary to deal with only one kind of data internally in the computer, namely words.

In other terms, it is customary to use only one kind of data, namely words to represent as diverse data structures as numbers, characters and images. As an example, one may decide to represent the letter $[\, B \,]^\circ$ by the $[\, 7 \,]^\cdot$-bit word $[\, \langle 1,0,0,0,0,1,0\rangle \,]^\cdot$ as you saw in Section 9.2. This of course does not mean that the letter $[\, B \,]^\circ$ really is the word $[\, \langle 1,0,0,0,0,1,0\rangle \,]^\cdot$. It just means that one chooses for convenience to represent that particular letter by that particular word.

Computers can work with numbers because computers can work with words and words can represent numbers. Computers can work with images because computers can work with words and words can represent images. A computer is general because it can work with words and words can represent lots of different things.

In Chapter 6 you saw an axiomatic system for proving lemmas about lists. I could go on and give you axiomatic systems for proving lemmas about numbers, functions, truth values, exceptions, and so on.

In the following, however, I will introduce the notion of a *map*. I will also introduce an axiom system which I call *map theory*. Map theory allows to prove lemmas about maps.

---

test
map
map theory

Map theory can prove lemmas about numbers because map theory can prove lemmas about maps and maps can represent numbers. Map theory can prove lemmas about truth values because map theory can prove lemmas about maps and maps can represent truth values. Map theory is general because it can prove lemmas about maps and maps can represent lots of different things.

Maps can represent more things than words. Otherwise, I would stick to words and present “word theory” instead of “map theory”. Here you have an example where maps are more general than words: Maps can represent decimal fractions as well as real numbers. Words can merely represent decimal fractions. A decimal fraction has finitely many digits which is why it can fit into a word. A real number like $[\,\pi\,]^{\cdot}$ has infinitely many digits and cannot fit into a word. Map theory can prove lemmas about real numbers, “word theory” cannot.

Maps can represent almost anything mathematicians have ever thought of such as numbers, functions, sets, cardinals, manifolds, integrals, choice operators and so on. At first, maps may look obscure and useless, but just wait and see.

9.5 Maps

Mathematicians more or less use “map” as a synonym for “function”. You saw functions in Chapter 8. However, I use “map” to denote a little more than “function”. Let me present some maps for you, one at a time.

The first map I want to present is an old friend:

$$[\,\bot\,]^{\cdot}.$$

This distinguished map represents total absense of information as you already know.

The next map I want to present is the *nil map*. You already know a number that represents “nothing”, namely $[\,0\,]^{\cdot}$. You also know a list with zero elements, namely $[\,\langle\rangle\,]^{\cdot}$. The nil map is a cousin of $[\,0\,]^{\cdot}$ and $[\,\langle\rangle\,]^{\cdot}$. $[\,0\,]^{\cdot}$ is the “nothing” in the world of numbers, $[\,\langle\rangle\,]^{\cdot}$ is the “nothing” in the world of lists, and the nil map is the “nothing” in the world of maps. The nil map also has a name:

$$\left[\,\boxed{\mathsf{N}}\,\right]^{\cdot}.$$

It is now time to present the three first properties of map theory:

$[\,\bot \equiv \bot\,]^{\cdot}$ holds,

$[\,\mathsf{N} \equiv \mathsf{N}\,]^{\cdot}$ holds, and

$[\,\bot \equiv \mathsf{N}\,]^{\cdot}$ fails.

There are infinitely many maps, so I cannot present all of them one at a time. Fortunately, all remaining maps are functions, so I can present all maps by saying: A map is $[\,\bot\,]^{\cdot}$ or $[\,\mathsf{N}\,]^{\cdot}$ or a function.

---

nil map

$[\,\mathsf{N}\,]^{\cdot}$ nil map

Here you have a simple version of map theory:

Constructs	Axioms			Contra-diction
$[\, \mathsf{N} \,]^{\cdot}$	$[\, \mathsf{N}\ \text{'}\ \mathcal{B}$	$\equiv$	$\mathsf{N} \,]^{\cdot}$	$[\, \bot \equiv \mathsf{N} \,]^{\cdot}$
$[\, \lambda \mathsf{x}.\mathcal{A} \,]^{\cdot}$	$[\, (\lambda \mathsf{x}.\mathcal{A})\ \text{'}\ \mathcal{B}$	$\equiv$	$\langle \mathcal{A} \mid \mathsf{x}{:=}\mathcal{B} \rangle \,]^{\cdot}$	
$[\, \bot \,]^{\cdot}$	$[\, \bot\ \text{'}\ \mathcal{B}$	$\equiv$	$\bot \,]^{\cdot}$	
$[\, \mathcal{A}\ \text{'}\ \mathcal{B} \,]^{\cdot}$	$[\, \mathrm{case}(\mathsf{N}, \mathcal{B}, \mathcal{C})$	$\equiv$	$\mathcal{B} \,]^{\cdot}$	
$[\, \mathrm{case}(\mathcal{A}, \mathcal{B}, \mathcal{C}) \,]^{\cdot}$	$[\, \mathrm{case}(\lambda \mathsf{x}.\mathcal{A}, \mathcal{B}, \mathcal{C})$	$\equiv$	$\mathcal{C} \,]^{\cdot}$	
$[\, \langle \mathcal{A} \mid \mathsf{x}{:=}\mathcal{B} \rangle \,]^{\cdot}$	$[\, \mathrm{case}(\bot, \mathcal{B}, \mathcal{C})$	$\equiv$	$\bot \,]^{\cdot}$	
$[\, \mathsf{x} \,]^{\cdot}$	$[\, \text{Substitution} \,]^{\cdot}$			

Above, $[\, \mathcal{A} \,]^{\cdot}$, $[\, \mathcal{B} \,]^{\cdot}$, $[\, \mathcal{C} \,]^{\cdot}$, and $[\, \mathsf{x} \,]^{\cdot}$ denote terms with the restriction that $[\, \mathsf{x} \,]^{\cdot}$ must be a variable.

The first column says that the terms of map theory are made up from the constructs $[\, \mathsf{N} \,]^{\cdot}$, $[\, \lambda \mathsf{x}.\mathcal{A} \,]^{\cdot}$, $[\, \bot \,]^{\cdot}$, $[\, \mathcal{A}\ \text{'}\ \mathcal{B} \,]^{\cdot}$, $[\, \mathrm{case}(\mathcal{A}, \mathcal{B}, \mathcal{C}) \,]^{\cdot}$, $[\, \langle \mathcal{A} \mid \mathsf{x}{:=}\mathcal{B} \rangle \,]^{\cdot}$, and variables. The construct $[\, \mathrm{case}(\mathcal{A}, \mathcal{B}, \mathcal{C}) \,]^{\cdot}$ is one you haven't seen before. I have put $[\, \mathsf{x} \,]^{\cdot}$ in the list of constructs to emphasize that terms of map theory may contain variables.

The second column lists seven axioms. $[\, \text{Substitution} \,]^{\cdot}$ allows to perform substitution and to rename bound variables.

Map can compute with maps and nothing else. Map pretends that it can compute with numbers, truth values, exceptions, and so on, but that is because I represent numbers, truth values, exceptions and so on by maps.

The construct

$$\left[\, \boxed{\mathrm{case}(\mathsf{x}, \mathsf{y}, \mathsf{z})} \,\right]^{\cdot}$$

is a new construct. Map computes $[\, \mathrm{case}(\mathsf{x}, \mathsf{y}, \mathsf{z}) \,]^{\cdot}$ thus: First Map computes the value of $[\, \mathsf{x} \,]^{\cdot}$. Then Map does one of the following three things: (1) If $[\, \mathsf{x} \,]^{\cdot}$ is the nil map, then Map computes $[\, \mathsf{y} \,]^{\cdot}$, and the value of $[\, \mathsf{y} \,]^{\cdot}$ becomes the value of $[\, \mathrm{case}(\mathsf{x}, \mathsf{y}, \mathsf{z}) \,]^{\cdot}$. (2) If $[\, \mathsf{x} \,]^{\cdot}$ is a function, then Map computes the value of $[\, \mathsf{z} \,]^{\cdot}$, and the value of $[\, \mathsf{z} \,]^{\cdot}$ becomes the value of $[\, \mathrm{case}(\mathsf{x}, \mathsf{y}, \mathsf{z}) \,]^{\cdot}$. (3) If Map never succeeds to compute a value for $[\, \mathsf{x} \,]^{\cdot}$, then $[\, \mathsf{x} \equiv \bot \,]^{\cdot}$ and $[\, \mathrm{case}(\mathsf{x}, \mathsf{y}, \mathsf{z}) \equiv \bot \,]^{\cdot}$. Map can only compute with maps, so there are no other possibilities than these three.

$[\, \mathrm{case}(\mathsf{x}, \mathsf{y}, \mathsf{z}) \,]^{\cdot}$ resembles $[\, \mathrm{if}(\mathsf{x}, \mathsf{y}, \mathsf{z}) \,]^{\cdot}$ in that is selects one out of two possibilities. Previously, I introduced

$$\left[\, \mathsf{x} \left\{ \begin{array}{l} \mathsf{y} \\ \mathsf{z} \end{array} \right. \right]^{\cdot}$$

$[\, \mathrm{case}(\mathsf{x}, \mathsf{y}, \mathsf{z}) \,]^{\cdot}$ case x comma y comma z end

as shorthand for $[\,\mathrm{if}(x,y,z)\,]^{\cdot}$. I now introduce a similar shorthand notation for $[\,\mathrm{case}(x,y,z)\,]^{\cdot}$:

$$\left[\ \boxed{x\langle\begin{smallmatrix}y\\z\end{smallmatrix}} \doteq \mathrm{case}(x,y,z)\ \right].$$

Full map theory has more constructs and more axioms than those listed above.

9.6 Simple pairs

Definition 9.6.1 $\left[\ \boxed{x \therefore y} \doteq \lambda z.\ \mathrm{case}(z,x,y)\ \right]$.

Definition 9.6.2 $\left[\ \boxed{x\ \mathrm{Head}} \doteq x\ '\ \mathsf{N}\ \right]$.

Definition 9.6.3 $\left[\ \boxed{x\ \mathrm{Tail}} \doteq x\ '\ \lambda z.\mathsf{N}\ \right]$.

Priority 9.6.4 $[\ x \therefore y \stackrel{\vee}{=} x :: y\ ]$.

Priority 9.6.5 $[\ x\ \mathrm{Head} \stackrel{\vee}{=} x\ \mathrm{Tail} \stackrel{\vee}{=} x\ \mathrm{head}\ ]$.

[**Mac lemma L9.6.6** : $(x \therefore y)\ \mathrm{Head} \equiv x$]

[**Mac proof of** L9.6.6**:**

Algebra ▷	$(x \therefore y)\ \mathrm{Head}$	;
Definition ▷	$(x \therefore y)\ '\ \mathsf{N}$	;
Definition ▷	$(\lambda z.\ \mathrm{case}(z,x,y))\ '\ \mathsf{N}$	;
Replace ▷ ApplyLambda ▷	$\langle \mathrm{case}(z,x,y) \mid z{:=}\mathsf{N}\rangle$	;
Substitution ▷	$\mathrm{case}(\mathsf{N},x,y)$	;
Replace ▷ $\mathrm{case}(\mathsf{N},x,y) \equiv x$ ▷	x	]

[**Mac lemma L9.6.7** : $(x \therefore y)\ \mathrm{Tail} \equiv y$]

[**Mac proof of** L9.6.7**:**

Algebra ▷	$(x \therefore y)\ \mathrm{Tail}$	;
Definition ▷	$(x \therefore y)\ '\ \lambda z.\mathsf{N}$	;
Definition ▷	$(\lambda z.\ \mathrm{case}(z,x,y))\ '\ \lambda z.\mathsf{N}$	;
Replace ▷ ApplyLambda ▷	$\langle \mathrm{case}(z,x,y) \mid z{:=}\lambda z.\mathsf{N}\rangle$	;
Substitution ▷	$\mathrm{case}(\lambda z.\mathsf{N},x,y)$	;
Replace ▷ $\mathrm{case}(\lambda x.\mathcal{A},x,y) \equiv y$ ▷	y	]

$\left[\ x\langle\begin{smallmatrix}y\\z\end{smallmatrix}\ \right]^{\cdot}$	x case y else z end
$[\ x \therefore y\ ]^{\cdot}$	x simple pair y
$[\ x\ \mathrm{Head}\ ]^{\cdot}$	x simple head
$[\ x\ \mathrm{Tail}\ ]^{\cdot}$	x simple tail

As you can see, $[\,x \therefore y\,]^{\cdot}$, $[\,x\ \text{Head}\,]^{\cdot}$, and $[\,x\ \text{Tail}\,]^{\cdot}$ behave like $[\,x :: y\,]^{\cdot}$, $[\,x\ \text{head}\,]^{\cdot}$ and $[\,x\ \text{tail}\,]^{\cdot}$. I am going to represent all constructs of mathematics by constructs defined from $[\,\mathsf{N}\,]^{\cdot}$, $[\,\lambda x.\mathcal{A}\,]^{\cdot}$, $[\,\mathcal{A}\,\text{'}\,\mathcal{B}\,]^{\cdot}$, and $[\,\text{case}(\mathcal{A},\mathcal{B},\mathcal{C})\,]^{\cdot}$ and a few more, so I could choose to represent $[\,x :: y\,]^{\cdot}$ by $[\,x \therefore y\,]^{\cdot}$, $[\,x\ \text{head}\,]^{\cdot}$ by $[\,x\ \text{Head}\,]^{\cdot}$, and $[\,x\ \text{tail}\,]^{\cdot}$ by $[\,x\ \text{Tail}\,]^{\cdot}$. I choose, however, to use $[\,x \therefore y\,]^{\cdot}$, $[\,x\ \text{Head}\,]^{\cdot}$, and $[\,x\ \text{Tail}\,]^{\cdot}$ in a slightly different way which allows me to represent a lot of things with $[\,x \therefore y\,]^{\cdot}$, $[\,x\ \text{Head}\,]^{\cdot}$, and $[\,x\ \text{Tail}\,]^{\cdot}$. I show that in the following.

9.7 Simple lists

I use terms like

$$[\,\langle a, b, c, d, e\rangle^{\cdot}\,]^{\cdot}$$

to stand for

$$[\,a \therefore b \therefore c \therefore d \therefore e \therefore \mathsf{N}\,]^{\cdot}.$$

The way I do this is completely analogous to what I did in Section 5.8:

$$\left[\,\langle x, y\rangle^{\cdot} \overset{\circ}{\rightarrow} x \therefore \langle y\rangle^{\cdot}\,\right],$$

$$\left[\,\boxed{\langle x\rangle^{\cdot}} \ddot{=} x \therefore \langle\,\rangle^{\cdot}\,\right], \text{ and}$$

$$[\,\langle\,\rangle^{\cdot} \ddot{=} \mathsf{N}\,].$$

9.8 Truth values and exceptions

I choose to represent $[\,\mathsf{T}\,]^{\cdot}$, $[\,\mathsf{F}\,]^{\cdot}$, and $[\,\bullet\,]^{\cdot}$ thus:

Definition 9.8.1 $\left[\,\boxed{\mathsf{T}} \doteq \mathsf{N}\,\right]$.

Definition 9.8.2 $\left[\,\boxed{\mathsf{F}} \doteq \mathsf{N} \therefore \mathsf{N}\,\right]$.

Definition 9.8.3 $[\,\boxed{\bullet} \doteq (\mathsf{N} \therefore \mathsf{N}) \therefore (\mathsf{N} \therefore \mathsf{N})\,]$.

These definitions allow me to define $[\,\text{if}(x, y, z)\,]^{\cdot}$:

Definition 9.8.4 $\left[\,\boxed{\text{if}(x, y, z)} \doteq \text{case}(x, y, \text{case}(x\ \text{Head}, z, \bullet))\,\right]$.

$[\,\langle x\rangle^{\cdot}\,]^{\cdot}$	simple tuple x end
$[\,\mathsf{T}\,]^{\cdot}$	true
$[\,\mathsf{F}\,]^{\cdot}$	false
$[\,\bullet\,]^{\cdot}$	exception
$[\,\text{if}(x, y, z)\,]^{\cdot}$	if x then y else z end

[**Mac lemma L9.8.5** : $\mathsf{T}\left\{\begin{array}{l}\mathsf{y}\\ \mathsf{z}\end{array}\right. \equiv \mathsf{y}$]

[**Mac proof of** L9.8.5**:**
Algebra $\rhd$ $\quad$ $\mathrm{if}(\mathsf{T},\mathsf{y},\mathsf{z})$;
Definition $\rhd$ $\quad$ $\mathrm{case}(\mathsf{T},\mathsf{y},\mathrm{case}(\mathsf{T}\ \mathrm{Head},\mathsf{y},\bullet))$;
Definition $\rhd$ $\quad$ $\mathrm{case}(\mathsf{N},\mathsf{y},\mathrm{case}(\mathsf{T}\ \mathrm{Head},\mathsf{y},\bullet))$;
Replace $\rhd$ $\mathrm{case}(\mathsf{N},\mathsf{y},\mathsf{z}) \equiv \mathsf{y}$ $\rhd$ $\quad$ y]

[**Mac lemma L9.8.6** : $\mathrm{case}(\mathsf{u} \therefore \mathsf{v},\mathsf{x},\mathsf{y}) \equiv \mathsf{y}$]

[**Mac proof of** L9.8.6**:**
Algebra $\rhd$ $\quad$ $\mathrm{case}(\mathsf{u} \therefore \mathsf{v},\mathsf{x},\mathsf{y})$;
Definition $\rhd$ $\quad$ $\mathrm{case}(\lambda \mathsf{z}.\ \mathrm{case}(\mathsf{z},\mathsf{u},\mathsf{v}),\mathsf{x},\mathsf{y})$;
Replace $\rhd$ $\mathrm{case}(\lambda \mathsf{x}.\mathcal{A},\mathsf{y},\mathsf{z}) \equiv \mathsf{z}$ $\rhd$ $\quad$ y]

[**Mac lemma L9.8.7** : $\mathsf{F}\left\{\begin{array}{l}\mathsf{y}\\ \mathsf{z}\end{array}\right. \equiv \mathsf{z}$]

[**Mac proof of** L9.8.7**:**
Algebra $\rhd$ $\quad$ $\mathrm{if}(\mathsf{F},\mathsf{y},\mathsf{z})$;
Definition $\rhd$ $\quad$ $\mathrm{case}(\mathsf{F},\mathsf{y},\mathrm{case}(\mathsf{F}\ \mathrm{Head},\mathsf{z},\bullet))$;
Definition $\rhd$ $\quad$ $\mathrm{case}(\mathsf{N} \therefore \mathsf{N},\mathsf{y},\mathrm{case}(\mathsf{F}\ \mathrm{Head},\mathsf{z},\bullet))$;
Replace $\rhd$ L9.8.6 $\rhd$ $\quad$ $\mathrm{case}(\mathsf{F}\ \mathrm{Head},\mathsf{z},\bullet)$;
Definition $\rhd$ $\quad$ $\mathrm{case}((\mathsf{N} \therefore \mathsf{N})\ \mathrm{Head},\mathsf{z},\bullet)$;
Replace $\rhd$ L9.6.6 $\rhd$ $\quad$ $\mathrm{case}(\mathsf{N},\mathsf{z},\bullet)$;
Replace $\rhd$ $\mathrm{case}(\mathsf{N},\mathsf{x},\mathsf{y}) \equiv \mathsf{x}$ $\rhd$ $\quad$ z]

[**Mac lemma L9.8.8** : $\bullet\left\{\begin{array}{l}\mathsf{y}\\ \mathsf{z}\end{array}\right. \equiv \bullet$]

[**Mac proof of** L9.8.8**:**
Algebra $\rhd$ $\quad$ $\mathrm{if}(\bullet,\mathsf{y},\mathsf{z})$;
Definition $\rhd$ $\quad$ $\mathrm{case}(\bullet,\mathsf{y},\mathrm{case}(\bullet\ \mathrm{Head},\mathsf{z},\bullet))$;
Definition $\rhd$ $\quad$ $\mathrm{case}((\mathsf{N}\therefore\mathsf{N})\therefore(\mathsf{N}\therefore\mathsf{N}),\mathsf{y},\mathrm{case}(\bullet\ \mathrm{Head},\mathsf{z},\bullet))$;
Replace $\rhd$ L9.8.6 $\rhd$ $\quad$ $\mathrm{case}(\bullet\ \mathrm{Head},\mathsf{z},\bullet)$;
Definition $\rhd$ $\quad$ $\mathrm{case}(((\mathsf{N} \therefore \mathsf{N}) \therefore (\mathsf{N}\therefore\mathsf{N}))\ \mathrm{Head},\mathsf{z},\bullet)$;
Replace $\rhd$ L9.6.6 $\rhd$ $\quad$ $\mathrm{case}(\mathsf{N} \therefore \mathsf{N},\mathsf{z},\bullet)$;
Replace $\rhd$ L9.8.6 $\rhd$ $\quad$ $\bullet$]

9.9 Weak and strong representations

I will say that [x]$^{\cdot}$ *weakly represents* falsehood if [$\mathrm{if}(\mathsf{x},\mathsf{F},\mathsf{T}) \equiv \mathsf{T}$]$^{\cdot}$ and that [x]$^{\cdot}$ *strongly represents* falsehood if [$\mathsf{x} \equiv \mathsf{F}$]$^{\cdot}$. Here you have some maps that

---

weakly represent
strongly represent

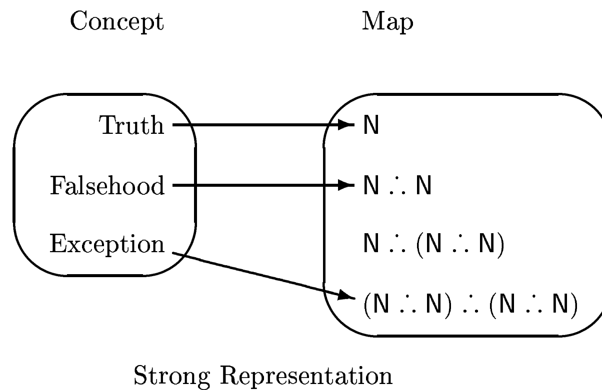
weakly represent falsehood:

$$[\, \mathsf{N} \therefore \mathsf{N} \,]^{\cdot}$$
$$[\, \mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N}) \,]^{\cdot}$$
$$[\, \mathsf{N} \therefore (\mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N})) \,]^{\cdot}$$
$$[\, \mathsf{N} \therefore ((\mathsf{N} \therefore \mathsf{N}) \therefore \mathsf{N}) \,]^{\cdot}$$

Falsehood has many weak representations and exactly one strong representation. In general, when I represent some concept by maps, I introduce both a weak and a strong representation.

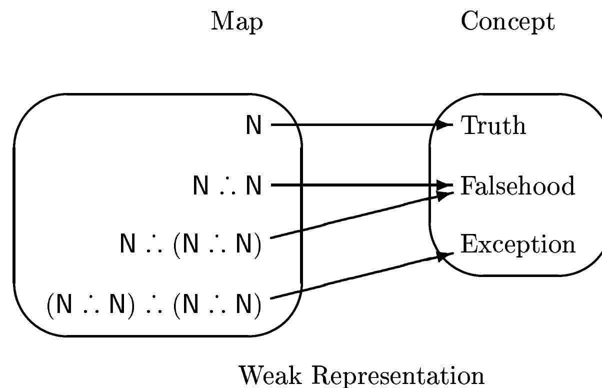
I will say that $[\, \mathsf{x} \,]^{\cdot}$ weakly represents truth if $[\, \mathsf{x} \equiv \mathsf{T} \,]^{\cdot}$ and that $[\, \mathsf{x} \,]^{\cdot}$ strongly represents truth if $[\, \mathsf{x} \equiv \mathsf{T} \,]^{\cdot}$. Truth is the only concept for which I choose the same definition for the weak and the strong representation.

You use the strong representation if you have a concept and want to find its representation. Here you have some examples:



Strong Representation

You use the weak representation if you have a map and want to know what it represents:



Weak Representation

Informally, you may think of the strong representation as a function $[\,\mathsf{S}\,]^{\cdot}$ which outputs a map when it inputs a concept:

$$\begin{array}{lcl} [\,\mathsf{S}\,\text{'}\,\text{truth} & \equiv & \mathsf{N}\,]^{\circ} \\ [\,\mathsf{S}\,\text{'}\,\text{falsehood} & \equiv & \mathsf{N} \therefore \mathsf{N}\,]^{\circ} \\ [\,\mathsf{S}\,\text{'}\,\text{exception} & \equiv & (\mathsf{N} \therefore \mathsf{N}) \therefore (\mathsf{N} \therefore \mathsf{N})\,]^{\circ} \end{array}$$

Likewise, you may think of the weak representation as a function $[\,\mathsf{W}\,]^{\cdot}$ which outputs a concept when it inputs a map:

$$\begin{array}{lcl} [\,\mathsf{W}\,\text{'}\,\mathsf{N} & \equiv & \text{truth}\,]^{\circ} \\ [\,\mathsf{W}\,\text{'}\,(\mathsf{N} \therefore \mathsf{N}) & \equiv & \text{falsehood}\,]^{\circ} \\ [\,\mathsf{W}\,\text{'}\,(\mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N})) & \equiv & \text{falsehood}\,]^{\circ} \\ [\,\mathsf{W}\,\text{'}\,((\mathsf{N} \therefore \mathsf{N}) \therefore (\mathsf{N} \therefore \mathsf{N})) & \equiv & \text{exception}\,]^{\circ} \end{array}$$

The statements above are informal because "truth", "falsehood", and "exception" are informal concepts and not something that can really occur in a mathematical formula.

I define a strong and a weak representation in this book. If you want to represent concepts differently, you must define your own strong and weak representation.

A strong and a weak representation must fit together in the following sense: If you take a concept $[\,\mathsf{x}\,]^{\cdot}$ and apply the strong representation to it, then you obtain a map $[\,\mathsf{S}\,\text{'}\,\mathsf{x}\,]^{\cdot}$ that represents $[\,\mathsf{x}\,]^{\cdot}$. If you then apply the weak representation to $[\,\mathsf{S}\,\text{'}\,\mathsf{x}\,]^{\cdot}$ then you must get $[\,\mathsf{x}\,]^{\cdot}$ back:

$$[\,\mathsf{W}\,\text{'}\,(\mathsf{S}\,\text{'}\,\mathsf{x}) \equiv \mathsf{x}\,]^{\cdot}.$$

The opposite is not true. You do not have $[\,\mathsf{S}\,\text{'}\,(\mathsf{W}\,\text{'}\,\mathsf{x}) \equiv \mathsf{x}\,]^{\cdot}$. As an example, $[\,\mathsf{S}\,\text{'}\,(\mathsf{W}\,\text{'}\,(\mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N}))) \equiv \mathsf{S}\,\text{'}\,\text{falsehood} \equiv \mathsf{N} \therefore \mathsf{N}\,]^{\circ}$.

Now let $[\,\mathsf{R}\,]^{\cdot}$ be the function for which

$$[\,\mathsf{R}\,\text{'}\,\mathsf{x} \equiv \mathsf{S}\,\text{'}\,(\mathsf{W}\,\text{'}\,\mathsf{x})\,]^{\cdot}.$$

As an example you have

$$\begin{array}{lcl} [\,\mathsf{R}\,\text{'}\,(\mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N})) & \equiv & \mathsf{S}\,\text{'}\,(\mathsf{W}\,\text{'}\,(\mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N}))) \\ & \equiv & \mathsf{S}\,\text{'}\,\text{falsehood} \\ & \equiv & \mathsf{N} \therefore \mathsf{N}\,]^{\circ} \end{array}$$

In other words you have

$$[\,\mathsf{R}\,\text{'}\,(\mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N})) \equiv \mathsf{N} \therefore \mathsf{N}\,]^{\cdot}.$$

The function $[\,\mathsf{R}\,]^{\cdot}$ is a function that takes formal mathematical objects as input and produces formal mathematical objects as output. I cannot define $[\,\mathsf{W}\,]^{\cdot}$ and $[\,\mathsf{S}\,]^{\cdot}$ formally because they deal with informal concepts like "truth", "falsehood", and "exception", but I could define $[\,\mathsf{R}\,]^{\cdot}$ formally if I wanted to.

A function like $[\,\mathsf{R}\,]^{\cdot}$ is known as a *retract*. It has the following property:

$$[\,\mathsf{R}\,'\,(\mathsf{R}\,'\,\mathsf{x}) \equiv \mathsf{R}\,'\,\mathsf{x}\,]^{\cdot}.$$

Here you have an informal proof of the statement:

$$\begin{array}{lll} [\,\mathsf{R}\,'\,(\mathsf{R}\,'\,\mathsf{x}) & \equiv & \mathsf{S}\,'\,(\mathsf{W}\,'\,(\mathsf{S}\,'\,(\mathsf{W}\,'\,\mathsf{x}))) \\ & \equiv & \mathsf{S}\,'\,(\mathsf{W}\,'\,\mathsf{x}) \\ & \equiv & \mathsf{R}\,'\,\mathsf{x}\,]^{\circ} \end{array}$$

In the second step I use $[\,\mathsf{W}\,'\,(\mathsf{S}\,'\,\mathsf{y}) \equiv \mathsf{y}\,]^{\circ}$.

Exercise 9.9.1 Does $[\,\lambda\mathsf{x}.\mathsf{T}\,]^{\cdot}$ weakly represent falsehood? Does $[\,\lambda\mathsf{x}.\mathsf{T}\,]^{\cdot}$ represent falsehood strongly?

9.10 Decimal fractions

I define

$$\begin{array}{lcr} [\,\underline{0} & \doteq & \mathsf{N}\,] \\ [\,\underline{1} & \doteq & \mathsf{N} \therefore \mathsf{N}\,] \\ [\,\underline{2} & \doteq & (\mathsf{N} \therefore \mathsf{N}) \therefore \mathsf{N}\,] \\ [\,\underline{3} & \doteq & \mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N})\,] \\ [\,\underline{4} & \doteq & ((\mathsf{N} \therefore \mathsf{N}) \therefore \mathsf{N}) \therefore \mathsf{N}\,] \\ [\,\underline{5} & \doteq & (\mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N})) \therefore \mathsf{N}\,] \\ [\,\underline{6} & \doteq & (\mathsf{N} \therefore \mathsf{N}) \therefore (\mathsf{N} \therefore \mathsf{N})\,] \\ [\,\underline{7} & \doteq & \mathsf{N} \therefore ((\mathsf{N} \therefore \mathsf{N}) \therefore \mathsf{N})\,] \\ [\,\underline{8} & \doteq & \mathsf{N} \therefore (\mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N}))\,] \\ [\,\underline{9} & \doteq & (((\mathsf{N} \therefore \mathsf{N}) \therefore \mathsf{N}) \therefore \mathsf{N}) \therefore \mathsf{N}\,] \\ [\,\underline{\infty} & \doteq & ((\mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N})) \therefore \mathsf{N}) \therefore \mathsf{N}\,] \\ [\,\oplus & \doteq & \mathsf{N}\,] \\ [\,\ominus & \doteq & \mathsf{N} \therefore \mathsf{N}\,] \\ [\,\underline{\mathrm{E}} & \doteq & \mathsf{N}\,] \\ [\,\underline{\mathrm{F}} & \doteq & \mathsf{N} \therefore \mathsf{N}\,] \end{array}$$

If $[\,\mathsf{x}\,]^{\cdot}$ is an exact fraction which differs from $[\,+\infty\,]^{\cdot}$ and $[\,-\infty\,]^{\cdot}$, then $[\,\mathsf{x}\,]^{\cdot}$ can be written on the form

$$[\,\mathsf{m}_s\,\mathsf{m}_0.\mathsf{m}_1\,\mathsf{m}_2\,\cdots\,\mathsf{m}_p \cdot 10^{\mathsf{e}_s\,\mathsf{e}_0\mathsf{e}_1\,\mathsf{e}_2\,\cdots\,\mathsf{e}_q}\,]^{\circ}$$

where

- $[\,\mathsf{m}_s\,]^{\cdot}$ is the sign of the mantissa.
- $[\,\mathsf{m}_0, \ldots, \mathsf{m}_p\,]^{\cdot}$ are the digits of the mantissa.
- $[\,\mathsf{e}_s\,]^{\cdot}$ is the sign of the exponent.

---
retract

- $[\, \mathsf{e}_0, \ldots, \mathsf{e}_q \,]^{\cdot}$ are the digits of the exponent.

I choose to represent such an $[\, \mathsf{x} \,]^{\cdot}$ thus:

$$[\, \langle\langle \mathsf{N}, \underline{\mathtt{E}}, \langle \mathsf{m}_s, \mathsf{m}_0, \mathsf{m}_1, \mathsf{m}_2, \ldots, \mathsf{m}_p \rangle^{\cdot}, \mathsf{e}_s, \mathsf{e}_0, \mathsf{e}_1, \mathsf{e}_2, \ldots, \mathsf{e}_q \rangle^{\cdot} \rangle^{\cdot} \,]^{\cdot}.$$

$[\, \underline{\mathtt{E}} \,]^{\cdot}$ indicates that the fraction is exact as opposed to floating. Here you have some examples:

$$[\, +1.23 \cdot 10^{+45} = \langle\langle \mathsf{N}, \underline{\mathtt{E}}, \langle \oplus, \underline{1}, \underline{2}, \underline{3} \rangle^{\cdot}, \oplus, \underline{4}, \underline{5} \rangle^{\cdot} \rangle^{\cdot} \,],$$

$$[\, -1.23 \cdot 10^{+45} = \langle\langle \mathsf{N}, \underline{\mathtt{E}}, \langle \ominus, \underline{1}, \underline{2}, \underline{3} \rangle^{\cdot}, \oplus, \underline{4}, \underline{5} \rangle^{\cdot} \rangle^{\cdot} \,],$$

$$[\, +1.23 \cdot 10^{-45} = \langle\langle \mathsf{N}, \underline{\mathtt{E}}, \langle \oplus, \underline{1}, \underline{2}, \underline{3} \rangle^{\cdot}, \ominus, \underline{4}, \underline{5} \rangle^{\cdot} \rangle^{\cdot} \,], \text{ and}$$

$$[\, -1.23 \cdot 10^{-45} = \langle\langle \mathsf{N}, \underline{\mathtt{E}}, \langle \ominus, \underline{1}, \underline{2}, \underline{3} \rangle^{\cdot}, \ominus, \underline{4}, \underline{5} \rangle^{\cdot} \rangle^{\cdot} \,].$$

I choose that zero has one digit and that the sign of zero is $[\, \oplus \,]^{\cdot}$:

$$[\, 1.23 = 1.23 \cdot 10^{0} = \langle\langle \mathsf{N}, \underline{\mathtt{E}}, \langle \oplus, \underline{1}, \underline{2}, \underline{3} \rangle^{\cdot}, \oplus, \underline{0} \rangle^{\cdot} \rangle^{\cdot} \,].$$

I choose that zero has exponent zero:

$$[\, 0 = 0 \cdot 10^{0} = \langle\langle \mathsf{N}, \underline{\mathtt{E}}, \langle \oplus, \underline{0} \rangle^{\cdot}, \oplus, \underline{0} \rangle^{\cdot} \rangle^{\cdot} \,].$$

I choose to represent $[\, +\infty \,]^{\cdot}$ and $[\, -\infty \,]^{\cdot}$ thus:

$$[\, +\infty = \langle\langle \mathsf{N}, \underline{\mathtt{E}}, \langle \oplus, \underline{\infty} \rangle^{\cdot}, \oplus, \underline{0} \rangle^{\cdot} \rangle^{\cdot} \,], \text{ and}$$

$$[\, -\infty = \langle\langle \mathsf{N}, \underline{\mathtt{E}}, \langle \ominus, \underline{\infty} \rangle^{\cdot}, \oplus, \underline{0} \rangle^{\cdot} \rangle^{\cdot} \,].$$

I choose to represent floating fractions like exact fractions except that I replace $[\, \underline{\mathtt{E}} \,]^{\cdot}$ by $[\, \underline{\mathtt{F}} \,]^{\cdot}$:

$$[\, -1.23\mathtt{F} \cdot 10^{45} = \langle\langle \mathsf{N}, \underline{\mathtt{F}}, \langle \ominus, \underline{1}, \underline{2}, \underline{3} \rangle^{\cdot}, \oplus, \underline{4}, \underline{5} \rangle^{\cdot} \rangle^{\cdot} \,].$$

The precision of a floating fractions equals the number of digits in the mantissa.

The mantissa of a floating fraction that equals $[\, +\infty \,]^{\cdot}$ and $[\, -\infty \,]^{\cdot}$ has form $[\, \langle \oplus, \underline{\infty}, \underline{0}, \ldots, \underline{0} \rangle^{\cdot} \,]^{\cdot}$ and $[\, \langle \ominus, \underline{\infty}, \underline{0}, \ldots, \underline{0} \rangle^{\cdot} \,]^{\cdot}$, respectively.

The last digit of the mantissa of a non-zero exact fractions is non-zero:

$$[\, 1200 = \langle\langle \mathsf{N}, \underline{\mathtt{E}}, \langle \oplus, \underline{1}, \underline{2} \rangle^{\cdot}, \oplus, \underline{3} \rangle^{\cdot} \rangle^{\cdot} \,].$$

Exercise 9.10.1 What are the strong representations of the following:

(a) $[\, 117 \,]^{\cdot}$

(b) $[\, 0.00117 \,]^{\cdot}$

(c) $[\, 117\mathtt{F} \,]^{\cdot}$

(d) $[\, \infty \,]^{\cdot}$

(e) $[\, \infty @ 5 \,]^{\cdot}$

(f) $[\, 11700 \,]^{\cdot}$

(g) $[\, 1170\emptyset\mathtt{F} \,]^{\cdot}$

9.11 Lists

I define

$[\,\langle\rangle \doteq (\mathsf{N} \therefore \mathsf{N}) \therefore \mathsf{N}\,]$, and

$[\,\mathsf{x} :: \mathsf{y} \doteq (\mathsf{N} \therefore \mathsf{N}) \therefore \mathsf{x} \therefore \mathsf{y} \therefore \mathsf{N}\,]$.

You have now seen my strong representation of truth, falsehood, exception, decimal fractions, the empty list, and pairs. The details of the representation are not so important. What you really need to remember in the following chapters are:

- I represent all mathematical concepts by maps.
- The representation consists of a weak and a strong representation.

9.12 Summary of representation

Here you have a summary of the notation where I use simple lists to make the structure of the representation clear:

$[\,\langle\rangle^{\cdot}\,]^{\cdot}$	truth
$[\,\langle\langle\rangle^{\cdot}\rangle^{\cdot}\,]^{\cdot}$	falsehood
$[\,\langle\langle\langle\rangle^{\cdot}\rangle^{\cdot}\rangle^{\cdot}\,]^{\cdot}$	empty list
$[\,\langle\langle\langle\rangle^{\cdot}\rangle^{\cdot}, \mathsf{x}\rangle^{\cdot}\,]^{\cdot}$	exception
$[\,\langle\langle\langle\rangle^{\cdot}\rangle^{\cdot}, \mathsf{x}, \mathsf{y}\rangle^{\cdot}\,]^{\cdot}$	pair
$[\,\langle\langle\langle\rangle^{\cdot}, \underline{\mathsf{E}}, \ldots\rangle^{\cdot}\rangle^{\cdot}\,]^{\circ}$	exact fraction
$[\,\langle\langle\langle\rangle^{\cdot}, \underline{\mathsf{F}}, \ldots\rangle^{\cdot}\rangle^{\cdot}\,]^{\circ}$	floating fraction

In the table above I state that $[\,\langle\langle\langle\rangle^{\cdot}\rangle^{\cdot}, \mathsf{x}\rangle^{\cdot}\,]^{\cdot}$ is an exception for all values of $[\,\mathsf{x}\,]^{\cdot}$. In this book I merely consider the exception $[\,\bullet\,]^{\cdot}$ for which $[\,\mathsf{x}\,]^{\cdot}$ is $[\,\mathsf{N}\,]^{\cdot}$. In some situations it is nice to have *more than one exception*. As an example, it could be convenient to have one exception for "division by zero" and another one for "square root of negative number". The representation above has room for infinitely many different exceptions in that you can choose $[\,\mathsf{x}\,]^{\cdot}$ in $[\,\langle\langle\langle\rangle^{\cdot}\rangle^{\cdot}, \mathsf{x}\rangle^{\cdot}\,]^{\cdot}$ freely.

9.13 The weak representation

In the following I define nine *type functions* that you will need in the chapters to come. I name the functions $[\,\overline{\mathbf{T}}\,]^{\cdot}$, $[\,\overline{\mathbf{F}}\,]^{\cdot}$, $[\,\overline{\mathbf{B}}\,]^{\cdot}$, $[\,\overline{\mathbf{X}}\,]^{\cdot}$, $[\,\overline{\mathbf{D}}_{\infty}\,]^{\cdot}$, $[\,\overline{\mathbf{D}}.\,]^{\cdot}$, $[\,\overline{\mathbf{D}}\,]^{\cdot}$, $[\,\overline{\mathbf{E}}\,]^{\cdot}$, and $[\,\overline{\mathbf{P}}\,]^{\cdot}$, and they have the following properties:

$[\,\overline{\mathbf{T}}\ \ '\mathsf{x} \equiv \mathsf{T}\,]^{\cdot}$ when $[\,\mathsf{x}\,]^{\cdot}$ weakly represents truth
$[\,\overline{\mathbf{F}}\ \ '\mathsf{x} \equiv \mathsf{T}\,]^{\cdot}$ when $[\,\mathsf{x}\,]^{\cdot}$ weakly represents falsehood

more than one exception
type function

$[\overline{\mathbf{B}}\ \ {}'x \equiv \mathsf{T}]^{\cdot}$ when $[x]^{\cdot}$ weakly represents a truth value
$[\overline{\mathbf{X}}\ \ {}'x \equiv \mathsf{T}]^{\cdot}$ when $[x]^{\cdot}$ weakly represents an exception
$[\overline{\mathbf{D}}_{\infty}\ {}'x \equiv \mathsf{T}]^{\cdot}$ when $[x]^{\cdot}$ weakly represents an exact fraction
$[\overline{\mathbf{D}}.\ \ {}'x \equiv \mathsf{T}]^{\cdot}$ when $[x]^{\cdot}$ weakly represents a floating fraction
$[\overline{\mathbf{D}}\ \ {}'x \equiv \mathsf{T}]^{\cdot}$ when $[x]^{\cdot}$ weakly represents a decimal fraction
$[\overline{\mathbf{E}}\ \ {}'x \equiv \mathsf{T}]^{\cdot}$ when $[x]^{\cdot}$ weakly represents the empty list
$[\overline{\mathbf{P}}\ \ {}'x \equiv \mathsf{T}]^{\cdot}$ when $[x]^{\cdot}$ weakly represents a pair

To define these functions I first define $[\tilde{\neg}x]^{\cdot}$, $[x \mathbin{\tilde{\wedge}} y]^{\cdot}$, and $[x \mathbin{\tilde{=}} y]^{\cdot}$. $[\tilde{\neg}x]^{\cdot}$ is a simplified version of $[\neg x]^{\cdot}$. $[\tilde{\neg}x]^{\cdot}$ is true if $[x]^{\cdot}$ is not true and vice verse, except that $[\tilde{\neg}\bot \equiv \bot]^{\cdot}$:

$$\left[\boxed{\tilde{\neg}x} \doteq x\left\langle\begin{matrix}\mathsf{F}\\ \mathsf{T}\end{matrix}\right.\right].$$

$[x \mathbin{\tilde{\wedge}} y]^{\cdot}$ is a simplified version of $[x \wedge y]^{\cdot}$. $[x \mathbin{\tilde{\wedge}} y]^{\cdot}$ is true if $[x]^{\cdot}$ and $[y]^{\cdot}$ are true:

$$\left[\boxed{x \mathbin{\tilde{\wedge}} y} \doteq x\left\langle\begin{matrix}\mathrm{case}(y, \mathsf{T}, \mathsf{F})\\ \mathrm{case}(y, \mathsf{F}, \mathsf{F})\end{matrix}\right.\right].$$

If $[x]^{\cdot}$ and $[y]^{\cdot}$ are built up from $[\mathsf{N}]^{\cdot}$ and $[u \therefore v]^{\cdot}$, then $[x \mathbin{\tilde{=}} y]^{\cdot}$ tests $[x]^{\cdot}$ and $[y]^{\cdot}$ for equality.

$$\left[\boxed{x \mathbin{\tilde{=}} y} \doteq x\left\langle\begin{matrix}\mathrm{case}(y, \mathsf{T}, \mathsf{F})\\ \mathrm{case}(y, \mathsf{F}, x\ \mathrm{Head} \mathbin{\tilde{=}} y\ \mathrm{Head} \mathbin{\tilde{\wedge}} x\ \mathrm{Tail} \mathbin{\tilde{=}} y\ \mathrm{Tail})\end{matrix}\right.\right]$$

$[x \mathbin{\tilde{\neq}} y]^{\cdot}$ is the negation of $[x \mathbin{\tilde{=}} y]^{\cdot}$:

$$\left[\boxed{x \mathbin{\tilde{\neq}} y} \doteq \tilde{\neg}x \mathbin{\tilde{=}} y\right]$$

Here you have some examples:

$[(\mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N})) \mathbin{\tilde{=}} (\mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N}))]$, and

$[(\mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N})) \mathbin{\tilde{\neq}} ((\mathsf{N} \therefore \mathsf{N}) \therefore \mathsf{N})]$.

$[\tilde{\neg}x]^{\cdot}$ simple not x
$[x \mathbin{\tilde{\wedge}} y]^{\cdot}$ x simple and y
$[x \mathbin{\tilde{=}} y]^{\cdot}$ x simple equal y
$[x \mathbin{\tilde{\neq}} y]^{\cdot}$ x simple unequal y

Without further ado, here are the definitions of the type functions:

$$[\ \boxed{\overline{\mathbf{T}}} \doteq \lambda x.\ \mathrm{case}(x, T, F)\ ]$$
$$[\ \boxed{\overline{\mathbf{F}}} \doteq \lambda x.\ \mathrm{case}(x, F, \mathrm{case}(x\ \mathrm{Head}, T, F))\ ]$$
$$[\ \boxed{\overline{\mathbf{B}}} \doteq \lambda x.\ \mathrm{case}(x\ \mathrm{Head}, T, F)\ ]$$
$$[\ \boxed{\overline{\mathbf{E}}} \doteq \lambda x.\ x\ \mathrm{Head} \tilde{=} F \tilde{\wedge} x\ \mathrm{Tail}\ ]$$
$$[\ \boxed{\overline{\mathbf{X}}} \doteq \lambda x.\ x\ \mathrm{Head} \tilde{=} F \tilde{\wedge} \tilde{\neg} x\ \mathrm{Tail} \tilde{\wedge} x\ \mathrm{Tail\ Tail}\ ]$$
$$[\ \boxed{\overline{\mathbf{P}}} \doteq \lambda x.\ x\ \mathrm{Head} \tilde{=} F \tilde{\wedge} \tilde{\neg} x\ \mathrm{Tail\ Tail}\ ]$$
$$[\ \boxed{\overline{\mathbf{D}}_\infty} \doteq \lambda x.\ \tilde{\neg} x\ \mathrm{Head} \tilde{\wedge} x\ \mathrm{Head\ Head} \tilde{\wedge} x\ \mathrm{Head\ Tail\ Head}\ ]$$
$$[\ \boxed{\overline{\mathbf{D}}.} \doteq \lambda x.\ x\ \mathrm{Head\ Head} \tilde{\wedge} \tilde{\neg} x\ \mathrm{Head\ Tail\ Head}\ ]$$
$$[\ \boxed{\overline{\mathbf{D}}} \doteq \lambda x.\ \mathrm{case}(\overline{\mathbf{D}}_\infty\ '\ x, T, \overline{\mathbf{D}}.\ '\ x)\ ]$$

Here you have some examples of use of the type functions:

$[\ \overline{\mathbf{T}}\ '\ T \equiv T\ ]^{\cdot}$ $[\ \overline{\mathbf{T}}\ '\ F \equiv F\ ]^{\cdot}$ $[\ \overline{\mathbf{T}}\ '\ \bullet \equiv F\ ]^{\cdot}$
$[\ \overline{\mathbf{F}}\ '\ T \equiv F\ ]^{\cdot}$ $[\ \overline{\mathbf{F}}\ '\ F \equiv T\ ]^{\cdot}$ $[\ \overline{\mathbf{F}}\ '\ \bullet \equiv F\ ]^{\cdot}$
$[\ \overline{\mathbf{B}}\ '\ T \equiv T\ ]^{\cdot}$ $[\ \overline{\mathbf{B}}\ '\ F \equiv T\ ]^{\cdot}$ $[\ \overline{\mathbf{B}}\ '\ \bullet \equiv F\ ]^{\cdot}$
$[\ \overline{\mathbf{X}}\ '\ T \equiv F\ ]^{\cdot}$ $[\ \overline{\mathbf{X}}\ '\ F \equiv F\ ]^{\cdot}$ $[\ \overline{\mathbf{X}}\ '\ \bullet \equiv T\ ]^{\cdot}$

It is straightforward but boring to prove the statements above.

9.14 Answers

Answer 9.9.1 $[\ \lambda x.T\ ]^{\cdot}$ weakly represents falsehood because $[\ \mathrm{if}(\lambda x.T, F, T) \equiv T\ ]^{\cdot}$ holds:

[**Mac lemma L9.14.1** : $\mathrm{if}(\lambda x.T, F, T) \equiv T$]

[**Mac proof of** L9.14.1:

Algebra ▷	$\mathrm{if}(\lambda x.T, F, T)$	;
Definition ▷	$\mathrm{case}(\lambda x.T, F, \mathrm{case}((\lambda x.T)\ \mathrm{Head}, T, \bullet))$	;
Replace ▷ CaseLambda ▷	$\mathrm{case}((\lambda x.T)\ \mathrm{Head}, T, \bullet)$	;
Definition ▷	$\mathrm{case}((\lambda x.T)\ '\ N, T, \bullet)$	;
Replace ▷ ApplyLambda ▷	$\mathrm{case}(\langle T \mid x{:=}N\rangle, T, \bullet)$	;
Substitution ▷	$\mathrm{case}(T, T, \bullet)$	;
Definition ▷	$\mathrm{case}(N, T, \bullet)$	;
Replace ▷ CaseNil ▷	T	]

$[\ \lambda x.T\ ]^{\cdot}$ does not represent falsehood strongly because $[\ \lambda x.T \equiv F\ ]^{\cdot}$ fails. To see that $[\ \lambda x.T \equiv F\ ]^{\cdot}$ fails proceed as follows: According to the definition of $[\ F\ ]^{\cdot}$ and $[\ x \therefore y\ ]^{\cdot}$ you have

$$[\ F \equiv N \therefore N \equiv \lambda z.\ \mathrm{case}(z, N, N)\ ]^{\cdot}.$$

Hence, $[\,\mathsf{F}\,]^{\cdot}$ is a function. Furthermore,

$$[\,\mathsf{F}\,'\perp \equiv \perp\,]^{\cdot}$$

whereas

$$[\,(\lambda \mathsf{x}.\mathsf{T})\,'\perp \equiv \mathsf{T}\,]^{\cdot}$$

so the functions $[\,(\lambda \mathsf{x}.\mathsf{T})\,]^{\cdot}$ and $[\,\mathsf{F}\,]^{\cdot}$ differ for at least one argument (for a more formal way of stating results like this, see Chapter 16).

Answer 9.10.1

(a) $[\,117 \equiv \langle\langle \mathsf{N}, \underline{\mathrm{E}}, \langle\oplus, \underline{1}, \underline{1}, \underline{7}\rangle^{\cdot}, \oplus, \underline{2}\rangle^{\cdot}\rangle^{\cdot}\,]^{\cdot}$.

(b) $[\,0.00117 \equiv \langle\langle \mathsf{N}, \underline{\mathrm{E}}, \langle\oplus, \underline{1}, \underline{1}, \underline{7}\rangle^{\cdot}, \ominus, \underline{3}\rangle^{\cdot}\rangle^{\cdot}\,]^{\cdot}$.

(c) $[\,117\mathrm{F} \equiv \langle\langle \mathsf{N}, \underline{\mathrm{F}}, \langle\oplus, \underline{1}, \underline{1}, \underline{7}\rangle^{\cdot}, \oplus, \underline{2}\rangle^{\cdot}\rangle^{\cdot}\,]^{\cdot}$.

(d) $[\,\infty \equiv \langle\langle \mathsf{N}, \underline{\mathrm{E}}, \langle\oplus, \underline{\infty}\rangle^{\cdot}, \oplus, \underline{0}\rangle^{\cdot}\rangle^{\cdot}\,]^{\cdot}$ ($[\,\infty\,]^{\cdot}$ and $[\,+\infty\,]^{\cdot}$ are the same).

(e) $[\,\infty@5 \equiv \langle\langle \mathsf{N}, \underline{\mathrm{F}}, \langle\oplus, \underline{\infty}, \underline{0}, \underline{0}, \underline{0}, \underline{0}\rangle^{\cdot}, \oplus, \underline{0}\rangle^{\cdot}\rangle^{\cdot}\,]^{\cdot}$ (pad with zeros to get precision right).

(f) $[\,11700 \equiv \langle\langle \mathsf{N}, \underline{\mathrm{E}}, \langle\oplus, \underline{1}, \underline{1}, \underline{7}\rangle^{\cdot}, \oplus, \underline{4}\rangle^{\cdot}\rangle^{\cdot}\,]^{\cdot}$ (remove trailing zeros in mantissas of exact fractions).

(g) $[\,1170\emptyset\mathrm{F} \equiv \langle\langle \mathsf{N}, \underline{\mathrm{F}}, \langle\oplus, \underline{1}, \underline{1}, \underline{7}, \underline{0}\rangle^{\cdot}, \oplus, \underline{4}\rangle^{\cdot}\rangle^{\cdot}\,]^{\cdot}$ (choose the right number of trailing zeros to get precision right).

Chapter 10

Sets

10.1 Introduction

$[117 \in \mathbf{N}]^{\cdot}$ is true. The term says that $[117]^{\cdot}$ belongs to the set of natural numbers. In the term,

$$[\boxed{\mathbf{N}}]^{\cdot}$$

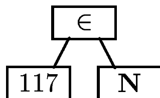
denotes the *set of natural numbers*, and

$$[\boxed{\mathsf{x} \in \mathsf{y}}]^{\cdot}$$

denotes the *membership relation*. Here you have some examples:

$$
\begin{array}{lll}
[117 \in \mathbf{N} & \equiv & \mathsf{T}]^{\cdot} \\
[117.1 \in \mathbf{N} & \equiv & \mathsf{F}]^{\cdot} \\
[\mathsf{T} \in \mathbf{N} & \equiv & \mathsf{F}]^{\cdot} \\
[\bullet \in \mathbf{N} & \equiv & \mathsf{F}]^{\cdot} \\
[\bot \in \mathbf{N} & \equiv & \bot]^{\cdot}
\end{array}
$$

The construct $[\mathsf{x} \in \mathsf{y}]^{\cdot}$ is binary infix and $[\mathbf{N}]^{\cdot}$ is null-ary outfix. $[\mathsf{x} \in \mathsf{y}]^{\cdot}$ has the same priority as $[\mathsf{x} = \mathsf{y}]^{\cdot}$. Here you have the syntax tree of $[117 \in \mathbf{N}]^{\cdot}$:



$[\mathsf{x} \in \mathsf{y}]^{\cdot}$ is strict in $[\mathsf{y}]^{\cdot}$:

[**Mac rule XBelongsToStrict** : $\mathsf{x} \in \bot \equiv \bot$]

Exercise 10.1.1 Draw the syntax tree of $[2 + 2 = 4 \in \mathbf{N}]^{\cdot}$. What is the value of the term?

$[\mathbf{N}]^{\cdot}$	the set of natural numbers
	set of natural numbers
$[\mathsf{x} \in \mathsf{y}]^{\cdot}$	x belongs to y
	membership relation

10.2 Some sets

Here you have some of the sets I will use in this chapter:

B the set of truth values.

D the set of decimal fractions.

X the set of exceptions.

Here you have some examples:

$[\mathsf{F} \in \mathbf{B} \equiv \mathsf{T}]$	$[117 \in \mathbf{B} \equiv \mathsf{F}]$	$[\bullet \in \mathbf{B} \equiv \mathsf{F}]$	$[\bot \in \mathbf{B} \equiv \bot]$
$[\mathsf{F} \in \mathbf{D} \equiv \mathsf{F}]$	$[117 \in \mathbf{D} \equiv \mathsf{T}]$	$[\bullet \in \mathbf{D} \equiv \mathsf{F}]$	$[\bot \in \mathbf{D} \equiv \bot]$
$[\mathsf{F} \in \mathbf{X} \equiv \mathsf{F}]$	$[117 \in \mathbf{X} \equiv \mathsf{F}]$	$[\bullet \in \mathbf{X} \equiv \mathsf{T}]$	$[\bot \in \mathbf{X} \equiv \bot]$

Exercise 10.2.1 What are the values of the following:

a $[\mathsf{T} \in \mathbf{B}]$.

b $[\mathsf{T} \in \mathbf{X}]$.

c $[2+3 \in \mathbf{D}]$.

d $[1/0 \in \mathbf{X}]$.

e $[(-1)! \in \mathbf{D}]$.

10.3 Classical maps

$[\mathsf{x} \in \mathbf{D}]$ equals $[\mathsf{T}]$ if $[\mathsf{x}]$ represents a decimal fraction and $[\mathsf{x}]$ is *classical*. $[\mathsf{x} \in \mathbf{D}]$ equals $[\bot]$ if $[\mathsf{x}]$ is *non-classical*. $[\mathsf{x} \in \mathbf{D}]$ equals $[\mathsf{F}]$ if $[\mathsf{x}]$ is classical but does not represent a decimal fraction ("represents" means "represents weakly").

I define classicallity later. Until further, you just need to know a few things about classicallity.

The function $[\boxed{\ell}]$ tests for classicallity:

$[\ell\,\text{'}\,\mathsf{x} \equiv \mathsf{T}]$	if and only if	$[\mathsf{x}]$ is classical
$[\ell\,\text{'}\,\mathsf{x} \equiv \bot]$	if and only if	$[\mathsf{x}]$ is non-classical

Classicallity has the following properties:

$[\mathbf{B}]$	the set of truth values
$[\mathbf{D}]$	the set of decimal fractions
$[\mathbf{X}]$	the set of exceptions
	classical
	non-classical
$[\ell]$	classical

$\left[\, \textbf{Mac rule ClassicalNil} \;:\; \ell \,\text{'}\, \mathsf{N} \equiv \mathsf{T} \,\right]$

$\left[\, \textbf{Mac rule ClassicalBottom} \;:\; \ell \,\text{'}\, \bot \equiv \bot \,\right]$

$\left[\, \textbf{Mac rule ClassicalPair} \;:\; \ell \,\text{'}\, (\mathsf{x} \therefore \mathsf{y}) \equiv \ell \,\text{'}\, \mathsf{x} \wedge \ell \,\text{'}\, \mathsf{y} \,\right]$

In other words: $[\, \mathsf{N} \,]^{\cdot}$ is classical, $[\, \bot \,]^{\cdot}$ is non-classical, $[\, \mathsf{x} \therefore \mathsf{y} \,]^{\cdot}$ is classical if $[\, \mathsf{x} \,]^{\cdot}$ and $[\, \mathsf{y} \,]^{\cdot}$ are both classical, and $[\, \mathsf{x} \therefore \mathsf{y} \,]^{\cdot}$ is non-classical if $[\, \mathsf{x} \,]^{\cdot}$ or $[\, \mathsf{y} \,]^{\cdot}$ or both are non-classical. Here you have some examples:

$[\, \ell \,\text{'}\, (\mathsf{N} \therefore \mathsf{N} \therefore \mathsf{N}) \equiv \mathsf{T} \,]^{\cdot}$, and

$[\, \ell \,\text{'}\, (\mathsf{N} \therefore \bot \therefore \mathsf{N}) \equiv \bot \,]^{\cdot}$.

Exercise 10.3.1

a Is $[\, \bullet :: \bullet \,]^{\cdot}$ classical?

b Is $[\, \bot :: \bot \,]^{\cdot}$ classical?

10.4 Relation to the weak representation

$[\, \mathsf{x} \in \mathbf{B} \,]^{\cdot}$ equals $[\, \mathsf{T} \,]^{\cdot}$ for classical $[\, \mathsf{x} \,]^{\cdot}$ that weakly represent a truth value. $[\, \mathsf{x} \in \mathbf{B} \,]^{\cdot}$ equals $[\, \mathsf{F} \,]^{\cdot}$ for classical $[\, \mathsf{x} \,]^{\cdot}$ that do not represent a truth value. $[\, \mathsf{x} \in \mathbf{B} \,]^{\cdot}$ equals $[\, \bot \,]^{\cdot}$ for non-classical $[\, \mathsf{x} \,]^{\cdot}$. Here you have some examples:

$[\, \mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N}) \,]^{\cdot}$	weakly represents falsehood and
$[\, \mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N}) \,]^{\cdot}$	is classical so
$[\, \mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N}) \,]^{\cdot}$	belongs to $[\, \mathbf{B} \,]^{\cdot}$.
$[\, \mathsf{N} \therefore (\bot \therefore \mathsf{N}) \,]^{\cdot}$	weakly represents falsehood but
$[\, \mathsf{N} \therefore (\bot \therefore \mathsf{N}) \,]^{\cdot}$	is non-classical so
$[\, \mathsf{N} \therefore (\bot \therefore \mathsf{N}) \,]^{\cdot}$	does not belong to $[\, \mathbf{B} \,]^{\cdot}$.

In other words,

$$
\begin{array}{lcl}
[\, \mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N}) \in \mathbf{B} & \equiv & \mathsf{T} \,]^{\cdot} \\
[\, \mathsf{N} \therefore (\bot \therefore \mathsf{N}) \in \mathbf{B} & \equiv & \bot \,]^{\cdot}
\end{array}
$$

Exercise 10.4.1 What are the values of the following:

a $[\, \mathsf{N} \in \mathbf{B} \,]^{\cdot}$.

b $[\, \mathsf{N} \therefore \mathsf{N} \in \mathbf{B} \,]^{\cdot}$.

c $[\, (\mathsf{N} \therefore \mathsf{N}) \therefore \mathsf{N} \in \mathbf{B} \,]^{\cdot}$.

d $[\, \mathsf{N} \therefore (\mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N})) \in \mathbf{B} \,]^{\cdot}$.

e $[\, (\mathsf{N} \therefore \bot) \therefore \mathsf{N} \in \mathbf{B} \,]^{\cdot}$.

f $[\, \mathsf{N} \therefore (\mathsf{N} \therefore \bot) \in \mathbf{B} \,]^{\cdot}$.

10.5 Sets of truth values

In addition to the set [**B**]˙ of truth values, I will use the following sets:

T the set of classical maps that weakly represent truth.

F the set of classical maps that weakly represent falsehood.

Exercise 10.5.1 What are the values of the following:

a [T ∈ **T**]˙.

b [F ∈ **T**]˙.

c [• ∈ **T**]˙.

d [⊥ ∈ **T**]˙.

e [N ∴ (N ∴ N) ∈ **T**]˙.

f [N ∴ (N ∴ N) ∈ **F**]˙.

g [N ∴ (⊥ ∴ N) ∈ **T**]˙.

h [N ∴ (⊥ ∴ N) ∈ **F**]˙.

10.6 Sets of decimal fractions

In addition to the set [**D**]˙ of decimal fractions, I will use the following sets:

$\mathbf{D}_{\mathsf{m}}$ the set of decimal fractions of precision [m]˙ including fractions of magnitude [$+\infty$]˙ and [$-\infty$]˙.

$\mathbf{D}^{\cdot}_{\mathsf{m}}$ the set of decimal fractions of precision [m]˙ excluding fractions of magnitude [$+\infty$]˙ and [$-\infty$]˙.

$\mathbf{D}^{\infty}_{\mathsf{m}}$ the set of decimal fractions of precision [m]˙ and magnitude [$+\infty$]˙.

$\mathbf{D}^{-\infty}_{\mathsf{m}}$ the set of decimal fractions of precision [m]˙ and magnitude [$-\infty$]˙.

N the set of natural numbers.

[**T**]˙	the set of true maps
[**F**]˙	the set of false maps
[$\mathbf{D}^{\cdot}_{\mathsf{x}}$]˙	the set of decimal fractions of precision x end
[$\mathbf{D}^{\infty}_{\mathsf{x}}$]˙	the set of infinities of precision x end
[$\mathbf{D}^{-\infty}_{\mathsf{x}}$]˙	the set of minus infinities of precision x end
[**N**]˙	the set of natural numbers
[**Z**]˙	the set of integers
[$\mathbf{Z}^{+}$]˙	the set of positive integers
[$\mathbf{Z}^{-}$]˙	the set of negative integers

$\boxed{\mathbf{Z}}$ the set of integers.

$\boxed{\mathbf{Z}^{+}}$ the set of positive integers.

$\boxed{\mathbf{Z}^{-}}$ the set of negative integers.

Here you have some examples:

$$
\begin{array}{llll}
[\,1.23\mathrm{F} & \in & \mathbf{D}^{\cdot}_{3} & \equiv \mathsf{T}\,]^{\cdot} \\
[\,1.23\mathrm{F} & \in & \mathbf{D}^{\cdot}_{4} & \equiv \mathsf{F}\,]^{\cdot} \\
[\,1.23 & \in & \mathbf{D}^{\cdot}_{3} & \equiv \mathsf{F}\,]^{\cdot} \\
[\,1.23 & \in & \mathbf{D}^{\cdot}_{\infty} & \equiv \mathsf{T}\,]^{\cdot} \\
[\,\infty & \in & \mathbf{D} & \equiv \mathsf{T}\,]^{\cdot} \\
[\,\infty & \in & \mathbf{D}^{\cdot}_{\infty} & \equiv \mathsf{F}\,]^{\cdot} \\
[\,\infty & \in & \mathbf{D}^{\infty}_{\infty} & \equiv \mathsf{T}\,]^{\cdot} \\
[\,-\infty & \in & \mathbf{D}^{-\infty}_{\infty} & \equiv \mathsf{T}\,]^{\cdot}
\end{array}
$$

Note that $[\,\mathbf{D}_{\mathsf{m}}\,]^{\cdot}$, $[\,\mathbf{D}^{\cdot}_{\mathsf{m}}\,]^{\cdot}$, $[\,\mathbf{D}^{\infty}_{\mathsf{m}}\,]^{\cdot}$, and $[\,\mathbf{D}^{-\infty}_{\mathsf{m}}\,]^{\cdot}$ denote sets of floating fractions when $[\,\mathsf{m}\,]^{\cdot}$ is a positive integer and denote sets of exact fractions when $[\,\mathsf{m}\,]^{\cdot}$ is $[\,\infty\,]^{\cdot}$.

Exercise 10.6.1 What are the values of the following:

a $[\,12\mathrm{F} \in \mathbf{D}^{\cdot}_{2}\,]^{\cdot}$

b $[\,\infty@3 \in \mathbf{D}^{\infty}_{3}\,]^{\cdot}$

c $[\,12\mathrm{F} \in \mathbf{N}\,]^{\cdot}$

d $[\,12 \in \mathbf{Z}\,]^{\cdot}$

e $[\,-12 \in \mathbf{Z}^{-}\,]^{\cdot}$

f $[\,0 \in \mathbf{N}\,]^{\cdot}$

g $[\,0 \in \mathbf{Z}^{+}\,]^{\cdot}$

h $[\,0 \in \mathbf{Z}^{-}\,]^{\cdot}$

10.7 The cartesian product

If $[\,\mathsf{A}\,]^{\cdot}$ and $[\,\mathsf{B}\,]^{\cdot}$ are sets, then

$$
\frac{\left[\,\boxed{\mathsf{A} \times \mathsf{B}}\,\right]^{\cdot}}{[\,\mathsf{x} \times \mathsf{y}\,]^{\cdot} \quad \text{x cartesian y}}
$$

denotes the set of pairs $[\mathsf{x} :: \mathsf{y}]^{\cdot}$ for which $[\mathsf{x} \in \mathsf{A}]^{\cdot}$ and $[\mathsf{y} \in \mathsf{B}]^{\cdot}$. Here you have some examples:

$$
\begin{array}{llll}
[1 :: \mathsf{T} & \in & \mathbf{N} \times \mathbf{B} & \equiv \mathsf{T}]^{\cdot} \\
[\mathsf{T} :: 1 & \in & \mathbf{N} \times \mathbf{B} & \equiv \mathsf{F}]^{\cdot} \\
[1 :: 1 & \in & \mathbf{N} \times \mathbf{B} & \equiv \mathsf{F}]^{\cdot} \\
[\mathsf{T} :: \mathsf{T} & \in & \mathbf{N} \times \mathbf{B} & \equiv \mathsf{F}]^{\cdot} \\
[1 :: \bot & \in & \mathbf{N} \times \mathbf{B} & \equiv \bot]^{\cdot} \\
[\bot :: 1 & \in & \mathbf{N} \times \mathbf{B} & \equiv \bot]^{\cdot}
\end{array}
$$

$[\mathsf{x} \times \mathsf{y}]^{\cdot}$ is binary, infix, and right associative:

$$[\mathsf{x} \times \mathsf{y} \times \mathsf{z} \stackrel{\smile}{\to} \mathsf{x} \times (\mathsf{y} \times \mathsf{z})].$$

Exercise 10.7.1 What are the values of the following:

a $[(1\mathrm{F} :: 1.2\mathrm{F}) :: 1.23\mathrm{F} \in (\mathbf{D}_1^{\cdot} \times \mathbf{D}_2^{\cdot}) \times \mathbf{D}_3^{\cdot}]^{\cdot}$.

b $[1\mathrm{F} :: (1.2\mathrm{F} :: 1.23\mathrm{F}) \in (\mathbf{D}_1^{\cdot} \times \mathbf{D}_2^{\cdot}) \times \mathbf{D}_3^{\cdot}]^{\cdot}$.

c $[1\mathrm{F} :: 1.2\mathrm{F} :: 1.23\mathrm{F} \in \mathbf{D}_1^{\cdot} \times \mathbf{D}_2^{\cdot} \times \mathbf{D}_3^{\cdot}]^{\cdot}$.

10.8 The empty list

I use $[\boxed{\mathbf{E}}]^{\cdot}$ to denote the set of classical maps that weakly represent the empty list $[\langle\rangle]^{\cdot}$:

$$
\begin{array}{llll}
[\langle\rangle & \in & \mathbf{E} & \equiv \mathsf{T}]^{\cdot} \\
[1 :: 2 & \in & \mathbf{E} & \equiv \mathsf{F}]^{\cdot}
\end{array}
$$

Exercise 10.8.1 What is the value of $[\langle 1\mathrm{F}, 1.2\mathrm{F}, 1.23\mathrm{F}\rangle \in \mathbf{D}_1^{\cdot} \times \mathbf{D}_2^{\cdot} \times \mathbf{D}_3^{\cdot} \times \mathbf{E}]^{\cdot}$.

10.9 Sets of lists

If $[\mathsf{A}]^{\cdot}$ is a set then I use $[\boxed{\mathsf{A}^*}]^{\cdot}$ to denote the set of lists

$$[\langle \mathsf{a}_1, \ldots, \mathsf{a}_n \rangle]^{\cdot}$$

for which all the elements of the list belong to $[\mathsf{A}]^{\cdot}$. Here you have some examples:

$$
\begin{array}{llll}
\langle 1,2,3\rangle & \in & \mathbf{N}^* & \equiv \mathsf{T} \\
\langle 1,2\rangle & \in & \mathbf{N}^* & \equiv \mathsf{T} \\
\langle 1\rangle & \in & \mathbf{N}^* & \equiv \mathsf{T} \\
\langle\rangle & \in & \mathbf{N}^* & \equiv \mathsf{T} \\
\langle\langle 1,2\rangle, \langle 3,4\rangle\rangle & \in & \mathbf{N}^{**} & \equiv \mathsf{T}
\end{array}
$$

$[\mathbf{E}]^{\cdot}$ the set of the empty list
$[\mathsf{x}^*]^{\cdot}$ x listset

Exercise 10.9.1 What are the values of the following:

a $[\, \langle 1, 1\mathrm{F}, 2\rangle \in \mathbf{D}^* \,]^{\cdot}$.

b $[\, \langle 1, 1\mathrm{F}, 2\rangle \in (\mathbf{D}^{\cdot}_{\infty})^* \,]^{\cdot}$.

c $[\, \langle 1, 1\mathrm{F}, \bullet\rangle \in \mathbf{D}^* \,]^{\cdot}$.

d $[\, \langle 1, 1\mathrm{F}, \bot\rangle \in \mathbf{D}^* \,]^{\cdot}$.

10.10 Infinite lists are non-classical

I define $[\, \mathrm{wf}(\mathsf{x}) \,]^{\cdot}$ thus:

$$\left[\ \boxed{\mathrm{wf}(\mathsf{x})} \doteq \mathsf{x} \left\langle \begin{array}{l} \mathsf{T} \\ \mathrm{wf}(\mathsf{x}\ \mathrm{Head})\ \tilde{\wedge}\ \mathrm{wf}(\mathsf{x}\ \mathrm{Tail}) \end{array} \right. \right]^{\cdot}$$

For all $[\, \mathsf{x} \,]^{\cdot}$, $[\, \mathrm{wf}(\mathsf{x}) \,]^{\cdot}$ equals $[\, \mathsf{T} \,]^{\cdot}$ or $[\, \bot \,]^{\cdot}$. $[\, \mathrm{wf}(\mathsf{x}) \,]^{\cdot}$ equals $[\, \mathsf{T} \,]^{\cdot}$ for maps that can be written using $[\, \mathsf{N} \,]^{\cdot}$ and $[\, \mathsf{x} \therefore \mathsf{y} \,]^{\cdot}$ a finite number of times. Here you have some examples:

$$\begin{array}{lcl}
[\, \mathrm{wf}(\mathsf{N}) & \equiv & \mathsf{T} \,]^{\cdot} \\
[\, \mathrm{wf}(\mathsf{N} \therefore \mathsf{N}) & \equiv & \mathsf{T} \,]^{\cdot} \\
[\, \mathrm{wf}((\mathsf{N} \therefore \mathsf{N}) \therefore \mathsf{N}) & \equiv & \mathsf{T} \,]^{\cdot} \\
[\, \mathrm{wf}(\mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N})) & \equiv & \mathsf{T} \,]^{\cdot} \\
[\, \mathrm{wf}(\mathsf{T}) & \equiv & \mathsf{T} \,]^{\cdot} \\
[\, \mathrm{wf}(\mathsf{F}) & \equiv & \mathsf{T} \,]^{\cdot} \\
[\, \mathrm{wf}(\bullet) & \equiv & \mathsf{T} \,]^{\cdot} \\
[\, \mathrm{wf}(117) & \equiv & \mathsf{T} \,]^{\cdot} \\
[\, \mathrm{wf}(\langle 1, 2, 3\rangle) & \equiv & \mathsf{T} \,]^{\cdot} \\
[\, \mathrm{wf}(\infty\text{-list}(1)) & \equiv & \bot \,]^{\cdot} \\
[\, \mathrm{wf}(\bot) & \equiv & \bot \,]^{\cdot}
\end{array}$$

You need to know the following about classicallity:

Fact 10.10.1 If $[\, \mathrm{wf}(\mathsf{x}) \equiv \bot \,]^{\cdot}$ then $[\, \mathsf{x} \,]^{\cdot}$ is non-classical.

As an example, $[\, \mathrm{wf}(\infty\text{-list}(1)) \equiv \bot \,]^{\cdot}$ so $[\, \infty\text{-list}(1) \,]^{\cdot}$ is non-classical. Hence:

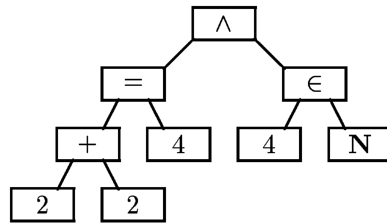
$$[\, \infty\text{-list}(1) \in \mathbf{N}^* \equiv \bot \,]^{\cdot}.$$

In other words, $[\, \mathbf{N}^* \,]^{\cdot}$ is the set of finite lists of natural numbers. It does not contain infinite lists like $[\, \infty\text{-list}(1) \,]^{\cdot}$.

$[\, \mathrm{wf}(\mathrm{x}) \,]^{\cdot}$ wellfounded x end

10.11 Answers

Answer 10.1.1 The value is $[\,\mathsf{T}\,]^{\cdot}$. The tree reads:



Answer 10.2.1 (a) $[\,\mathsf{T}\,]^{\cdot}$. (b) $[\,\mathsf{F}\,]^{\cdot}$. (c) $[\,\mathsf{T}\,]^{\cdot}$. (d) $[\,\mathsf{T}\,]^{\cdot}$. (e) $[\,\bot\,]^{\cdot}$.

Answer 10.3.1 (a) Yes. Using the definitions of $[\,\bullet\,]^{\cdot}$ and $[\,\mathsf{x} :: \mathsf{y}\,]^{\cdot}$ you can write $[\,\bullet :: \bullet\,]^{\cdot}$ using the constructs $[\,\mathsf{N}\,]^{\cdot}$ and $[\,\mathsf{x} \therefore \mathsf{y}\,]^{\cdot}$, so $[\,\bullet :: \bullet\,]^{\cdot}$ is classical according to $[\,\text{ClassicalNil}\,]^{\cdot}$ and $[\,\text{ClassicalPair}\,]^{\cdot}$. (b) No. Using the definition of $[\,\mathsf{x} :: \mathsf{y}\,]^{\cdot}$ you can write $[\,\bot :: \bot\,]^{\cdot}$ using the constructs $[\,\mathsf{N}\,]^{\cdot}$, $[\,\bot\,]^{\cdot}$, and $[\,\mathsf{x} \therefore \mathsf{y}\,]^{\cdot}$. If you do so, the expression will contain $[\,\bot\,]^{\cdot}$ at least once (exactly twice, actually). Hence, $[\,\bot :: \bot\,]^{\cdot}$ is non-classical according to $[\,\text{ClassicalBottom}\,]^{\cdot}$ and $[\,\text{ClassicalPair}\,]^{\cdot}$.

Answer 10.4.1

a $[\,\mathsf{N} \in \mathbf{B} \equiv \mathsf{T}\,]^{\cdot}$.

b $[\,\mathsf{N} \therefore \mathsf{N} \in \mathbf{B} \equiv \mathsf{T}\,]^{\cdot}$.

c $[\,(\mathsf{N} \therefore \mathsf{N}) \therefore \mathsf{N} \in \mathbf{B} \equiv \mathsf{F}\,]^{\cdot}$.

d $[\,\mathsf{N} \therefore (\mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N})) \in \mathbf{B} \equiv \mathsf{T}\,]^{\cdot}$.

e $[\,(\mathsf{N} \therefore \bot) \therefore \mathsf{N} \in \mathbf{B} \equiv \bot\,]^{\cdot}$.

f $[\,\mathsf{N} \therefore (\mathsf{N} \therefore \bot) \in \mathbf{B} \equiv \bot\,]^{\cdot}$.

Answer 10.5.1

a $[\,\mathsf{T} \in \mathbf{T} \equiv \mathsf{T}\,]^{\cdot}$.

b $[\,\mathsf{F} \in \mathbf{T} \equiv \mathsf{F}\,]^{\cdot}$.

c $[\,\bullet \in \mathbf{T} \equiv \mathsf{F}\,]^{\cdot}$.

d $[\,\bot \in \mathbf{T} \equiv \bot\,]^{\cdot}$.

e $[\,\mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N}) \in \mathbf{T} \equiv \mathsf{F}\,]^{\cdot}$.

f $[\,\mathsf{N} \therefore (\mathsf{N} \therefore \mathsf{N}) \in \mathbf{F} \equiv \mathsf{T}\,]^{\cdot}$.

g $[\,\mathsf{N} \therefore (\bot \therefore \mathsf{N}) \in \mathbf{T} \equiv \bot\,]^{\cdot}$.

h $[\ \mathsf{N} \therefore (\bot \therefore \mathsf{N}) \in \mathbf{F} \equiv \bot\ ]^{\cdot}$.

Answer 10.6.1

a $[\ 12\text{F} \in \mathbf{D}^{\cdot}_2 \equiv \mathsf{T}\ ]^{\cdot}$

b $[\ \infty@3 \in \mathbf{D}^{\infty}_3 \equiv \mathsf{T}\ ]^{\cdot}$

c $[\ 12\text{F} \in \mathbf{N} \equiv \mathsf{F}\ ]^{\cdot}$

d $[\ 12 \in \mathbf{Z} \equiv \mathsf{T}\ ]^{\cdot}$

e $[\ -12 \in \mathbf{Z}^{-} \equiv \mathsf{T}\ ]^{\cdot}$

f $[\ 0 \in \mathbf{N} \equiv \mathsf{T}\ ]^{\cdot}$

g $[\ 0 \in \mathbf{Z}^{+} \equiv \mathsf{F}\ ]^{\cdot}$

h $[\ 0 \in \mathbf{Z}^{-} \equiv \mathsf{F}\ ]^{\cdot}$

Answer 10.7.1

a $[\ (1\text{F} :: 1.2\text{F}) :: 1.23\text{F} \in (\mathbf{D}^{\cdot}_1 \times \mathbf{D}^{\cdot}_2) \times \mathbf{D}^{\cdot}_3 \equiv \mathsf{T}\ ]^{\cdot}$.

b $[\ 1\text{F} :: (1.2\text{F} :: 1.23\text{F}) \in (\mathbf{D}^{\cdot}_1 \times \mathbf{D}^{\cdot}_2) \times \mathbf{D}^{\cdot}_3 \equiv \mathsf{F}\ ]^{\cdot}$.

c $[\ 1\text{F} :: 1.2\text{F} :: 1.23\text{F} \in \mathbf{D}^{\cdot}_1 \times \mathbf{D}^{\cdot}_2 \times \mathbf{D}^{\cdot}_3 \equiv \mathsf{T}\ ]^{\cdot}$.

Answer 10.8.1 $[\ \mathsf{T}\ ]^{\cdot}$.

Answer 10.9.1

a $[\ \langle 1, 1\text{F}, 2\rangle \in \mathbf{D}^{*} \equiv \mathsf{T}\ ]^{\cdot}$.

b $[\ \langle 1, 1\text{F}, 2\rangle \in (\mathbf{D}^{\cdot}_{\infty})^{*} \equiv \mathsf{F}\ ]^{\cdot}$.

c $[\ \langle 1, 1\text{F}, \bullet\rangle \in \mathbf{D}^{*} \equiv \mathsf{F}\ ]^{\cdot}$.

d $[\ \langle 1, 1\text{F}, \bot\rangle \in \mathbf{D}^{*} \equiv \bot\ ]^{\cdot}$.

Chapter 11

Derivations

11.1 Introduction

You saw algebraic proofs in Chapter 6. Algebraic proofs are convenient for reasoning about recursive functions. In principle, algebraic proofs are all you need to develop any mathematical subject. There are, however, other kinds of proofs that are more convenient in some situations. In this chapter I will show you another kind of proof, namely the *Hilbert style proof*. I will refer to Hilbert style proofs as *derivations* in order to save a little ink.

11.2 Derivation rules

If $[\,x\,]^{\cdot}$, $[\,y\,]^{\cdot}$, and $[\,z\,]^{\cdot}$ are terms, and if $[\,x \equiv y\,]^{\cdot}$ and $[\,y \equiv z\,]^{\cdot}$, then $[\,x \equiv z\,]^{\cdot}$. I will state this thus:

[**Mac rule Transitivity** : $x \equiv y \boxed{\vdash} y \equiv z \vdash x \equiv z$]

The statement above is an example of a *derivation rule*. Here you have another derivation rule:

[**Mac rule TypeN+N** : $x \in \mathbf{N} \equiv \mathsf{T} \vdash y \in \mathbf{N} \equiv \mathsf{T} \vdash x+y \in \mathbf{N} \equiv \mathsf{T}$]

The derivation rule says that if $[\,x\,]^{\cdot}$ and $[\,y\,]^{\cdot}$ are natural numbers, then $[\,x+y\,]^{\cdot}$ is a natural number.

In the derivation rule $[\,x \equiv y \vdash y \equiv z \vdash x \equiv z\,]^{\cdot}$, I call $[\,x \equiv y\,]^{\cdot}$ and $[\,y \equiv z\,]^{\cdot}$ the *premisses* of the rule and I call $[\,x \equiv z\,]^{\cdot}$ the *conclusion* of the rule.

Hilbert style proof
derivation
$[\,x \vdash y\,]^{\cdot}$ x infers y
derivation rule
premisses
conclusion

Any derivation rule has exactly one conclusion and zero, one or more premisses. The premisses and conclusion of a derivation are equations, i.e. they have the form [x ≡ y]˙ where [x]˙ and [y]˙ are terms.

Here you have a derivation rule with one premise:

[**Mac rule Commutativity** : x ≡ y ⊢ y ≡ x]

The [Commutativity]˙ rule says that if [x]˙ equals [y]˙ then [y]˙ equals [x]˙. Here you have one with zero premisses:

[**Mac rule HeadPair** : (x :: y) head ≡ x]

As you can see, a derivation rule with zero premisses is simply an equation that holds. In other words, a derivation rule with zero premisses is the same as an algebraic rule.

The litterature uses the word *axiom* to denote derivation rules with zero premisses and the phrase *inference rule* to denote derivation rules with at least one premisse. I use "derivation rule" or just *rule* as a common name for axioms and inference rules.

In Section 6.6 I stated that you could take "axiom" and "rule" as synonyms until Section 11.2. Now you read Section 11.2, and time has come for an adjustment: An axiom is a rule with zero premisses. An inference rule is a rule with one or more premisses. A rule is an axiom or an inference rule.

11.3 Instantiation statements

TailPair says that [(x :: y) tail ≡ y]˙. If you replace the variables of a derivation rule by terms, then the result is an *instance* of the rule. As an example,

[(2 :: 1) tail ≡ 1]˙

is an instance of TailPair. I will use

[TailPair ⊳ (2 :: 1) tail ≡ 1]˙

to denote that [(2 :: 1) tail ≡ 1]˙ is an instance of TailPair.

I will refer to statements like [TailPair ⊳ (2 :: 1) tail ≡ 1]˙ as *instantiation statements*. An instantiation statement consists of an *argumentation* and a *conclusion*. In [TailPair ⊳ (2 :: 1) tail ≡ 1]˙, TailPair is the argumentation and [(2 :: 1) tail ≡ 1]˙ is the conclusion.

---

	axiom
	inference rule
	rule
	instance of derivation rule
[x ⊳ y]˙	x concludes y
	instantiation statement
	argumentation
	conclusion

Here you have some further instantiation statements:

$$
\begin{array}{lll}
[\,\mathrm{TailPair} & \rhd & (\mathsf{u} :: \mathsf{v})\ \mathrm{tail} \equiv \mathsf{v}\,]^{\cdot} \\
[\,\mathrm{TailPair} & \rhd & (\mathsf{x} :: \mathsf{y})\ \mathrm{tail} \equiv \mathsf{y}\,]^{\cdot} \\
[\,\mathrm{TailPair} & \rhd & (\mathsf{y} :: \mathsf{x})\ \mathrm{tail} \equiv \mathsf{x}\,]^{\cdot} \\
[\,\mathrm{TailPair} & \rhd & (\mathsf{x} :: \mathsf{x})\ \mathrm{tail} \equiv \mathsf{x}\,]^{\cdot} \\
[\,\mathrm{TailPair} & \rhd & (2+2 :: 3+3)\ \mathrm{tail} \equiv 3+3\,]^{\cdot} \\
[\,\mathrm{HeadPair} & \rhd & (1 :: 2)\ \mathrm{head} \equiv 1\,]^{\cdot}
\end{array}
$$

Commutativity says that if $[\,\mathsf{x} \equiv \mathsf{y}\,]^{\cdot}$ then $[\,\mathsf{y} \equiv \mathsf{x}\,]^{\cdot}$ for all terms $[\,\mathsf{x}\,]^{\cdot}$ and $[\,\mathsf{y}\,]^{\cdot}$. As an example, Commutativity says that if

$$[\,(2 :: 1)\ \mathrm{tail} \equiv 1\,]^{\cdot}$$

then

$$[\,1 \equiv (2 :: 1)\ \mathrm{tail}\,]^{\cdot}.$$

I will use the instantiation statement

$$[\,\mathrm{Commutativity} \rhd (2 :: 1)\ \mathrm{tail} \equiv 1 \rhd 1 \equiv (2 :: 1)\ \mathrm{tail}\,]^{\cdot}$$

to denote that. The argumentation of the instantiation statement reads

$$[\,\mathrm{Commutativity} \rhd (2 :: 1)\ \mathrm{tail} \equiv 1\,]^{\cdot}$$

and the conclusion reads

$$[\,1 \equiv (2 :: 1)\ \mathrm{tail}\,]^{\cdot}.$$

Transitivity says that if $[\,\mathsf{x} \equiv \mathsf{y}\,]^{\cdot}$ and $[\,\mathsf{y} \equiv \mathsf{z}\,]^{\cdot}$ then $[\,\mathsf{x} \equiv \mathsf{z}\,]^{\cdot}$. As an example, Transitivity says that if

$$[\,(1 :: 2)\ \mathrm{head} \equiv 1\,]^{\cdot}$$

and

$$[\,1 \equiv (2 :: 1)\ \mathrm{tail}\,]^{\cdot}$$

then

$$[\,(1 :: 2)\ \mathrm{head} \equiv (2 :: 1)\ \mathrm{tail}\,]^{\cdot}.$$

I will use the instantiation statement

$$
\begin{array}{llll}
[& \mathrm{Transitivity} & \rhd & \\
 & (1 :: 2)\ \mathrm{head} \equiv 1 & \rhd & \\
 & 1 \equiv (2 :: 1)\ \mathrm{tail} & \rhd & (1 :: 2)\ \mathrm{head} \equiv (2 :: 1)\ \mathrm{tail}\,]^{\cdot}
\end{array}
$$

to denote that. The argumentation of the instantiation statement reads

$$[\,\mathrm{Transitivity} \rhd (1 :: 2)\ \mathrm{head} \equiv 1 \rhd 1 \equiv (2 :: 1)\ \mathrm{tail}\,]^{\cdot}$$

and the conclusion reads

$$[\,(1 :: 2)\ \mathrm{head} \equiv (2 :: 1)\ \mathrm{tail}\,]^{\cdot}.$$

Exercise 11.3.1 Which of the following are correct instantiations:

a $[\ \mathrm{Commutativity} \rhd 2+2 \equiv 4 \rhd 4 \equiv 2+2\ ]^{\cdot}$.

b $[\ \mathrm{Commutativity} \rhd 2+2 \equiv 4 \rhd 1+3 \equiv 2+2\ ]^{\cdot}$.

c $[\ \mathrm{Commutativity} \rhd 2+2 \equiv 5 \rhd 5 \equiv 2+2\ ]^{\cdot}$.

d $[\ \mathrm{TailPair} \rhd (1+2 :: 3+4)\ \mathrm{tail} \equiv 3+4\ ]^{\cdot}$.

e $[\ \mathrm{TailPair} \rhd (1 :: 2)\ \mathrm{tail} + 3 \equiv 2+3\ ]^{\cdot}$.

11.4 Derivation proofs

Here you have a lemma and a *derivation proof*:

[**Mac lemma L11.4.1** : $(1 :: 2)\ \mathrm{head} \equiv (2 :: 1)\ \mathrm{tail}$]

[**Mac proof of** L11.4.1:

HeadPair $\rhd$	$(1 :: 2)\ \mathrm{head} \equiv 1$	;
TailPair $\rhd$	$(2 :: 1)\ \mathrm{tail} \equiv 1$	;
Commutativity $\rhd$ $(2 :: 1)\ \mathrm{tail} \equiv 1 \rhd$	$1 \equiv (2 :: 1)\ \mathrm{tail}$	;
Transitivity $\rhd$ $(1 :: 2)\ \mathrm{head} \equiv 1 \rhd$ $1 \equiv (2 :: 1)\ \mathrm{tail} \rhd$	$(1 :: 2)\ \mathrm{head} \equiv (2 :: 1)\ \mathrm{tail}$	]

The proof consists of four *proof lines* separated by semicolons. Each proof line is an instantiation statement. The conclusion of the last proof line is identical to the lemma to be proved. Furthermore, each premise that occurs in an argumentation is identical to the conclusion of a previous proof line.

In general, a derivation proof is a non-empty list of instantiation statements. I refer to the conclusion of the last instantiation statement as the *conclusion of the proof*. A proof is *correct* if each proof line is a correct instantiation statement and, furthermore, every premise of the argumentation of every proof line is identical to the conclusion of a previous proof line.

11.5 Labelled proof lines

If $[\ \mathsf{x}\ ]^{\cdot}$ is a variable and $[\ \mathsf{y}\ ]^{\cdot}$ is an instantiation statement, then

$[\ \mathsf{x} \boxed{:} \mathsf{y}\ ]^{\cdot}$

---

	derivation proof
	proof line
	conclusion of proof
	correct proof
$[\ \mathsf{x} : \mathsf{y}\ ]^{\cdot}$	x colon y

means the same as [y]˙, but also introduces [x]˙ as shorthand for the conclusion of [y]˙. As an example, here you have another proof of L11.4.1:

[**Mac proof of** L11.4.1:
x : HeadPair ▷ (1 :: 2) head ≡ 1 ;
y : TailPair ▷ (2 :: 1) tail ≡ 1 ;
z : Commutativity ▷
(2 :: 1) tail ▷ 1 ≡ (2 :: 1) tail ;
u : Transitivity ▷
(1 :: 2) head ≡ 1 ▷
1 ≡ (2 :: 1) tail ▷ (1 :: 2) head ≡ (2 :: 1) tail]

Inside the proof,
[x]˙ is shorthand for [(1 :: 2) head ≡ 1]˙,
[y]˙ is shorthand for [(2 :: 1) tail ≡ 1]˙,
[z]˙ is shorthand for [1 ≡ (2 :: 1) tail]˙, and
[u]˙ is shorthand for [(1 :: 2) head ≡ (2 :: 1) tail]˙.
I can use this to state the proof shorter:

[**Mac proof of** L11.4.1:
x : HeadPair ▷ (1 :: 2) head ≡ 1 ;
y : TailPair ▷ (2 :: 1) tail ≡ 1 ;
z : Commutativity ▷ y ▷ 1 ≡ (2 :: 1) tail ;
u : Transitivity ▷ x ▷ z ▷ (1 :: 2) head ≡ (2 :: 1) tail]

11.6 Indexed variables

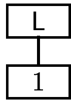
The statement

[**variable** x]

declares that [x]˙ is a variable. Statements like the one above allows to declare variables one by one. However, the statement

[**variable** [L[n]]]

declares infinitely many variables in one go, namely the variables [L1]˙, [L2]˙, [L3]˙, and so on. Actually, you may replace [n]˙ above by any term so that [L1.2F]˙ and [L∞]˙ are also variables. The syntax tree for [L1]˙ reads



[**variable** [L[n]]] declares [L]˙ as a unary operator and declares that any syntax tree with [L]˙ in the root denotes a variable. Two syntax trees with [L]˙ in the

[**variable** x]˙ variable x
[Lx]˙ indexed capital l x

root denote the same variable if the syntax trees are identical. As an example [L4]˙ and [L(2 + 2)]˙ denote two distinct variables.

I assign maximum priority (c.f. Section A.2) to [Lx]˙ so that e.g. [L2 + 2]˙ means [(L2) + 2]˙.

Indexed variables are convenient when stating proofs. Here you have one more proof of L11.4.1:

[**Mac proof of** L11.4.1**:**

L1 :	HeadPair ⊳	$(1 :: 2)$ head $\equiv 1$	;
L2 :	TailPair ⊳	$(2 :: 1)$ tail $\equiv 1$	;
L3 :	Commutativity ⊳ L2 ⊳	$1 \equiv (2 :: 1)$ tail	;
L4 :	Transitivity ⊳ L1 ⊳ L3 ⊳	$(1 :: 2)$ head $\equiv (2 :: 1)$ tail	]

Exercise 11.6.1 Prove the following:

[**Mac lemma L11.6.2** : $(x :: y)$ head $\equiv (x :: z)$ head]

11.7 Instantiation and binding constructs

Here you have a derivation rule that involves a binding construct:

[**Mac rule LambdaSub** : $x \equiv y \vdash \lambda z.x \equiv \lambda z.y$]

The derivation rule expresses substitution of equals under a lambda. In the derivation rule, [x]˙, [y]˙, and [z]˙ denote arbitrary terms by default. however, [λz.x]˙ and [λz.y]˙ require [z]˙ to be a variable. Hence, [x]˙ and [y]˙ denote arbitrary terms and [z]˙ denotes an arbitrary variable. Here you have an example of an instantiation:

[LambdaSub ⊳ $(u :: v)$ head $\equiv u$ ⊳ $\lambda u.(u :: v)$ head $\equiv \lambda u.u$]˙.

In the instantiation, I have replaced [x]˙ by [(u :: v) head]˙, and [y]˙ and [z]˙ by [u]˙. The variable that you substitute for [z]˙ may or may not occur free in the terms you substitute for [x]˙ and [y]˙.

Instantiation statements allow renaming of bound variables, so here you have another instantiation:

[LambdaSub ⊳ $(u :: v)$ head $\equiv u$ ⊳ $\lambda a.(a :: v)$ head $\equiv \lambda b.b$]˙.

Exercise 11.7.1 Which of the following are correct instantiations:

a [Reflexivity ⊳ $\lambda u.u \equiv \lambda v.v$]˙.

b [$(\lambda x.x \text{ ' } x) \text{ ' } y \equiv y \text{ ' } y$ ⊳ $(\lambda y.y \text{ ' } y) \text{ ' } 2 \equiv 2 \text{ ' } 2$]˙.

c [$(\lambda x.x \text{ ' } x) \text{ ' } y \equiv y \text{ ' } y$ ⊳ $(\lambda 2.2 \text{ ' } 2) \text{ ' } y \equiv y \text{ ' } y$]°.

d [$(y \equiv \lambda x.y \text{ ' } x \vdash \text{case}(y, u, v) \equiv v)$ ⊳ $F \equiv \lambda L1.F \text{ ' } L1$ ⊳ $\text{case}(F, u, v) \equiv v$]˙.

11.8 Substitution

In instantiation statements I define $[\,\langle x \mid y{:=}z\rangle\,]^{\cdot}$ to have the special property that you may perform substitutions during instantiation. As an example, consider the following rule:

[**Mac rule ApplyLambda** : $(\lambda x.\mathcal{A})\ '\ \mathcal{S} \equiv \langle \mathcal{A} \mid x{:=}\mathcal{S}\rangle$]

Here you have an instantiation of that rule:

$$[\ \text{ApplyLambda} \rhd (\lambda x.x+3)\ '\ 2 \equiv \langle x+3 \mid x{:=}2\rangle\ ]^{\cdot}.$$

However, since you are allowed to perform substitutions during instantiation, the following is also a correct instantiation:

$$[\ \text{ApplyLambda} \rhd (\lambda x.x+3)\ '\ 2 \equiv 2+3\ ]^{\cdot}.$$

In general, you may do the following when instantiating derivation rules:

1. You may replace variables in the rule by terms.
2. You may rename bound variables.
3. You may perform substitutions that occur in the rule.

Exercise 11.8.1 Which of the following are correct instantiations:

(a) $[\ (x \equiv y \vdash \langle x \mid u{:=}2+2\rangle \equiv \langle y \mid u{:=}4\rangle) \rhd (u :: v)\ \text{head} \equiv u \rhd (2+2 :: v)\ \text{head} \equiv 4\ ]^{\cdot}$

(b) $[\ (\langle a \mid x{:=}N\rangle \equiv \langle b \mid x{:=}N\rangle \vdash \text{case}(x,a,c) \equiv \text{case}(x,b,c)) \rhd 1+N \equiv 2+N \rhd \text{case}(u,1+u,\bot) \equiv \text{case}(u,2+u,\bot)\ ]^{\cdot}.$

(c) $[\ (x \equiv y \vdash y \equiv x) \rhd 2+2 \equiv \langle x \mid x{:=}4\rangle \rhd 4 \equiv 2+2\ ]^{\cdot}.$

Exercise 11.8.2 Prove the following:

[**Mac lemma L11.8.3** : $(\lambda x.x)\ '\ ((\lambda y.y)\ '\ N) \equiv N$]

11.9 Terms that represent formulas

Whenever a term $[\ \mathcal{A}\ ]^{\cdot}$ occurs in a position where a formula is expected, that term is shorthand for $[\ \mathcal{A} \equiv \mathsf{T}\ ]^{\cdot}$.

Exercise 11.9.1 Is the following lemma and proof correct:

[**Mac lemma L11.9.2** : $((\mathsf{T} :: \mathsf{T})\ \text{head} :: \mathsf{T})\ \text{head}$]$^{\cdot}$

[**Mac proof of** L11.9.2**:**

L1 :	HeadPair $\rhd$	$((\mathsf{T}::\mathsf{T})\ \text{head}::\mathsf{T})\ \text{head} \equiv (\mathsf{T}::\mathsf{T})\ \text{head}$	;
L2 :	HeadPair $\rhd$	$(\mathsf{T}::\mathsf{T})\ \text{head}$	;
L3 :	Transitivity $\rhd$ L1 $\rhd$ L2 $\rhd$	$((\mathsf{T}::\mathsf{T})\ \text{head}::\mathsf{T})\ \text{head}$	]$^{\cdot}$

Exercise 11.9.3 What does the following lemma say? Prove the lemma.

[**Mac lemma L11.9.4** : $(\lambda x.x)\ '\ ((\lambda y.y)\ '\ \mathsf{T})$]

11.10 Type inference

In Section A.43 and Section A.13 you can read the following mac rules about sets:

[**Mac rule TypeTInB** : $\mathsf{T} \in \mathbf{B}$]

[**Mac rule TypeFInB** : $\mathsf{F} \in \mathbf{B}$]

Written out in full (using the convention in Section 11.9), the derivation rules read:

[**Mac rule TypeTInB** : $\mathsf{T} \in \mathbf{B} \equiv \mathsf{T}$]

[**Mac rule TypeFInB** : $\mathsf{F} \in \mathbf{B} \equiv \mathsf{T}$]

If you look in the "Type table" section of Section A.46 you find the following table:

Type table

$x \wedge y$	$\mathbf{B}$	$\mathbf{D}$	$\mathbf{X}$	$\mathbf{E}$	$x \wedge y$	$\mathbf{N}$	$\mathbf{Z}$	$\mathbf{Z}^+$	$\mathbf{Z}^-$
$\mathbf{B}$	$\mathbf{B}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{N}$	$\mathbf{N}$	$\mathbf{Z}$	$\mathbf{N}$	$\mathbf{Z}^-$
$\mathbf{D}$	$\mathbf{X}$	–	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{Z}$	$\mathbf{Z}$	$\mathbf{Z}$	$\mathbf{Z}$	$\mathbf{Z}^-$
$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{Z}^+$	$\mathbf{N}$	$\mathbf{Z}$	$\mathbf{Z}^+$	$\mathbf{Z}^-$
$\mathbf{E}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{Z}^-$	$\mathbf{Z}^-$	$\mathbf{Z}^-$	$\mathbf{Z}^-$	$\mathbf{Z}^-$
$x \wedge y$	$\mathbf{D}^{\infty}_{m}$	$\mathbf{D}^{\cdot}_{m}$	$\mathbf{D}^{-\infty}_{m}$	$\mathbf{D}^{\infty}_{\infty}$	$\mathbf{D}^{\cdot}_{\infty}$	$\mathbf{D}^{-\infty}_{\infty}$	$\mathbf{D}^{\infty}_{n}$	$\mathbf{D}^{\cdot}_{n}$	$\mathbf{D}^{-\infty}_{n}$
$\mathbf{D}^{\infty}_{m}$	$\mathbf{D}^{\infty}_{m}$	$\mathbf{D}^{\cdot}_{m}$	$\mathbf{D}^{-\infty}_{m}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$
$\mathbf{D}^{\cdot}_{m}$	$\mathbf{D}^{\cdot}_{m}$	$\mathbf{D}^{\cdot}_{m}$	$\mathbf{D}^{-\infty}_{m}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$
$\mathbf{D}^{-\infty}_{m}$	$\mathbf{D}^{-\infty}_{m}$	$\mathbf{D}^{-\infty}_{m}$	$\mathbf{D}^{-\infty}_{m}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$
$\mathbf{D}^{\infty}_{\infty}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{D}^{\infty}_{\infty}$	$\mathbf{D}^{\cdot}_{\infty}$	$\mathbf{D}^{-\infty}_{\infty}$	$x \wedge y$	$\mathbf{T}$	$\mathbf{F}$
$\mathbf{D}^{\cdot}_{\infty}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{D}^{\cdot}_{\infty}$	$\mathbf{D}^{\cdot}_{\infty}$	$\mathbf{D}^{-\infty}_{\infty}$	$\mathbf{T}$	$\mathbf{T}$	$\mathbf{F}$
$\mathbf{D}^{-\infty}_{\infty}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{X}$	$\mathbf{D}^{-\infty}_{\infty}$	$\mathbf{D}^{-\infty}_{\infty}$	$\mathbf{D}^{-\infty}_{\infty}$	$\mathbf{F}$	$\mathbf{F}$	$\mathbf{F}$

That table consists of four sub-tables that contain 16, 16, 45, and 4 entries, respectively, adding up to a total of 81 entries. That table represents 81 mac rules (actually, the table merely represents 80 rules, as I explain soon).

As an example of reading the table, look near the upper left corner of the table where the row labelled [**B**]˙ meets the column labelled [**D**]˙. There you will find the character [**X**]˙. That combination of [**B**]˙, [**D**]˙, and [**X**]˙ represents the following mac rule:

[**Mac rule TypeB∧D** : $x \in \mathbf{B} \vdash y \in \mathbf{D} \vdash (x \wedge y) \in \mathbf{X}$]

The rule says that if [x]˙ is a truth value and [y]˙ is a decimal fraction, then [x ∧ y]˙ is an exception.

If you look where the row labelled [**D**]˙ meets the column labelled [**D**]˙ you find a dash. That dash indicates that the table does not provide a rule of form

$[\, \mathsf{x} \in \mathbf{D} \vdash \mathsf{y} \in \mathbf{D} \vdash (\mathsf{x} \wedge \mathsf{y}) \in \mathcal{A} \,]^{\cdot}$. That is why the table represents 80 rather than 81 rules.

Here you have two rules you can look up in the “Tables” section of Section A.46 and Section A.77:

[**Mac rule TypeB∧B** : $\mathsf{x} \in \mathbf{B} \vdash \mathsf{y} \in \mathbf{B} \vdash (\mathsf{x} \wedge \mathsf{y}) \in \mathbf{B}$]

[**Mac rule TypeB∨B** : $\mathsf{x} \in \mathbf{B} \vdash \mathsf{y} \in \mathbf{B} \vdash (\mathsf{x} \vee \mathsf{y}) \in \mathbf{B}$]

Written out in full (using the convention in Section 11.9), the derivation rules read:

[**Mac rule TypeB∧B** : $\mathsf{x} \in \mathbf{B} \equiv \mathsf{T} \vdash \mathsf{y} \in \mathbf{B} \equiv \mathsf{T} \vdash (\mathsf{x} \wedge \mathsf{y}) \in \mathbf{B} \equiv \mathsf{T}$]

[**Mac rule TypeB∨B** : $\mathsf{x} \in \mathbf{B} \equiv \mathsf{T} \vdash \mathsf{y} \in \mathbf{B} \equiv \mathsf{T} \vdash (\mathsf{x} \vee \mathsf{y}) \in \mathbf{B} \equiv \mathsf{T}$]

In Appendix A you can find a zillion derivation rules about sets. Here you have a lemma with a proof that uses the derivation rules:

[**Mac lemma L11.10.1** : $(\mathsf{T} \wedge \mathsf{F} \vee \mathsf{T}) \in \mathbf{B}$]

[**Mac proof of** L11.10.1:

L1 :	TypeTInB ▷	$\mathsf{T} \in \mathbf{B}$	;
L2 :	TypeFInB ▷	$\mathsf{F} \in \mathbf{B}$	;
L3 :	Type**B**∧**B** ▷ L1 ▷ L2 ▷	$(\mathsf{T} \wedge \mathsf{F}) \in \mathbf{B}$	;
L4 :	Type**B**∨**B** ▷ L3 ▷ L1 ▷	$(\mathsf{T} \wedge \mathsf{F} \vee \mathsf{T}) \in \mathbf{B}$	]

Written out in full, [L11.10.1]˙ and its proof read:

[**Mac lemma L11.10.1** : $\mathsf{T} \wedge \mathsf{F} \vee \mathsf{T} \in \mathbf{B} \equiv \mathsf{T}$]

[**Mac proof of** L11.10.1:

L1 :	TypeTInB ▷	$\mathsf{T} \in \mathbf{B} \equiv \mathsf{T}$	;
L2 :	TypeFInB ▷	$\mathsf{F} \in \mathbf{B} \equiv \mathsf{T}$	;
L3 :	Type**B**∧**B** ▷ L1 ▷ L2 ▷	$(\mathsf{T} \wedge \mathsf{F}) \in \mathbf{B} \equiv \mathsf{T}$	;
L4 :	Type**B**∨**B** ▷ L3 ▷ L1 ▷	$(\mathsf{T} \wedge \mathsf{F} \vee \mathsf{T}) \in \mathbf{B} \equiv \mathsf{T}$	]

11.11 Numeral types

As I mentioned in Section 7.6, numerals are constructs that consist of sequences of digits, possibly with a decimal point [.]° and a floating point mark [F]°. [117]˙ and [1.234F]˙ are numerals. [∞]˙ and [$2 + 3$]˙ are not numerals.

[TypeNumeralInN]˙ says that [$\mathsf{x} \in \mathbf{N}$]˙ equals [T]˙ for all numerals that denote a natural number. [TypeNumeralInN]˙ is one among several *numeral type rules*.

Here you have a lemma with a proof that uses numeral types:

---

numeral type rule

[**Mac lemma L11.11.1** : $1+2+3 \in \mathbf{N}$]

[**Mac proof of** L11.11.1**:**
L1 : TypeNumeralInN ⊳ $1 \in \mathbf{N}$;
L2 : TypeNumeralInN ⊳ $2 \in \mathbf{N}$;
L3 : TypeNumeralInN ⊳ $3 \in \mathbf{N}$;
L4 : Type**N**+**N** ⊳ L1 ⊳ L2 ⊳ $1+2 \in \mathbf{N}$;
L5 : Type**N**+**N** ⊳ L4 ⊳ L3 ⊳ $1+2+3 \in \mathbf{N}$]

[TypeNumeralNotInN]˙ says that if [x]˙ is a numeral that does not denote a natural number then [$x \notin \mathbf{N}$]˙. You can find [TypeNumeralInN]˙ and [TypeNumeralNotInN]˙ under "the set of natural numbers" in the reference manual (Section A.37), and you can find many other numeral type rules in Appendix A.

11.12 Floating point type inference

In Section A.80 you find the following type table for [$x+y$]˙:

Type table

$x+y$	**B**	**D**	**X**	**E**	$x+y$	**N**	**Z**	$\mathbf{Z}^+$	$\mathbf{Z}^-$
B	**B**	**X**	**X**	**X**	**N**	**N**	**Z**	**N**	**Z**
D	**X**	–	**X**	**X**	**Z**	**Z**	**Z**	**Z**	**Z**
X	**X**	**X**	**X**	**X**	$\mathbf{Z}^+$	**N**	**Z**	$\mathbf{Z}^+$	**Z**
E	**X**	**X**	**X**	**X**	$\mathbf{Z}^-$	**Z**	**Z**	**Z**	$\mathbf{Z}^-$

$x+y$	$\mathbf{D}_m^\infty$	$\mathbf{D}_m^\cdot$	$\mathbf{D}_m^{-\infty}$	$\mathbf{D}_\infty^\infty$	$\mathbf{D}_\infty^\cdot$	$\mathbf{D}_\infty^{-\infty}$	$\mathbf{D}_n^\infty$	$\mathbf{D}_n^\cdot$	$\mathbf{D}_n^{-\infty}$
$\mathbf{D}_m^\infty$	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**
$\mathbf{D}_m^\cdot$	**X**	$\mathbf{D}_m^\cdot$	**X**	**X**	$\mathbf{D}_m^\cdot$	**X**	**X**	**X**	**X**
$\mathbf{D}_m^{-\infty}$	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**
$\mathbf{D}_\infty^\infty$	**X**	**X**	**X**	**X**	**X**	**X**	$x+y$	**T**	**F**
$\mathbf{D}_\infty^\cdot$	**X**	$\mathbf{D}_m^\cdot$	**X**	**X**	$\mathbf{D}_\infty^\cdot$	**X**	**T**	**T**	**T**
$\mathbf{D}_\infty^{-\infty}$	**X**	**X**	**X**	**X**	**X**	**X**	**F**	**T**	**F**

The table contains four subtables:

- one for the types [**B**]˙, [**D**]˙, [**X**]˙, and [**E**]˙ (16 entries),
- one for the types [**N**]˙, [**Z**]˙, [$\mathbf{Z}^+$]˙, and [$\mathbf{Z}^-$]˙ (16 entries),
- one for the types [**T**]˙ and [**F**]˙ (4 entries), and
- one for floating types (45 entries).

The subtable for floating types has a special interpretation. If you look where the row labelled [$\mathbf{D}_m^\cdot$]˙ meets the column labelled [$\mathbf{D}_\infty^\cdot$]˙ you find the set [$\mathbf{D}_m^\cdot$]˙. That ought to represent a rule something like

$$[\, x \in \mathbf{D}_m^\cdot \vdash y \in \mathbf{D}_\infty^\cdot \vdash x+y \in \mathbf{D}_m^\cdot \,]^\cdot.$$

The rule above, however, merely holds when $[\, \mathsf{m} \,]^{\cdot}$ is a positive integer, so the type rule actually reads

[**Mac rule TypeD$^{\cdot}_{\mathsf{m}}$+D$^{\cdot}_{\infty}$** : $\mathsf{m} \in \mathbf{Z}^{+} \vdash \mathsf{x} \in \mathbf{D}^{\cdot}_{\mathsf{m}} \vdash \mathsf{y} \in \mathbf{D}^{\cdot}_{\infty} \vdash \mathsf{x} + \mathsf{y} \in \mathbf{D}^{\cdot}_{\mathsf{m}}$]

This rule says that if $[\, \mathsf{m} \,]^{\cdot}$ is a positive integer, if $[\, \mathsf{x} \,]^{\cdot}$ is a decimal fraction of precision $[\, \mathsf{m} \,]^{\cdot}$, and if $[\, \mathsf{y} \,]^{\cdot}$ is a decimal fraction of precision $[\, \infty \,]^{\cdot}$, then $[\, \mathsf{x} + \mathsf{y} \,]^{\cdot}$ is a decimal fraction of precision $[\, \mathsf{m} \,]^{\cdot}$.

In general, whenever $[\, \mathbf{D}^{\infty}_{\mathsf{m}} \,]^{\cdot}$, $[\, \mathbf{D}^{\cdot}_{\mathsf{m}} \,]^{\cdot}$, or $[\, \mathbf{D}^{-\infty}_{\mathsf{m}} \,]^{\cdot}$ occur in a type rule, you must remember to include the premise $[\, \mathsf{m} \in \mathbf{Z}^{+} \,]^{\cdot}$.

Exercise 11.12.1 What does Type$\mathbf{D}^{\infty}_{\mathsf{m}} \wedge \mathbf{D}^{\cdot}_{\mathsf{m}}$ say?

If you look where the row labelled $[\, \mathbf{D}^{\cdot}_{\mathsf{m}} \,]^{\cdot}$ meets the column labelled $[\, \mathbf{D}^{\cdot}_{\mathsf{n}} \,]^{\cdot}$ you find the set $[\, \mathbf{X} \,]^{\cdot}$. The corresponding inference rule reads

[**Mac rule TypeD$^{\cdot}_{\mathsf{m}}$+D$^{\cdot}_{\mathsf{n}}$** : $\mathsf{m} \in \mathbf{Z}^{+} \vdash \mathsf{n} \in \mathbf{Z}^{+} \vdash \mathsf{m} \neq \mathsf{n} \vdash \mathsf{x} \in \mathbf{D}^{\cdot}_{\mathsf{m}} \vdash \mathsf{y} \in \mathbf{D}^{\cdot}_{\mathsf{n}} \vdash \mathsf{x} + \mathsf{y} \in \mathbf{X}$]

This rule says that if $[\, \mathsf{m} \,]^{\cdot}$ and $[\, \mathsf{n} \,]^{\cdot}$ are different positive integers, and if $[\, \mathsf{x} \,]^{\cdot}$ and $[\, \mathsf{y} \,]^{\cdot}$ are floating fractions of precision $[\, \mathsf{m} \,]^{\cdot}$ and $[\, \mathsf{n} \,]^{\cdot}$, respectively, then $[\, \mathsf{x} + \mathsf{y} \,]^{\cdot}$ is an exception.

Exercise 11.12.2 What does Type$\mathbf{D}^{\infty}_{\mathsf{m}} \wedge \mathbf{D}^{\cdot}_{\mathsf{n}}$ say?

Here you have an example of a lemma whose proof uses floating type rules:

[**Mac lemma L11.12.3** : $1.23\mathrm{F} + 2.34\mathrm{F} + 0.5 \in \mathbf{D}^{\cdot}_{3}$]

[**Mac proof of** L11.12.3**:**

L1 :	TypeNumeralInZp ▷	$3 \in \mathbf{Z}^{+}$	;
L2 :	TypeNumeralInDm ▷	$1.23\mathrm{F} \in \mathbf{D}^{\cdot}_{3}$	;
L3 :	TypeNumeralInDm ▷	$2.34\mathrm{F} \in \mathbf{D}^{\cdot}_{3}$	;
L4 :	Type$\mathbf{D}^{\cdot}_{\mathsf{m}}$+$\mathbf{D}^{\cdot}_{\mathsf{m}}$ ▷		
	L1 ▷ L2 ▷ L3 ▷	$1.23\mathrm{F} + 2.34\mathrm{F} \in \mathbf{D}^{\cdot}_{3}$	;
L5 :	TypeNumeralInDi ▷	$0.5 \in \mathbf{D}^{\cdot}_{\infty}$	;
L6 :	Type$\mathbf{D}^{\cdot}_{\mathsf{m}}$+$\mathbf{D}^{\cdot}_{\infty}$ ▷		
	L1 ▷ L4 ▷ L5 ▷	$1.23\mathrm{F} + 2.34\mathrm{F} + 0.5 \in \mathbf{D}^{\cdot}_{3}$	]

11.13 Lemmas with premisses

Here you have a lemma with a premise:

[**Mac lemma L11.13.1** : $\mathsf{x} \in \mathbf{N} \vdash \mathsf{x} + 1 \in \mathbf{N}$]

[L11.13.1] says that if [x] is a natural number, then [$x+1$] is a natural number. Here you have a proof of the lemma:

[**Mac proof of** L11.13.1:
L1 : Premise ⊳ $x \in \mathbf{N}$;
L2 : TypeNumeralInN ⊳ $1 \in \mathbf{N}$;
L3 : Type**N**+**N** ⊳ L1 ⊳ L2 ⊳ $x+1 \in \mathbf{N}$]

As you can see, Line 1 in the proof contains a new argumentation that reads *premise*. A lemma can have any number of premisses, and each premise can be stated in the proof using the "premise" argumentation.

11.14 Use of lemmas

Once a lemma is proved, it can be used as a new derivation rule. Here you have an example:

[**Mac lemma L11.14.1** : $x \in \mathbf{N} \vdash x+1+1 \in \mathbf{N}$]

[**Mac proof of** L11.14.1:
L1 : Premise ⊳ $x \in \mathbf{N}$;
L2 : L11.13.1 ⊳ L1 ⊳ $x+1 \in \mathbf{N}$;
L3 : L11.13.1 ⊳ L2 ⊳ $x+1+1 \in \mathbf{N}$]

Exercise 11.14.2 Prove [$x \in \mathbf{N} \vdash x{+}1{+}1{+}1{+}1 \in \mathbf{N}$] with a proof containing at most three proof lines.

11.15 Circular proofs

You must be careful to avoid *circular proofs*. Here you have an example of a circular proof:

[**Mac lemma L11.15.1** : $1 \equiv 2$]

[**Mac proof of** L11.15.1:
L1 : L11.15.2 ⊳ $2 \equiv 1$;
L2 : Commutativity ⊳ L1 ⊳ $1 \equiv 2$]

[**Mac lemma L11.15.2** : $2 \equiv 1$]

[**Mac proof of** L11.15.2:
L1 : L11.15.1 ⊳ $1 \equiv 2$;
L2 : Commutativity ⊳ L1 ⊳ $2 \equiv 1$]

When seen in isolation, each of the two proofs is correct. However, the proofs depend on each other, which is illegal.

---

premise
circular proof

11.16 Subtypes

Any natural number is also an integer:

[**Mac rule SubtypeNZ** : $x \in \mathbf{N} \vdash x \in \mathbf{Z}$]

[SubtypeNZ] is an example of a *subtype rule*. You can look up [SubtypeNZ] under "the set of natural numbers" in the reference manual (Section A.37), and you can find more subtype rules in Appendix A. Here you have some more subtype rules:

[**Mac rule SubtypeTB** : $x \in \mathbf{T} \vdash x \in \mathbf{B}$]

[**Mac rule SubtypeFB** : $x \in \mathbf{F} \vdash x \in \mathbf{B}$]

[**Mac rule SubtypeZD** : $x \in \mathbf{Z} \vdash x \in \mathbf{D}$]

Here you have an example of use:

[**Mac lemma L11.16.1** : $x \in \mathbf{F} \vdash y \in \mathbf{N} \vdash x + y \in \mathbf{X}$]

Appendix A contains many type rules, but not the one expressed by L11.16.1. However, you can find a rule, Type**B+D**, that says that if [x] is a truth value and [y] is a decimal fraction then [x + y] is an exception. Together with subtype rules, this gives the desired result:

[**Mac proof of** L11.16.1**:**

L1 :	Premise ▷	$x \in \mathbf{F}$	;
L2 :	Premise ▷	$y \in \mathbf{N}$	;
L3 :	SubtypeFB ▷ L1 ▷	$x \in \mathbf{B}$	;
L4 :	SubtypeNZ ▷ L2 ▷	$y \in \mathbf{Z}$	;
L5 :	SubtypeZD ▷ L4 ▷	$y \in \mathbf{D}$	;
L6 :	Type**B+D** ▷ L3 ▷ L5 ▷	$x + y \in \mathbf{X}$	]

11.17 Sets and representations

The set [**B**] of truth values contains those maps that are classical and which weakly represent a truth value:

[**Mac rule SubtypeBC** : $x \in \mathbf{B} \vdash \ell' x$]

[**Mac rule ElimB** : $x \in \mathbf{B} \vdash \overline{\mathbf{B}}' x$]

[**Mac rule IntroB** : $\ell' x \vdash \overline{\mathbf{B}}' x \vdash x \in \mathbf{B}$]

[$x \in \mathbf{B}$] is false if [x] is classical but does not represent a truth value:

---
subtype rule

[**Mac rule SubtypeNotBC** : $x \notin \mathbf{B} \vdash \ell' x$]

[**Mac rule ElimNotB** : $x \notin \mathbf{B} \vdash \neg\overline{\mathbf{B}}' x$]

[**Mac rule IntroNotB** : $\ell' x \vdash \neg\overline{\mathbf{B}}' x \vdash x \notin \mathbf{B}$]

You can find the rules above under "the set of truth values" in the reference manual (Section A.42).

The representation of truth is special in that truth merely has one representation, namely [T]˙. That gives rise to two extra rules:

[**Mac rule ElimTT** : $x \in \mathbf{T} \vdash x$]

[**Mac rule IntroTT** : $x \vdash x \in \mathbf{T}$]

You can find these two rules above under "the set of true maps" in the reference manual (Section A.41).

11.18 The class of sets

[**Set**]˙ denotes the *class of all sets*. [$x \in$ **Set**]˙ is true if [x]˙ is a set.

Classes behave very much like sets. Actually, the distinction between classes and sets is a rather technical one which you don't need to worry about while reading this book. All you need to know is that [**Set**]˙ is called the "class" of all sets instead of the "set" of all sets. [**B**]˙ is a set:

[**Mac rule SetB** : $\mathbf{B} \in \mathbf{Set}$]

You can find the rule above under "the set of truth values" in the reference manual (Section A.42), and you can find many similar rules in Appendix A.

The following rules resemble the subtype and introduction rules in Section 11.17 but they are more general because they say something about all sets:

[**Mac rule SubtypeSetC** : $y \in \mathbf{Set} \vdash x \in y \vdash \ell' x$]

[**Mac rule SubtypeNotSetC** : $y \in \mathbf{Set} \vdash x \notin y \vdash \ell' x$]

[**Mac rule TypeCInSet** : $y \in \mathbf{Set} \vdash \ell' x \vdash (x \in y) \in \mathbf{B}$]

You can find the rules above under "x belongs to y" in the reference manual (Section A.50)

[**Set**]˙ the class of sets
the class of all sets

11.19 Replacement

In Chapter 6 I presented the notion of *replacement* [Replace]˙ and *reverse replacement* [Reverse]˙. You can also use [Replace]˙ and [Reverse]˙ in derivation proofs:

[**Mac rule Replace** : **if** replace(x, y, u, v) **then** $\mathsf{x} \equiv \mathsf{y} \vdash \mathsf{u} \equiv \mathsf{v}$]

[**Mac rule Reverse** : **if** replace(x, y, u, v) **then** $\mathsf{x} \equiv \mathsf{y} \vdash \mathsf{v} \equiv \mathsf{u}$]

Here you have an example of use:

[**Mac lemma L11.19.1** : (1 : : 2 : : 3) tail head ≡ 2]

[**Mac proof of** L11.19.1**:**

L1 :	Replace ▷ TailPair ▷	(1: :2: :3) tail head ≡ (2: :3) head	;
L2 :	HeadPair ▷	(2: :3) head ≡ 2	;
L3 :	Transitivity ▷ L1 ▷ L2 ▷	(1: :2: :3) tail head ≡ 2	]

Exercise 11.19.2 Prove the following:

[**Mac lemma L11.19.3** : a head ≡ a ⊢ a head head ≡ a]

11.20 Definition

In Chapter 6 I presented the use of defintions in algebraic proofs. You may also use definitions in derivation proofs by the following rule: If [$\mathsf{u} \doteq \mathsf{v}$]˙ or [$\mathsf{u} \stackrel{\cdot}{=} \mathsf{v}$]˙ and if [$\mathsf{x} \equiv \mathsf{y}$]˙ is a replacement or reverse replacement of [$\mathsf{u} \equiv \mathsf{v}$]˙, then [$\mathsf{x} \equiv \mathsf{y}$]˙. The [Definition]˙ rule reads:

[**Mac rule Definition** : **if** definition(x, y) **then** $\mathsf{x} \equiv \mathsf{y}$]

Here you have an example of use:

[**Mac lemma L11.20.1** : nlist(1) head ≡ 1]

[**Mac proof of** L11.20.1**:**

L1 :	Definition ▷	nlist(1) head ≡ (1: :nlist(1+1)) head	;
L2 :	HeadPair ▷	(1: :nlist(1+1)) head ≡ 1	;
L3 :	Transitivity▷L1▷L2 ▷	nlist(1) head ≡ 1	]

---

replacement
reverse replacement

11.21 Replacement with two premisses

The following replacement rule with two premisses is handy in derivation proofs: If $[\, \mathsf{u} \equiv \mathsf{a}\,]^{\cdot}$ and $[\, \mathsf{v} \equiv \mathsf{b}\,]^{\cdot}$ are replacements of $[\, \mathsf{x} \equiv \mathsf{y}\,]^{\cdot}$ then $[\, \mathsf{x} \equiv \mathsf{y} \vdash \mathsf{u} \equiv \mathsf{v} \vdash \mathsf{a} \equiv \mathsf{b}\,]^{\cdot}$:

[**Mac rule Replace'** : **if** $\mathrm{replace}(\mathsf{x}, \mathsf{y}, \mathsf{u}, \mathsf{a}) \wedge \mathrm{replace}(\mathsf{x}, \mathsf{y}, \mathsf{v}, \mathsf{b})$ **then** $\mathsf{x} \equiv \mathsf{y} \vdash \mathsf{u} \equiv \mathsf{v} \vdash \mathsf{a} \equiv \mathsf{b}$]

The reverse version reads:

[**Mac rule Reverse'** : **if** $\mathrm{replace}(\mathsf{x}, \mathsf{y}, \mathsf{u}, \mathsf{a}) \wedge \mathrm{replace}(\mathsf{x}, \mathsf{y}, \mathsf{v}, \mathsf{b})$ **then** $\mathsf{x} \equiv \mathsf{y} \vdash \mathsf{a} \equiv \mathsf{b} \vdash \mathsf{u} \equiv \mathsf{v}$]

Here you have an example of use:

[**Mac lemma L11.21.1** : $\mathsf{x} \in \mathbf{N} \vdash f_3(\mathsf{x}) \in \mathbf{N}$]

[**Mac proof of** L11.21.1**:**

L1 :	Premise ▷	$\mathsf{x} \in \mathbf{N}$	;
L2 :	TypeNumeralInN ▷	$2 \in \mathbf{N}$	;
L3 :	TypeNumeralInN ▷	$4 \in \mathbf{N}$	;
L4 :	Type**N·N** ▷ L2 ▷ L1 ▷	$2 \cdot \mathsf{x} \in \mathbf{N}$	;
L5 :	Type**N+N** ▷ L4 ▷ L3 ▷	$2 \cdot \mathsf{x} + 4 \in \mathbf{N}$	;
L6 :	Definition ▷	$f_3(\mathsf{x}) \equiv 2 \cdot \mathsf{x} + 4$	;
L7 :	Reverse' ▷ L6 ▷ L5 ▷	$f_3(\mathsf{x}) \in \mathbf{N}$	]

In the proof above, Line 5 is shorthand for $[\, 2 \cdot \mathsf{x} + 4 \in \mathbf{N} \equiv \mathsf{T}\,]^{\cdot}$ and Line 7 is shorthand for $[\, f_3(\mathsf{x}) \in \mathbf{N} \equiv \mathsf{T}\,]^{\cdot}$. The latter arises from the former by replacing $[\, 2 \cdot \mathsf{x} + 4\,]^{\cdot}$ by $[\, f_3(\mathsf{x})\,]^{\cdot}$.

11.22 Repetition

In algebraic proofs, Reflexivity allows to repeat a term with a change in notation. Likewise, the following rule allows to repeat proof lines in derivation proofs:

[**Mac rule Repetition** : $\mathsf{x} \equiv \mathsf{y} \vdash \mathsf{x} \equiv \mathsf{y}$]

Here you have an example:

[**Mac lemma L11.22.1** : $\mathsf{x} \in \mathbf{N} \vdash f_4(\mathsf{x}) \in \mathbf{N}$]

[**Mac proof of** L11.22.1**:**

L1 :	Premise ▷	$\mathsf{x} \in \mathbf{N}$	;
L2 :	TypeNumeralInN ▷	$2 \in \mathbf{N}$	;
L3 :	TypeNumeralInN ▷	$4 \in \mathbf{N}$	;
L4 :	Type**N·N** ▷ L2 ▷ L1 ▷	$2 \cdot \mathsf{x} \in \mathbf{N}$	;
L5 :	Type**N+N** ▷ L4 ▷ L3 ▷	$2 \cdot \mathsf{x} + 4 \in \mathbf{N}$	;
L6 :	Repetition ▷ L5 ▷	$f_4(\mathsf{x}) \in \mathbf{N}$	]

Exercise 11.22.2 Prove the following:

[**Mac lemma L11.22.3** : $\mathsf{x} \equiv \langle \mathsf{y}, \mathsf{z} \rangle \vdash \mathsf{x}\ \mathrm{head} \equiv \mathsf{y}$]

11.23 Selection

Here you have some rules about selection:

[**Mac rule IfT** : $\mathsf{x} \in \mathbf{T} \vdash \mathrm{if}(\mathsf{x},\mathsf{y},\mathsf{z}) \equiv \mathsf{y}$]

[**Mac rule IfF** : $\mathsf{x} \in \mathbf{F} \vdash \mathrm{if}(\mathsf{x},\mathsf{y},\mathsf{z}) \equiv \mathsf{z}$]

And here you have an example of use:

[**Mac lemma L11.23.1** : $(\mathsf{n} = 0) \in \mathbf{F} \vdash \mathsf{n}! \equiv \mathsf{n} \cdot (\mathsf{n} - 1)!$]

[**Mac proof of** L11.23.1**:**

L1 :	Premise ▷	$(\mathsf{n} = 0) \in \mathbf{F}$	;
L2 :	Definition ▷	$\mathsf{n}! \equiv \mathsf{n} = 0 \begin{cases} 1 \\ \mathsf{n} \cdot (\mathsf{n}-1)! \end{cases}$	;
L3 :	IfF ▷ L1 ▷	$\mathsf{n} = 0 \begin{cases} 1 \\ \mathsf{n} \cdot (\mathsf{n}-1)! \end{cases} \equiv \mathsf{n} \cdot (\mathsf{n}-1)!$	;
L4 :	Transitivity ▷ L2 ▷ L3 ▷	$\mathsf{n}! \equiv \mathsf{n} \cdot (\mathsf{n}-1)!$	]

Exercise 11.23.2 I define $\left[\boxed{f_9(\mathsf{x})} \doteq \mathrm{if}(\mathsf{x}, \mathsf{T}, f(\neg \mathsf{x})) \right]$. Prove the following:

[**Mac lemma L11.23.3** : $\mathsf{x} \in \mathbf{F} \vdash f_9(\mathsf{x}) \equiv \mathsf{T}$]

11.24 Computation

A term is *closed* if it has no free variables. If [x]˙ and [y]˙ are closed and compute to the same value, then [x ≡ y]˙:

[**Mac rule Computation** : **if** closed(x) ∧ closed(y) ∧ equal(x, y) **then** x ≡ y]

Here you have an example of use:

[**Mac lemma L11.24.1** : $(2 + 2 = 5) \in \mathbf{F}$]

[**Mac proof of** L11.24.1**:**

L1 :	Computation ▷	$2 + 2 = 5 \equiv \mathsf{F}$	;
L2 :	TypeFInF ▷	$\mathsf{F} \in \mathbf{F}$	;
L3 :	Reverse' ▷ L1 ▷ L2 ▷	$(2 + 2 = 5) \in \mathbf{F}$	]

Like in algebraic proofs I allow computation axioms to occur in the premisses of argumentations:

[**Mac proof of** L11.24.1**:**

L1 :	TypeFInF ▷	$\mathsf{F} \in \mathbf{F}$	;
L2 :	Reverse' ▷ $2 + 2 = 5 \equiv \mathsf{F}$ ▷ L1 ▷	$(2 + 2 = 5) \in \mathbf{F}$	]

Exercise 11.24.2 Prove the following:

[**Mac lemma L11.24.3** : $5! \in \mathbf{N}$]

[$f_9(\mathsf{x})$]˙	f nine of x end closed

11.25 Derivation systems

Derivation systems are similar to algebraic systems: A derivation system consists of a finite collection of constructs, a finite collection of derivation rules and a single contradiction. Here you have a derivation system:

Constructs	Axioms	Contradiction
$[\langle\rangle]^{\cdot}$	HeadEmpty	$[\langle\rangle \equiv \bot]^{\cdot}$
$[\bot]^{\cdot}$	HeadBottom	
$[\mathsf{x} :: \mathsf{y}]^{\cdot}$	HeadPair	
$[\mathsf{x}\ \text{head}]^{\cdot}$	TailEmpty	
$[\mathsf{x}\ \text{tail}]^{\cdot}$	TailBottom	
	TailPair	
	Commutativity	
	Transitivity	

You may regard this book as one, big derivation system, namely the $[\mathsf{Mac}]^{\cdot}$ system. When you solve exercises in this book, you can use all the rules and lemmas in the $[\mathsf{Mac}]^{\cdot}$ system. Later on, if you don't like my derivation system, you can choose another one or even make your own.

I present lots of constructs and rules in this book, so I just need to present a contradiction to complete the $[\mathsf{Mac}]^{\cdot}$ derivation system. I choose the following contradicion:

[**Mac antirule Contradiction** : $\bot \equiv \mathsf{T}$]

In other words, I claim that nobody can prove $[\bot \equiv \mathsf{T}]^{\cdot}$ in the $[\mathsf{Mac}]^{\cdot}$ derivation system. If you succeed to prove $[\bot \equiv \mathsf{T}]^{\cdot}$ in the $[\mathsf{Mac}]^{\cdot}$ derivation system, then you have proved that my derivation system is *inconsistent*, and then I have some work ahead of me.

11.26 Answers

Answer 11.3.1 (a), (c), and (d).

Answer 11.6.1

[**Mac proof of** L11.6.2**:**

L1 :	HeadPair $\rhd$	$(\mathsf{x} :: \mathsf{y})$ head $\equiv \mathsf{x}$	;
L2 :	HeadPair $\rhd$	$(\mathsf{x} :: \mathsf{z})$ head $\equiv \mathsf{x}$	;
L3 :	Commutativity $\rhd$ L2 $\rhd$	$\mathsf{x} \equiv (\mathsf{x} :: \mathsf{z})$ head	;
L4 :	Transitivity $\rhd$ L1 $\rhd$ L3 $\rhd$	$(\mathsf{x} :: \mathsf{y})$ head $\equiv (\mathsf{x} :: \mathsf{z})$ head	]

Answer 11.7.1 (a), (b) and (d).

Answer 11.8.1 (a) and (b) are correct instantiations. (c) is not.

derivation system
inconsistent

In (b), the rule $[\, \langle a \mid x{:=}N \rangle \equiv \langle b \mid x{:=}N \rangle \vdash \text{case}(x, a, c) \equiv \text{case}(x, b, d) \,]^{\cdot}$ is instantiated thus: First, $[\, x \,]^{\cdot}$ is replaced by $[\, u \,]^{\cdot}$, $[\, a \,]^{\cdot}$ is replaced by $[\, 1 + u \,]^{\cdot}$, $[\, b \,]^{\cdot}$ is replaced by $[\, 2 + u \,]^{\cdot}$, and $[\, c \,]^{\cdot}$ and $[\, d \,]^{\cdot}$ are replaced by $[\, \bot \,]^{\cdot}$. This gives rise to the rule $[\, \langle 1 + u \mid u{:=}N \rangle \equiv \langle 2 + u \mid u{:=}N \rangle \vdash \text{case}(u, 1 + u, \bot) \equiv \text{case}(u, 2 + u, \bot) \,]^{\cdot}$. Second, the substitutions are performed. This gives rise to the rule $[\, 1 + N \equiv 2 + N \vdash \text{case}(u, 1 + u, \bot) \equiv \text{case}(u, 2 + u, \bot) \,]^{\cdot}$.

(c) is not a correct instantiations because there is no way to turn $[\, x \equiv y \vdash y \equiv x \,]^{\cdot}$ into $[\, 2 + 2 \equiv \langle x \mid x{:=}4 \rangle \vdash 4 \equiv 2 + 2 \,]^{\cdot}$ using the three legal means for instantiation.

Answer 11.8.2

[**Mac proof of** L11.8.3**:**
- L1 : ApplyLambda $\rhd$ $\quad (\lambda x.x)\, '\, ((\lambda y.y)\, '\, N) \equiv (\lambda y.y)\, '\, N$;
- L2 : ApplyLambda $\rhd$ $\quad (\lambda y.y)\, '\, N \equiv N$;
- L3 : Transitivity $\rhd$ L1 $\rhd$ L2 $\rhd$ $\quad (\lambda x.x)\, '\, ((\lambda y.y)\, '\, N) \equiv N$]

Answer 11.9.1 Yes.

Answer 11.9.3 The lemma says $[\, (\lambda x.x)\, '\, ((\lambda y.y)\, '\, T) \equiv T \,]^{\cdot}$. Here you have one proof:

[**Mac proof of** L11.9.4**:**
- L1 : ApplyLambda $\rhd$ $\quad (\lambda x.x)\, '\, ((\lambda y.y)\, '\, T) \equiv (\lambda y.y)\, '\, T$;
- L2 : ApplyLambda $\rhd$ $\quad (\lambda y.y)\, '\, T \equiv T$;
- L3 : Transitivity $\rhd$ L1 $\rhd$ L2 $\rhd$ $\quad (\lambda x.x)\, '\, ((\lambda y.y)\, '\, T) \equiv T$]

Here you have the same proof in which $[\ \equiv T \,]^{\circ}$ is understood in Line $[\, L2 \,]^{\cdot}$ and $[\, L3 \,]^{\cdot}$.

[**Mac proof of** L11.9.4**:**
- L1 : ApplyLambda $\rhd$ $\quad (\lambda x.x)\, '\, ((\lambda y.y)\, '\, T) \equiv (\lambda y.y)\, '\, T$;
- L2 : ApplyLambda $\rhd$ $\quad (\lambda y.y)\, '\, T$;
- L3 : Transitivity $\rhd$ L1 $\rhd$ L2 $\rhd$ $\quad (\lambda x.x)\, '\, ((\lambda y.y)\, '\, T)$]

Answer 11.12.1 Type$\mathbf{D}^{\infty}_{m}\wedge\mathbf{D}^{\cdot}_{m}$ says that the minimum of a positive infinity and a finite quantity is a finite quantity:

[**Mac rule Type**$\mathbf{D}^{\infty}_{m}\wedge\mathbf{D}^{\cdot}_{m}$: $m \in \mathbf{Z}^{+} \vdash x \in \mathbf{D}^{\infty}_{m} \vdash y \in \mathbf{D}^{\cdot}_{m} \vdash x \wedge y \in \mathbf{D}^{\cdot}_{m}$]

Answer 11.12.2 Type$\mathbf{D}^{\infty}_{m}\wedge\mathbf{D}^{\cdot}_{n}$ says that the minimum of a positive infinity and a finite quantity of different precisions is an exception:

[**Mac rule Type**$\mathbf{D}^{\infty}_{m}\wedge\mathbf{D}^{\cdot}_{m}$: $m \in \mathbf{Z}^{+} \vdash n \in \mathbf{Z}^{+} \vdash m \neq n \vdash x \in \mathbf{D}^{\infty}_{m} \vdash y \in \mathbf{D}^{\cdot}_{n} \vdash x \wedge y \in \mathbf{X}$]

Answer 11.14.2

[**Mac lemma L11.26.1** : $x \in \mathbf{N} \vdash x + 1 + 1 + 1 + 1 \in \mathbf{N}$]

[**Mac proof of** L11.26.1**:**

L1 :	Premise ▷	$x \in \mathbf{N}$	;
L2 :	L11.14.1 ▷ L1 ▷	$x + 1 + 1 \in \mathbf{N}$	;
L3 :	L11.14.1 ▷ L2 ▷	$x + 1 + 1 + 1 + 1 \in \mathbf{N}$	]

Answer 11.19.2

[**Mac proof of** L11.19.3**:**

L1 :	Premise ▷	a head $\equiv a$	;
L2 :	Replace ▷ L1 ▷	a head head $\equiv a$ head	;
L3 :	Transitivity ▷ L2 ▷ L1 ▷	a head head $\equiv a$	]

Answer 11.22.2

[**Mac proof of** L11.22.3**:**

L1 :	Premise ▷	$x \equiv \langle y, z \rangle$	;
L2 :	Repetition ▷ L1 ▷	$x \equiv y :: \langle z \rangle$	;
L3 :	HeadPair ▷	$(y :: \langle z \rangle)$ head $\equiv y$	;
L4 :	Reverse' ▷ L2 ▷ L3 ▷	x head $\equiv y$	]

Answer 11.23.2

[**Mac proof of** L11.23.3**:**

L1 :	Premise ▷	$x \in \mathbf{F}$	;
L2 :	Definition ▷	$f_9(x) \equiv \mathrm{if}(x, T, f_9(\neg x))$	;
L3 :	IfF ▷ L1 ▷	$\mathrm{if}(x, T, f_9(\neg x)) \equiv f_9(\neg x)$	;
L4 :	Transitivity ▷ L2 ▷ L3 ▷	$f_9(x) \equiv f_9(\neg x)$	;
L5 :	Type¬F ▷ L1 ▷	$\neg x \in \mathbf{T}$	;
L6 :	Definition ▷	$f_9(\neg x) \equiv \mathrm{if}(\neg x, T, f_9(\neg\neg x))$	;
L7 :	IfT ▷ L5 ▷	$\mathrm{if}(\neg x, T, f_9(\neg\neg x)) \equiv T$	;
L8 :	Transitivity ▷ L6 ▷ L7 ▷	$f_9(\neg x) \equiv T$	;
L9 :	Transitivity ▷ L4 ▷ L8 ▷	$f_9(x) \equiv T$	]

Answer 11.24.2

[**Mac proof of** L11.24.3**:**

L1 :	TypeNumeralInN ▷	$120 \in \mathbf{N}$	;
L2 :	Reverse' ▷ $5! \equiv 120$ ▷ L1 ▷	$5! \in \mathbf{N}$	]

Chapter 12

Deduction

12.1 Introduction

A statement like

$$[\, x \in \mathbf{Z} \to x + 2 \in \mathbf{Z} \,]^{\cdot}$$

says

if $[\, x \in \mathbf{Z} \,]^{\cdot}$ then $[\, x + 2 \in \mathbf{Z} \,]^{\cdot}$.

$[\, x \to y \,]^{\cdot}$ is an *implication*, and implication is what this chapter is all about.

12.2 Implication

I formally define $[\, x \to y \,]^{\cdot}$ thus:

$\left[\, \textbf{construct}\ \boxed{x \to y} \,\right]$,

$[\, x \to y \to z \stackrel{\smile}{\to} x \to (y \to z) \,]$, and

$\left[\, x \to y \equiv z \stackrel{\circ}{\to} \mathrm{case}(x, y, \mathsf{T}) \equiv \mathrm{case}(x, z, \mathsf{T}) \,\right]$.

Here you have some examples of use:

$[\, x{\in}\mathbf{Z} \to \mathrm{if}(x, y, z){\equiv}\bullet \,]^{\cdot}$	says	if $[\, x{\in}\mathbf{Z} \,]^{\cdot}$ then $[\, \mathrm{if}(x, y, z){\equiv}\bullet \,]^{\cdot}$.
$[\, x{\in}\mathbf{Z} \to y{\in}\mathbf{Z} \to x{+}y{\equiv}y{+}x \,]^{\cdot}$	says	if $[\, x{\in}\mathbf{Z} \,]^{\cdot}$ and $[\, y{\in}\mathbf{Z} \,]^{\cdot}$ then $[\, x{+}y{\equiv}y{+}x \,]^{\cdot}$.
$[\, x{\in}\mathbf{Z} \to y{\in}\mathbf{Z} \to x{+}y{=}y{+}x \,]^{\cdot}$	says	if $[\, x{\in}\mathbf{Z} \,]^{\cdot}$ and $[\, y{\in}\mathbf{Z} \,]^{\cdot}$ then $[\, x{+}y{=}y{+}x{\equiv}\mathsf{T} \,]^{\cdot}$.
$[\, x{\in}\mathbf{Z} \to x{+}2{\in}\mathbf{Z} \,]^{\cdot}$	says	if $[\, x{\in}\mathbf{Z} \,]^{\cdot}$ then $[\, x{+}2{\in}\mathbf{Z}{\equiv}\mathsf{T} \,]^{\cdot}$.

---

implication

$[\, x \to y \,]^{\cdot}$ x deduces y

In general, if $[\,\mathsf{x}\,]^{\cdot}$ is a term and $[\,\mathsf{y}\,]^{\cdot}$ is a formula, then

$$[\,\mathsf{x} \rightarrow \mathsf{y}\,]^{\cdot}$$

says that if $[\,\mathsf{x} \equiv \mathsf{T}\,]^{\cdot}$ then $[\,\mathsf{y}\,]^{\cdot}$ holds. If $[\,\mathsf{x}\,]^{\cdot}$ and $[\,\mathsf{y}\,]^{\cdot}$ are terms, then

$$[\,\mathsf{x} \rightarrow \mathsf{y}\,]^{\cdot}$$

has the property that the term $[\,\mathsf{y}\,]^{\cdot}$ occurs in a place where a formula is expected, so $[\,\mathsf{y}\,]^{\cdot}$ is shorthand for $[\,\mathsf{y} \equiv \mathsf{T}\,]^{\cdot}$. Hence, $[\,\mathsf{x} \rightarrow \mathsf{y}\,]^{\cdot}$ means: if $[\,\mathsf{x} \equiv \mathsf{T}\,]^{\cdot}$ then $[\,\mathsf{y} \equiv \mathsf{T}\,]^{\cdot}$.

The left hand side of $[\,\mathsf{x} \rightarrow \mathsf{y}\,]^{\cdot}$ must be a term. $[\,\mathsf{x} \rightarrow \mathsf{y}\,]^{\cdot}$ does not make sense when $[\,\mathsf{x}\,]^{\cdot}$ is a formula.

12.3 Implication versus inference

The inference

$$[\,\mathsf{x} \equiv \mathsf{y} \vdash \mathsf{y} \equiv \mathsf{z} \vdash \mathsf{x} \equiv \mathsf{z}\,]^{\cdot}$$

says that $[\,\mathsf{x} \equiv \mathsf{y}\,]^{\cdot}$ and $[\,\mathsf{y} \equiv \mathsf{z}\,]^{\cdot}$ *infer* $[\,\mathsf{x} \equiv \mathsf{z}\,]^{\cdot}$. I refer to $[\,\mathsf{x} \equiv \mathsf{y}\,]^{\cdot}$ and $[\,\mathsf{y} \equiv \mathsf{z}\,]^{\cdot}$ as *premises* and to $[\,\mathsf{x} \equiv \mathsf{z}\,]^{\cdot}$ as the *conclusion* of the inference. In contrast, the implication

$$[\,\mathsf{x} \in \mathbf{N} \rightarrow \mathsf{y} \in \mathbf{N} \rightarrow \mathsf{x} + \mathsf{y} \equiv \mathsf{y} + \mathsf{x}\,]^{\cdot}$$

says that $[\,\mathsf{x} \in \mathbf{N}\,]^{\cdot}$ and $[\,\mathsf{y} \in \mathbf{N}\,]^{\cdot}$ *imply* $[\,\mathsf{x} + \mathsf{y} \equiv \mathsf{y} + \mathsf{x}\,]^{\cdot}$. I refer to $[\,\mathsf{x} \in \mathbf{N}\,]^{\cdot}$ and $[\,\mathsf{y} \in \mathbf{N}\,]^{\cdot}$ as *hypotheses* and to $[\,\mathsf{x} + \mathsf{y} \equiv \mathsf{y} + \mathsf{x}\,]^{\cdot}$ as the *consequence* of the implication.

The inference

$$[\,\mathsf{x} \equiv \mathsf{y} \vdash \mathsf{y} \equiv \mathsf{z} \vdash \mathsf{x} \equiv \mathsf{z}\,]^{\cdot}$$

says

If $[\,\mathsf{x} \equiv \mathsf{y}\,]^{\cdot}$ and $[\,\mathsf{y} \equiv \mathsf{z}\,]^{\cdot}$ then $[\,\mathsf{x} \equiv \mathsf{z}\,]^{\cdot}$

and the implication

$$[\,\mathsf{x} \in \mathbf{N} \rightarrow \mathsf{y} \in \mathbf{N} \rightarrow \mathsf{x} + \mathsf{y} \equiv \mathsf{y} + \mathsf{x}\,]^{\cdot}$$

says

If $[\,\mathsf{x} \in \mathbf{N} \equiv \mathsf{T}\,]^{\cdot}$ and $[\,\mathsf{y} \in \mathbf{N} \equiv \mathsf{T}\,]^{\cdot}$ then $[\,\mathsf{x} + \mathsf{y} \equiv \mathsf{y} + \mathsf{x}\,]^{\cdot}$

---

infer
premise
conclusion
imply
hypothesis
consequence

so inferences and implications are very similar. One difference is that the premises of an inference are equations whereas the hypotheses of an implication are terms.

Another difference is that, contrary to inferences, implications are shorthand for equations. As an example,

$$[\, \mathsf{x} \in \mathbf{N} \rightarrow \mathsf{y} \in \mathbf{N} \rightarrow \mathsf{x} + \mathsf{y} \equiv \mathsf{y} + \mathsf{x} \,]^{\cdot}$$

is shorthand for the equation

$$\left[\mathsf{x} \in \mathbf{N} \left\langle \begin{matrix} \mathsf{y} \in \mathbf{N} \left\langle \begin{matrix} \mathsf{x}+\mathsf{y} \\ \mathsf{T} \end{matrix} \right. \\ \mathsf{T} \end{matrix} \right. \equiv \mathsf{x} \in \mathbf{N} \left\langle \begin{matrix} \mathsf{y} \in \mathbf{N} \left\langle \begin{matrix} \mathsf{y}+\mathsf{x} \\ \mathsf{T} \end{matrix} \right. \\ \mathsf{T} \end{matrix} \right. \right]^{\cdot}.$$

A statement of form

$$[\, (\mathsf{x} \equiv \mathsf{y} \vdash \mathsf{z} \equiv \mathsf{u}) \vdash \mathsf{v} \equiv \mathsf{w} \,]^{\cdot}$$

does not make sense since the premise of an inference cannot be an inference. Premises must be equations. Likewise, an implication of form

$$[\, (\mathsf{x} \equiv \mathsf{y} \rightarrow \mathsf{z} \equiv \mathsf{u}) \rightarrow \mathsf{v} \equiv \mathsf{w} \,]^{\cdot}$$

does not make sense since the hypothesis of an implication cannot be an implication. Hypotheses must be terms. However, a statement of form

$$[\, (\mathsf{x} \rightarrow \mathsf{y} \equiv \mathsf{z}) \vdash \mathsf{u} \equiv \mathsf{v} \,]^{\cdot}$$

makes perfect sense since $[\, \mathsf{x} \rightarrow \mathsf{y} \equiv \mathsf{z} \,]^{\cdot}$ is shorthand for an equation and, hence, $[\, \mathsf{x} \rightarrow \mathsf{y} \equiv \mathsf{z} \,]^{\cdot}$ makes sense as a premise. Here you have an application of that:

$$[\, \mathsf{x} \in \mathbf{T} \rightarrow \mathsf{u} \equiv \mathsf{v} \vdash \mathsf{x} \in \mathbf{F} \rightarrow \mathsf{u} \equiv \mathsf{v} \vdash \mathsf{x} \in \mathbf{B} \rightarrow \mathsf{u} \equiv \mathsf{v} \,]^{\cdot}.$$

The inference says

if

  if $[\, \mathsf{x} \in \mathbf{T} \,]^{\cdot}$ then $[\, \mathsf{u} \equiv \mathsf{v} \,]^{\cdot}$

and

  if $[\, \mathsf{x} \in \mathbf{F} \,]^{\cdot}$ then $[\, \mathsf{u} \equiv \mathsf{v} \,]^{\cdot}$

then

  if $[\, \mathsf{x} \in \mathbf{B} \,]^{\cdot}$ then $[\, \mathsf{u} \equiv \mathsf{v} \,]^{\cdot}$.

Or, in other words,

if

  $[\, \mathsf{u} \equiv \mathsf{v} \,]^{\cdot}$ for all $[\, \mathsf{x} \,]^{\cdot}$ that represent truth

and

  $[\, \mathsf{u} \equiv \mathsf{v} \,]^{\cdot}$ for all $[\, \mathsf{x} \,]^{\cdot}$ that represent falsehood

then

  $[\, \mathsf{u} \equiv \mathsf{v} \,]^{\cdot}$ for all $[\, \mathsf{x} \,]^{\cdot}$ that represent a truth value.

12.4 Deduction

I now introduce the proof technique of *deduction*. Here you have an example of deduction:

[**Mac lemma L12.4.1** : $\mathsf{x} \in \mathbf{Z} \to \mathsf{x} + 2 \in \mathbf{Z}$]

[**Mac proof of** L12.4.1**:**

L1 :	Block ▷	Begin	;
L2 :	Hypothesis ▷	$\mathsf{x} \in \mathbf{Z}$	;
L3 :	TypeNumeralInZ ▷	$2 \in \mathbf{Z}$	;
L4 :	Type**Z**+**Z** ▷ L2 ▷ L3 ▷	$\mathsf{x} + 2 \in \mathbf{Z}$	;
L5 :	Block ▷	End	;
L6 :	Repetition ▷ L4 ▷	$\mathsf{x} \in \mathbf{Z} \to \mathsf{x} + 2 \in \mathbf{Z}$	]

The proof above contains a *block*. The block starts at Line [L1] and ends at Line [L5]. Inside the block, Line [L2] states a hypothesis, namely [$\mathsf{x} \in \mathbf{Z}$]. In other words, [$\mathsf{x} \in \mathbf{Z}$] is assumed to be true from Line [L1] to Line [L5]. The assumption has no effect outside the block.

Line [L4] uses the hypothesis [$\mathsf{x} \in \mathbf{Z}$] from Line [L2] to conclude [$\mathsf{x}+2 \in \mathbf{Z}$]. The conclusion [$\mathsf{x} + 2 \in \mathbf{Z}$] holds inside the block where [$\mathsf{x} \in \mathbf{Z}$], but does not hold outside the block.

Outside the block, Line [L6] uses Line [L4] to conclude [$\mathsf{x} \in \mathbf{Z} \to \mathsf{x} + 2 \in \mathbf{Z}$]. In words, Line [L6] says "if [$\mathsf{x} \in \mathbf{Z}$] then [$\mathsf{x} + 2 \in \mathbf{Z}$]".

When seen from inside the block, Line [L2] to Line [L4] read [$\mathsf{x} \in \mathbf{Z}$], [$2 \in \mathbf{Z}$], and [$\mathsf{x} + 2 \in \mathbf{Z}$], respectively. When seen from outside the block, Line [L2] to Line [L4] read [$\mathsf{x} \in \mathbf{Z} \to \mathsf{x} \in \mathbf{Z}$], [$\mathsf{x} \in \mathbf{Z} \to 2 \in \mathbf{Z}$], and [$\mathsf{x} \in \mathbf{Z} \to \mathsf{x}+2 \in \mathbf{Z}$], respectively. In other words, the lines look different from different points of view.

When Line [L4] refers to Line [L2] and Line [L3], those lines read [$\mathsf{x} \in \mathbf{Z}$] and [$2 \in \mathbf{Z}$], respectively, which allows to conclude [$\mathsf{x} + 2 \in \mathbf{Z}$] by Type**Z**+**Z**.

When Line [L6] refers to Line [L4], Line [L4] reads [$\mathsf{x} \in \mathbf{Z} \to \mathsf{x}+2 \in \mathbf{Z}$] which allows to conclude [$\mathsf{x} \in \mathbf{Z} \to \mathsf{x} + 2 \in \mathbf{Z}$] by repetition.

Exercise 12.4.2 Prove the following:

[**Mac lemma L12.4.3** : $\mathsf{x} \in \mathbf{F} \to \mathsf{if}(\mathsf{x}, \mathsf{y}, \mathsf{z}) \equiv \mathsf{z}$]

12.5 Deduction in two levels

Here you have a proof with nested blocks:

[**Mac lemma L12.5.1** : $\mathsf{x} \in \mathbf{Z} \to \mathsf{y} \in \mathbf{Z} \to \mathsf{x} + \mathsf{y} \cdot \mathsf{x} \in \mathbf{Z}$]

---

deduction
block

[**Mac proof of** L12.5.1:

L01 :	Block $\rhd$	Begin	;
L02 :	Hypothesis $\rhd$	$\quad x \in \mathbf{Z}$	;
L03 :	Block $\rhd$	$\quad$ Begin	;
L04 :	Hypothesis $\rhd$	$\qquad y \in \mathbf{Z}$	;
L05 :	Type$\mathbf{Z}{\cdot}\mathbf{Z}$ $\rhd$ L4 $\rhd$ L2 $\rhd$	$\qquad y \cdot x \in \mathbf{Z}$	;
L06 :	Type$\mathbf{Z}{+}\mathbf{Z}$ $\rhd$ L2 $\rhd$ L5 $\rhd$	$\qquad x + y \cdot x \in \mathbf{Z}$	;
L07 :	Block $\rhd$	$\quad$ End	;
L08 :	Repetition $\rhd$ L6 $\rhd$	$\quad y \in \mathbf{Z} \rightarrow x + y \cdot x \in \mathbf{Z}$	;
L09 :	Block $\rhd$	End	;
L10 :	Repetition $\rhd$ L8 $\rhd$	$x \in \mathbf{Z} \rightarrow y \in \mathbf{Z} \rightarrow x + y \cdot x \in \mathbf{Z}$	]

Here you have some important observations:

[1]˙ and [01]˙ have the same syntax tree, so [L1]˙ and [L01]˙ denote the same variable.

The proof has two blocks. One goes from Line [L1]˙ to Line [L9]˙. The other goes from Line [L3]˙ to Line [L7]˙. The block from Line [L3]˙ to Line [L7]˙ is *inside* the block from Line [L1]˙ to Line [L9]˙. I will call the block from Line [L1]˙ to Line [L9]˙ the *outer block* and the one from Line [L3]˙ to Line [L7]˙ the *inner block*.

Line [L8]˙ refers to Line [L6]˙, so Line [L8]˙ refers into the inner block. Seen from Line [L8]˙, Line [L6]˙ reads [$y \in \mathbf{Z} \rightarrow x + y \cdot x \in \mathbf{Z}$]˙.

Line [L5]˙ refers to Line [L2]˙, so Line [L5]˙ refers out of the inner block. Seen from Line [L5]˙, Line [L2]˙ reads [$x \in \mathbf{Z}$]˙.

Line [L5]˙ also refers to Line [L4]˙. Seen from Line [L5]˙, Line [L4]˙ reads [$y \in \mathbf{Z}$]˙.

In general, the hypothesis of a block is added when refering into a block, but nothing is added when refering out of a block.

Exercise 12.5.2 Prove the following:

[**Mac lemma L12.5.3** : $x \in \mathbf{T} \rightarrow y \in \mathbf{N} \rightarrow \mathrm{if}(x, y, z) \in \mathbf{N}$]

12.6 References into two blocks

Here you have a new proof for L12.5.1:

[**Mac proof of** L12.5.1:

---

inside
outer block
inner block

L1 :	Block ▷	Begin	;
L2 :	Hypothesis ▷	$x \in \mathbf{Z}$	;
L3 :	Block ▷	Begin	;
L4 :	Hypothesis ▷	$y \in \mathbf{Z}$	;
L5 :	Type$\mathbf{Z}\cdot\mathbf{Z}$ ▷ L4 ▷ L2 ▷	$y \cdot x \in \mathbf{Z}$	;
L6 :	Type$\mathbf{Z}+\mathbf{Z}$ ▷ L2 ▷ L5 ▷	$x + y \cdot x \in \mathbf{Z}$	;
L7 :	Block ▷	End	;
L8 :	Block ▷	End	;
L9 :	Repetition ▷ L6 ▷	$x \in \mathbf{Z} \to y \in \mathbf{Z} \to x + y \cdot x \in \mathbf{Z}$	]

Line [L9]˙ refers to Line [L6]˙ which is inside two blocks. Seen from Line [L9]˙, Line [L6]˙ reads [$x \in \mathbf{Z} \to y \in \mathbf{Z} \to x + y \cdot x \in \mathbf{Z}$]˙. In general, when refering into several blocks in one go, the hypotheses of all the blocks are prepended. The example above illustrates in what order the hypotheses are prepended.

Exercise 12.6.1 Prove the following:

[**Mac lemma L12.6.2** : $x \in \mathbf{N} \to y \in \mathbf{N} \to z \in \mathbf{N} \to x \in \mathbf{Z}$]

12.7 References in and out of blocks

Here you have a rather constructed example that illustrates references in and out of blocks:

[**Mac lemma L12.7.1** : $x \to y \to u \to v \to u$]

[**Mac proof of** L12.7.1**:**

L01 :	Block ▷	Begin	;
L02 :	Hypothesis ▷	u	;
L03 :	Block ▷	Begin	;
L04 :	Hypothesis ▷	v	;
L05 :	Repetition ▷ L2 ▷	u	;
L06 :	Block ▷	End	;
L07 :	Block ▷	End	;
L08 :	Block ▷	Begin	;
L09 :	Hypothesis ▷	x	;
L10 :	Block ▷	Begin	;
L11 :	Hypothesis ▷	y	;
L12 :	Repetition ▷ L5 ▷	$u \to v \to u$	;
L13 :	Block ▷	End	;
L14 :	Block ▷	End	;
L15 :	Repetition ▷ L12 ▷	$x \to y \to u \to v \to u$	]

Line [L12]˙ refers out of two blocks and into two other ones to reach Line [L5]˙. The hypotheses of the two blocks Line [L12]˙ refers into are prepended. Line [L15]˙ then refers into two blocks to reach Line [L12]˙, so the hypotheses of those two blocks are prepended.

12.8 Modus Ponens

The following rule says that if $[x]^{\cdot}$ is true and if $[x]^{\cdot}$ implies $[y \equiv z]^{\cdot}$ then $[y \equiv z]^{\cdot}$ holds:

[**Mac rule ModusPonens** : $x \equiv \mathsf{T} \vdash x \rightarrow y \equiv z \vdash y \equiv z$]

I hurry to introduce a shorthand notation for applications of this rule:

$$\left[\boxed{x \trianglerighteq y} \doteq \text{ModusPonens} \triangleright y \triangleright x \right].$$

Here you have an example of use:

[**Mac lemma L12.8.1** : $x \rightarrow y \rightarrow z \vdash y \rightarrow x \rightarrow z$]

The lemma says that if $[x]^{\cdot}$ and $[y]^{\cdot}$ imply $[z]^{\cdot}$ then $[y]^{\cdot}$ and $[x]^{\cdot}$ imply $[z]^{\cdot}$.

[**Mac proof of** L12.8.1**:**

L01 :	Premise $\triangleright$	$x \rightarrow y \rightarrow z$	;
L02 :	Block $\triangleright$	Begin	;
L03 :	Hypothesis $\triangleright$	y	;
L04 :	Block $\triangleright$	Begin	;
L05 :	Hypothesis $\triangleright$	x	;
L06 :	$L1 \trianglerighteq L5 \triangleright$	$y \rightarrow z$	;
L07 :	$L6 \trianglerighteq L3 \triangleright$	z	;
L08 :	Block $\triangleright$	End	;
L09 :	Block $\triangleright$	End	;
L10 :	Repetition $\triangleright$ L7 $\triangleright$	$y \rightarrow x \rightarrow z$	]

Exercise 12.8.2 Prove the following:

[**Mac lemma L12.8.3** : $x \rightarrow y \vdash x \rightarrow x \rightarrow y$]

[**Mac lemma L12.8.4** : $x \rightarrow x \rightarrow y \vdash x \rightarrow y$.]

12.9 Proofs that end inside a block

Here you have another proof of $[\text{L12.8.1}]^{\cdot}$:

[**Mac proof of** L12.8.1**:**

L1 :	Premise $\triangleright$	$x \rightarrow y \rightarrow z$	;
L2 :	Block $\triangleright$	Begin	;
L3 :	Hypothesis $\triangleright$	y	;
L4 :	Block $\triangleright$	Begin	;
L5 :	Hypothesis $\triangleright$	x	;
L6 :	$L1 \trianglerighteq L5 \triangleright$	$y \rightarrow z$	;
L7 :	$L6 \trianglerighteq L3 \triangleright$	z	;
L8 :	Block $\triangleright$	End	;
L9 :	Block $\triangleright$	End	]

$[x \trianglerighteq y]^{\cdot}$ x modus ponens y

The last line of the proof is Line [L7]˙ since Line [L8]˙ and Line [L9]˙ just mark the end of two proof blocks. Hence, Line [L7]˙ contains the conclusion of the proof. Seen from the outside, Line [L7]˙ reads [$y \to x \to z$]˙, so the proof is indeed a proof of [L12.8.1]˙.

12.10 Hypotheses that do not begin a block

Until now, each hypothesis has been stated right after the beginning of a proof block. However, I choose to allow hypotheses to occur anywhere in a proof. A hypothesis is in effect until the end of the smallest enclosing block, or until the end of the proof if there is no enclosing block. Here you have an example of use:

[**Mac proof of** L12.8.1:

L1 :	Premise ▷	$x \to y \to z$	;
L2 :	Hypothesis ▷	y	;
L3 :	Hypothesis ▷	x	;
L4 :	L1 ⊵ L3 ▷	$y \to z$	;
L5 :	L4 ⊵ L2 ▷	z	]

In the proof the hypothesis [y]˙ is in effect from Line [L2]˙ to Line [L5]˙ and the hypothesis [x]˙ is in effect from Line [L3]˙ to Line [L5]˙.

Exercise 12.10.1 Prove the following:

[**Mac lemma L12.10.2** : $x \in \mathbf{N} \vdash y \in \mathbf{N} \to z \in \mathbf{N} \to x + y + z \in \mathbf{N}$]

12.11 Variable fixation

Deduction has a peculiarity you need to be aware of: All variables that occur free in a hypothesis behave as if they were constants within the scope of the assumption. Here you have an example of a faulty lemma and proof:

[**Mac lemma L12.11.1** : $T \equiv F$]˙

[**Mac proof of** L12.11.1:

L01 :	Block ▷	Begin	;
L02 :	Hypothesis ▷	x	;
L03 :	Repetition ▷ L2 ▷	$x \equiv T$	;
L04 :	Replace ▷ L3 ▷	$F \equiv T$	;
L05 :	Block ▷	End	;
L06 :	Repetition ▷ L4 ▷	$x \to F \equiv T$	;
L07 :	Replace ▷ L6 ▷	$T \to F \equiv T$	;
L08 :	Reflexivity ▷	T	;
L09 :	L7 ⊵ L8 ▷	$F \equiv T$	;
L10 :	Commutativity ▷ L9 ▷	$T \equiv F$	]˙

The error is in Line [L4]. The variable [x] occurs free in the assumtpion in Line [L2]. Therefore, the variable [x] is *fixed* between Line [L1] and Line [L5]. In other words, [x] behaves as if it were a constant and not a variable between Line [L1] and Line [L5]. [Replace] allows to replace variables by terms, but since [x] behaves as if it were a constant, [Replace] cannot be applied to it. I will refer to this peculiarity as *variable fixation*.

Note that the instantiation in Line [L7] is perfectly valid. [x] is merely fixed from Line [L1] to Line [L5].

Exercise 12.11.2 Is the following correct:

[**Mac lemma VariableFixation 12.11.3** : $x \in \mathbf{N} \rightarrow \lambda x.x \in \mathbf{N} \equiv \lambda x.T$]

[**Mac proof of** VariableFixation 12.11.3**:**

L1 :	Hypothesis ▷	$x \in \mathbf{N}$	;
L2 :	Repetition ▷ L1 ▷	$x \in \mathbf{N} \equiv T$	;
L3 :	LambdaSub ▷ L2 ▷	$\lambda x.x \in \mathbf{N} \equiv \lambda x.T$	]

12.12 Answers

Answer 12.4.2

[**Mac proof of** L12.4.3**:**

L1 :	Block ▷	Begin	;
L2 :	Hypothesis ▷	$x \in \mathbf{F}$	;
L3 :	IfF ▷ L2 ▷	$\text{if}(x, y, z) \equiv z$	;
L4 :	Block ▷	End	;
L5 :	Repetition ▷ L3 ▷	$x \in \mathbf{F} \rightarrow \text{if}(x, y, z) \equiv z$	]

Answer 12.5.2

[**Mac proof of** L12.5.3**:**

L01 :	Block ▷	Begin	;
L02 :	Hypothesis ▷	$x \in \mathbf{T}$	;
L03 :	Block ▷	Begin	;
L04 :	Hypothesis ▷	$y \in \mathbf{N}$	;
L05 :	IfT ▷ L2 ▷	$\text{if}(x, y, z) \equiv y$	;
L06 :	Reverse' ▷ L5 ▷ L4 ▷	$\text{if}(x, y, z) \in \mathbf{N}$	;
L07 :	Block ▷	End	;
L08 :	Repetition ▷ L6 ▷	$y \in \mathbf{N} \rightarrow \text{if}(x, y, z) \in \mathbf{N}$	;
L09 :	Block ▷	End	;
L10 :	Repetition ▷ L8 ▷	$x \in \mathbf{T} \rightarrow y \in \mathbf{N} \rightarrow \text{if}(x, y, z) \in \mathbf{N}$	]

Answer 12.6.1

fixed variable
variable fixation

[**Mac proof of** L12.6.2:

L01 :	Block ▷	Begin	;
L02 :	Hypothesis ▷	$x \in \mathbf{N}$	;
L03 :	Block ▷	Begin	;
L04 :	Hypothesis ▷	$y \in \mathbf{N}$	;
L05 :	Block ▷	Begin	;
L06 :	Hypothesis ▷	$z \in \mathbf{N}$	;
L07 :	SubtypeNZ ▷ L2 ▷	$x \in \mathbf{Z}$	;
L08 :	Block ▷	End	;
L09 :	Block ▷	End	;
L10 :	Block ▷	End	;
L11 :	Repetition ▷ L7 ▷	$x \in \mathbf{N} \to y \in \mathbf{N} \to z \in \mathbf{N} \to x \in \mathbf{Z}$	]

Answer 12.8.2

[**Mac proof of** L12.8.3:

L1 :	Premise ▷	$x \to y$	;
L2 :	Block ▷	Begin	;
L3 :	Hypothesis ▷	x	;
L4 :	Block ▷	Begin	;
L5 :	Hypothesis ▷	x	;
L6 :	L1 $\trianglerighteq$ L5 ▷	y	;
L7 :	Block ▷	End	;
L8 :	Block ▷	End	;
L9 :	Repetition ▷ L6 ▷	$x \to x \to y$	]

[**Mac proof of** L12.8.4:

L1 :	Premise ▷	$x \to x \to y$	;
L2 :	Block ▷	Begin	;
L3 :	Hypothesis ▷	x	;
L4 :	L1 $\trianglerighteq$ L3 ▷	$x \to y$	;
L5 :	L4 $\trianglerighteq$ L3 ▷	y	;
L6 :	Block ▷	End	;
L7 :	Repetition ▷ L5 ▷	$x \to y$	]

Answer 12.10.1

[**Mac proof of** L12.10.2:

L1 :	Premise ▷	$x \in \mathbf{N}$	;
L2 :	Hypothesis ▷	$y \in \mathbf{N}$	;
L3 :	Hypothesis ▷	$z \in \mathbf{N}$	;
L4 :	Type**N**+**N** ▷ L1 ▷ L2 ▷	$x + y \in \mathbf{N}$	;
L5 :	Type**N**+**N** ▷ L4 ▷ L3 ▷	$x + y + z \in \mathbf{N}$	]

Answer 12.11.2 No. Line [L3]˙ is in error.

Chapter 13

Proof techniques

13.1 Brute force

If x and y are terms and if $x \equiv y$ is a theorem, i.e. if there exists a proof of $x \equiv y$, then you can find a proof of $x \equiv y$ by *brute force* as follows:

Step 1. Set n to 1.

Step 2. Write down all sequences of characters that are n characters long.

Step 3. If one of the sequences of characters from Step 2 is a proof of $x \equiv y$ then stop. Otherwise, add one to n and go to Step 2.

Mathematicians use about 1000 different characters, so you have to write down about 1000^{n} sequences of characters in Step 2. If the shortest proof of $x \equiv y$ is e.g. 100 characters long, then you have to write down 1000^{100} sequences of characters in Step 2 before you find the proof. 1000^{100} is larger than the number of elementary particles in the visible part of the universe, so you may find it difficult to find a large enough piece of paper for performing Step 2.

In other words, the brute force method works in principle, but the brute force method is not *feasible*.

Whenever you have to prove something, you must use your intuition and intelligence to “guess” the proof. Once you have written down the proof it is straightforward to check whether or not it is correct, but writing the proof is not at all trivial.

To develop the skill of theorem proving you need to practice and you need a bag full of feasible proof techniques. A good mathematician develops his or her own proof techniques. In this chapter I present some proof techniques you can start with.

---

brute force
feasible

13.2 Backchaining

Backchaining is a rather powerful proof technique. In backchaining, you write the proof backwards, starting with the conclusion. Here you have an example:

[**Mac lemma L13.2.1** : $1 + 2 \cdot 3 \in \mathbf{N}$]

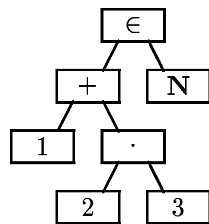
A proof of L13.2.1 must have [$1 + 2 \cdot 3 \in \mathbf{N}$]˙ as its conclusion:

[**Mac proof of** L13.2.1**:**

A : $\cdots \rhd$ $1 + 2 \cdot 3 \in \mathbf{N}$]˙

I have chosen [A]˙ as the name for the last proof line.

The next task is to prove [$1 + 2 \cdot 3 \in \mathbf{N}$]˙. The syntax tree of that reads:



If you look at the root node of the conclusion and the two nodes below it you see that you have to prove something of form

[$\mathcal{A} + \mathcal{B} \in \mathbf{N}$]˙.

Now you scan your mind (or Appendix A) for an inference rule whose conclusion has form [$\mathcal{A} + \mathcal{B} \in \mathbf{N}$]˙. You probably find Type**N**+**N** sooner or later:

[$\mathsf{x} \in \mathbf{N} \vdash \mathsf{y} \in \mathbf{N} \vdash \mathsf{x} + \mathsf{y} \in \mathbf{N}$]˙.

You can now write the argumentation of the last line of the proof:

[**Mac proof of** L13.2.1**:**

A : Type**N**+**N** $\rhd$ B $\rhd$ C $\rhd$ $1 + 2 \cdot 3 \in \mathbf{N}$]˙

Type**N**+**N** has two premisses. In the proof above, I have called the premisses [B]˙ and [C]˙. Now recall that Type**N**+**N** reads

[$\mathsf{x} \in \mathbf{N} \vdash \mathsf{y} \in \mathbf{N} \vdash \mathsf{x} + \mathsf{y} \in \mathbf{N}$]˙.

Since the conclusion reads [$1 + 2 \cdot 3 \in \mathbf{N}$]˙, you can see that [x]˙ must be [1]˙ and [y]˙ must be [$2 \cdot 3$]˙. Hence, Premisse [B]˙ must be [$1 \in \mathbf{N}$]˙ and Premise [C]˙ must be [$2 \cdot 3 \in \mathbf{N}$]˙:

---

backchaining

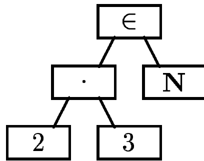
[**Mac proof of** L13.2.1:
B : $\cdots \rhd$ $\qquad 1 \in \mathbf{N}$;
C : $\cdots \rhd$ $\qquad 2 \cdot 3 \in \mathbf{N}$;
A : Type$\mathbf{N}$+$\mathbf{N}$ $\rhd$ B $\rhd$ C $\rhd$ $\qquad 1 + 2 \cdot 3 \in \mathbf{N}$ $]^{\cdot}$

You started out having one problem, namely to prove $[\, 1+2\cdot 3 \in \mathbf{N}\,]^{\cdot}$. You now have two problems, namely to prove $[\, 1 \in \mathbf{N}\,]^{\cdot}$ and $[\, 2\cdot 3 \in \mathbf{N}\,]^{\cdot}$. This may look as if the situation has worsened. However, you started with one complicated problem and now have two simpler ones. In many situations, a complicated problem can be solved by dividing it into several simpler problems and then attacking the simpler problems one at a time. This is known as *divide and conquer*.

When attacking the simpler problems one at a time, I prefer to attack the simplest ones first. Among $[\, 1 \in \mathbf{N}\,]^{\cdot}$ and $[\, 2\cdot 3 \in \mathbf{N}\,]^{\cdot}$, $[\, 1 \in \mathbf{N}\,]^{\cdot}$ is simplest since it is an instance of TypeNumeralInN:

[**Mac proof of** L13.2.1:
B : TypeNumeralInN $\rhd$ $\qquad 1 \in \mathbf{N}$;
C : $\cdots \rhd$ $\qquad 2 \cdot 3 \in \mathbf{N}$;
A : Type$\mathbf{N}$+$\mathbf{N}$ $\rhd$ B $\rhd$ C $\rhd$ $\qquad 1 + 2 \cdot 3 \in \mathbf{N}$ $]^{\cdot}$

The next task is to prove $[\, 2\cdot 3 \in \mathbf{N}\,]^{\cdot}$. The syntax tree of that reads:



If you look at the root node of the conclusion and the two nodes below it you see that you have to prove something of form

$[\, \mathcal{A} \cdot \mathcal{B} \in \mathbf{N}\,]^{\cdot}$.

Now you scan your mind (or Appendix A) for an inference rule whose conclusion has form $[\, \mathcal{A} \cdot \mathcal{B} \in \mathbf{N}\,]^{\cdot}$. You probably find Type$\mathbf{N}$·$\mathbf{N}$ sooner or later:

$[\, \mathsf{x} \in \mathbf{N} \vdash \mathsf{y} \in \mathbf{N} \vdash \mathsf{x} \cdot \mathsf{y} \in \mathbf{N}\,]^{\cdot}$.

You can now write the argumentation of Line $[\, \mathsf{C}\,]^{\cdot}$ of the proof:

[**Mac proof of** L13.2.1:
B : TypeNumeralInN $\rhd$ $\qquad 1 \in \mathbf{N}$;
C : Type$\mathbf{N}$·$\mathbf{N}$ $\rhd$ D $\rhd$ E $\rhd$ $\qquad 2 \cdot 3 \in \mathbf{N}$;
A : Type$\mathbf{N}$+$\mathbf{N}$ $\rhd$ B $\rhd$ C $\rhd$ $\qquad 1 + 2 \cdot 3 \in \mathbf{N}$ $]^{\cdot}$

---

divide and conquer

Type$\mathbf{N}{\cdot}\mathbf{N}$ has two premisses. In the proof above, I have called the premisses $[\, \mathsf{D} \,]^{\cdot}$ and $[\, \mathsf{E} \,]^{\cdot}$. Now recall that Type$\mathbf{N}{\cdot}\mathbf{N}$ reads

$$[\, \mathsf{x} \in \mathbf{N} \vdash \mathsf{y} \in \mathbf{N} \vdash \mathsf{x} \cdot \mathsf{y} \in \mathbf{N} \,]^{\cdot}.$$

Since the conclusion reads $[\, 2 \cdot 3 \in \mathbf{N} \,]^{\cdot}$, you can see that $[\, \mathsf{x} \,]^{\cdot}$ must be $[\, 2 \,]^{\cdot}$ and $[\, \mathsf{y} \,]^{\cdot}$ must be $[\, 3 \,]^{\cdot}$. Hence, Premisse $[\, \mathsf{D} \,]^{\cdot}$ must be $[\, 2 \in \mathbf{N} \,]^{\cdot}$ and Premise $[\, \mathsf{E} \,]^{\cdot}$ must be $[\, 3 \in \mathbf{N} \,]^{\cdot}$:

[**Mac proof of** L13.2.1:

B :	TypeNumeralInN $\triangleright$	$1 \in \mathbf{N}$	;
D :	$\cdots \triangleright$	$2 \in \mathbf{N}$	;
E :	$\cdots \triangleright$	$3 \in \mathbf{N}$	;
C :	Type$\mathbf{N}{\cdot}\mathbf{N}$ $\triangleright$ D $\triangleright$ E $\triangleright$	$2 \cdot 3 \in \mathbf{N}$	;
A :	Type$\mathbf{N}{+}\mathbf{N}$ $\triangleright$ B $\triangleright$ C $\triangleright$	$1 + 2 \cdot 3 \in \mathbf{N}$	]$^{\cdot}$

Now I am back where there are two problems left. Both are very easy, however, since both Line $[\, \mathsf{D} \,]^{\cdot}$ and Line $[\, \mathsf{E} \,]^{\cdot}$ are instances of TypeNumeralInN:

[**Mac proof of** L13.2.1:

B :	TypeNumeralInN $\triangleright$	$1 \in \mathbf{N}$	;
D :	TypeNumeralInN $\triangleright$	$2 \in \mathbf{N}$	;
E :	TypeNumeralInN $\triangleright$	$3 \in \mathbf{N}$	;
C :	Type$\mathbf{N}{\cdot}\mathbf{N}$ $\triangleright$ D $\triangleright$ E $\triangleright$	$2 \cdot 3 \in \mathbf{N}$	;
A :	Type$\mathbf{N}{+}\mathbf{N}$ $\triangleright$ B $\triangleright$ C $\triangleright$	$1 + 2 \cdot 3 \in \mathbf{N}$	]

Now I have solved the problem, which was to prove L13.2.1. It is a custom among mathematicians, however, that they try to polish their results. I cannot do much to enhance the proof above, but I can at least renumber it. First, I decide which line numbers to use:

[**Mac proof of** L13.2.1:

L1 : B :	TypeNumeralInN $\triangleright$	$1 \in \mathbf{N}$	;
L2 : D :	TypeNumeralInN $\triangleright$	$2 \in \mathbf{N}$	;
L3 : E :	TypeNumeralInN $\triangleright$	$3 \in \mathbf{N}$	;
L4 : C :	Type$\mathbf{N}{\cdot}\mathbf{N}$ $\triangleright$ D $\triangleright$ E $\triangleright$	$2 \cdot 3 \in \mathbf{N}$	;
L5 : A :	Type$\mathbf{N}{+}\mathbf{N}$ $\triangleright$ B $\triangleright$ C $\triangleright$	$1 + 2 \cdot 3 \in \mathbf{N}$	]

The proof above is valid, but it is unusual to let two variables denote each line. Then I update the references in the argumentations:

[**Mac proof of** L13.2.1:

L1 : B :	TypeNumeralInN $\triangleright$	$1 \in \mathbf{N}$	;
L2 : D :	TypeNumeralInN $\triangleright$	$2 \in \mathbf{N}$	;
L3 : E :	TypeNumeralInN $\triangleright$	$3 \in \mathbf{N}$	;
L4 : C :	Type$\mathbf{N}{\cdot}\mathbf{N}$ $\triangleright$ L2 $\triangleright$ L3 $\triangleright$	$2 \cdot 3 \in \mathbf{N}$	;
L5 : A :	Type$\mathbf{N}{+}\mathbf{N}$ $\triangleright$ L1 $\triangleright$ L4 $\triangleright$	$1 + 2 \cdot 3 \in \mathbf{N}$	]

Finially, I remove the old names:

[**Mac proof of** L13.2.1:

L1 :	TypeNumeralInN ▷	$1 \in \mathbf{N}$	;
L2 :	TypeNumeralInN ▷	$2 \in \mathbf{N}$	;
L3 :	TypeNumeralInN ▷	$3 \in \mathbf{N}$	;
L4 :	Type**N**·**N** ▷ L2 ▷ L3 ▷	$2 \cdot 3 \in \mathbf{N}$	;
L5 :	Type**N**+**N** ▷ L1 ▷ L4 ▷	$1 + 2 \cdot 3 \in \mathbf{N}$	]

13.3 Backward application

I will says that a rule is *suited for backchaining* if all variables that occur in the premisses also occur in the conclusion. As an example,

$$[\, \mathsf{x} \in \mathbf{N} \vdash \mathsf{y} \in \mathbf{N} \vdash \mathsf{x} + \mathsf{y} \in \mathbf{N} \,]^{\cdot}$$

is suited for backchaining because the variables $[\, \mathsf{x} \,]^{\cdot}$ and $[\, \mathsf{y} \,]^{\cdot}$ that occur in the premisses also occur in the conclusion. In contrast,

$$[\, \mathsf{x} \vdash \mathsf{x} \rightarrow \mathsf{y} \vdash \mathsf{y} \,]^{\cdot}$$

is unsuited for backchaining because the variable $[\, \mathsf{x} \,]^{\cdot}$ occurs in the premisses but not in the conclusion. All axioms are suited for backchaining.

If $[\, \mathcal{R} \,]^{\cdot}$ is a rule that is suited for backchaining and if $[\, \mathcal{S} \,]^{\cdot}$ is an equation, then I will say that $[\, \mathcal{R} \,]^{\cdot}$ is suited for backchaining of $[\, \mathcal{S} \,]^{\cdot}$ if $[\, \mathcal{S} \,]^{\cdot}$ is an instance of the conclusion of $[\, \mathcal{R} \,]^{\cdot}$. As an example,

$$[\, \mathsf{x} \in \mathbf{N} \vdash \mathsf{y} \in \mathbf{N} \vdash \mathsf{x} + \mathsf{y} \in \mathbf{N} \,]^{\cdot}$$

is suited for backchaining of

$$[\, 1 + 2 \cdot 3 \in \mathbf{N} \,]^{\cdot}$$

because $[\, 1 + 2 \cdot 3 \in \mathbf{N} \,]^{\cdot}$ is an instance of $[\, \mathsf{x} + \mathsf{y} \in \mathbf{N} \,]^{\cdot}$.

As another example, TypeNumeralInN is suited for backchaining of $[\, 1 \in \mathbf{N} \,]^{\cdot}$ because $[\, 1 \in \mathbf{N} \,]^{\cdot}$ is an intance of TypeNumeralInN. As a final example,

$$[\, \mathsf{x} \in \mathbf{N} \vdash \mathsf{y} \in \mathbf{N} \vdash \mathsf{x} \cdot \mathsf{y} \in \mathbf{N} \,]^{\cdot}$$

is unsuited for backchaining of

$$[\, 1 + 2 \cdot 3 \in \mathbf{N} \,]^{\cdot}$$

because $[\, 1 + 2 \cdot 3 \in \mathbf{N} \,]^{\cdot}$ is not an instance of $[\, \mathsf{x} \cdot \mathsf{y} \in \mathbf{N} \,]^{\cdot}$.

If a rule $[\, \mathcal{R} \,]^{\cdot}$ is suited for backchaining of a formula $[\, \mathcal{S} \,]^{\cdot}$, then you can instantiate $[\, \mathcal{R} \,]^{\cdot}$ such that the conclusion of $[\, \mathcal{R} \,]^{\cdot}$ is identical to $[\, \mathcal{S} \,]^{\cdot}$. If you do so, then I will refer to the list of premisses of the instantiated rule as the

---

suited for backchaining

backward application of $[\,\mathcal{R}\,]^{\cdot}$ *to* $[\,\mathcal{S}\,]^{\cdot}$. As an example, the backward application of

$$[\,\mathsf{x} \in \mathbf{N} \vdash \mathsf{y} \in \mathbf{N} \vdash \mathsf{x} + \mathsf{y} \in \mathbf{N}\,]^{\cdot}$$

to

$$[\,1 + 2 \cdot 3 \in \mathbf{N}\,]^{\cdot}$$

is

$$[\,1 \in \mathbf{N}; 2 \cdot 3 \in \mathbf{N}\,]^{\cdot}.$$

In general, backward application of a rule $[\,\mathcal{R}\,]^{\cdot}$ to a formula $[\,\mathcal{S}\,]^{\cdot}$ either fails or succeeds. If it succeeds, the result is a list that contains zero, one, or more formulas. The number of formulas equals the number of premisses in $[\,\mathcal{R}\,]^{\cdot}$. If the list contains more than one formula, I use semicolons to separate them.

The process of backchaining is thus:

Step 1 Find a rule that is suited for backchaining of the given conclusion.

Step 2 Apply the rule backwards to the conclusion to obtain a list of formulas.

Step 3 Apply backchaining to each formula obtained in Step 2.

The list of formulas from Step 2 is empty if the rule is an axiom (i.e. has no premisses). In that case there is no work to do in Step 3.

Exercise 13.3.1 Which of the following are suited for backchaining:

a $[\,\mathsf{x} \in \mathbf{Z} \vdash \mathsf{y} \in \mathbf{Z} \vdash \mathsf{x} + \mathsf{y} \in \mathbf{Z}\,]^{\cdot}$.

b $[\,\mathsf{x} \in \mathbf{N} \vdash \mathsf{x} \in \mathbf{Z}\,]^{\cdot}$.

c $[\,\mathsf{x} \in \mathbf{N} \vdash \mathsf{y} \in \mathbf{N} \vdash \mathsf{x} + \mathsf{y} = \mathsf{y} \vdash \mathsf{x} = 0\,]^{\cdot}$.

Exercise 13.3.2 Apply $[\,\mathsf{n} \in \mathbf{N} \rightarrow (\mathsf{n} = 0) \in \mathbf{T} \rightarrow \mathcal{A} \vdash \mathsf{n} \in \mathbf{N} \rightarrow (\mathsf{n} = 0) \in \mathbf{F} \rightarrow \langle \mathcal{A} \mid \mathsf{n}{:=}\mathsf{n} - 1 \rangle \rightarrow \mathcal{A} \vdash \mathsf{n} \in \mathbf{N} \rightarrow \mathcal{A}\,]^{\cdot}$ backwards to $[\,\mathsf{n} \in \mathbf{N} \rightarrow \mathsf{n}! \in \mathbf{N}\,]^{\cdot}$.

13.4 Backward application of deduction

Deduction is suited for backchaining of $[\,\mathcal{A} \rightarrow \mathcal{B}\,]^{\cdot}$. Here you have an example of *backward application of deduction*:

[**Mac lemma L13.4.1** : $\mathsf{x} \in \mathbf{N} \rightarrow \mathsf{x} + 2 \in \mathbf{N}$]

---

backward application of rule to formula
backward application of deduction

The first step in backchaining is to write the conclusion:

[**Mac proof of** L13.4.1:

A :	$\cdots \triangleright$	$x \in \mathbf{N} \rightarrow x + 2 \in \mathbf{N}$	]

If you decide to prove the implication by deduction then the next step is to apply deduction backwards:

[**Mac proof of** L13.4.1:

B :	Block $\triangleright$	Begin	;
C :	Hypothesis $\triangleright$	$x \in \mathbf{N}$	;
D :	$\cdots \triangleright$	$x + 2 \in \mathbf{N}$	;
E :	Block $\triangleright$	End	;
A :	Repetition $\triangleright$ D $\triangleright$	$x \in \mathbf{N} \rightarrow x + 2 \in \mathbf{N}$	]

I hope you can see how backward application of deduction works from the example above. The idea is to introduce a new block, to state the hypothesis of the implication as a hypothesis at the beginning of the block, and to state the consequence of the implication at the end of the block. The implication may then be proved by Repetition.

The only remaining task in the proof above is to prove [$x + 2 \in \mathbf{N}$] (you don't have to prove [$x \in \mathbf{N}$] in Line [C] because that proof line already has an argumentation). To prove [$x+2 \in \mathbf{N}$] you can apply Type**N**+**N** backwards to [$x + 2 \in \mathbf{N}$]:

[**Mac proof of** L13.4.1:

B :	Block $\triangleright$	Begin	;
C :	Hypothesis $\triangleright$	$x \in \mathbf{N}$	;
F :	$\cdots \triangleright$	$x \in \mathbf{N}$	;
G :	$\cdots \triangleright$	$2 \in \mathbf{N}$	;
D :	Type**N**+**N** $\triangleright$ F $\triangleright$ G $\triangleright$	$x + 2 \in \mathbf{N}$	;
E :	Block $\triangleright$	End	;
A :	Repetition $\triangleright$ D $\triangleright$	$x \in \mathbf{N} \rightarrow x + 2 \in \mathbf{N}$	]

Line [F] is identical to Line [C], so I delete Line [F] and change the reference to Line [F]:

[**Mac proof of** L13.4.1:

B :	Block $\triangleright$	Begin	;
C :	Hypothesis $\triangleright$	$x \in \mathbf{N}$	;
G :	$\cdots \triangleright$	$2 \in \mathbf{N}$	;
D :	Type**N**+**N** $\triangleright$ C $\triangleright$ G $\triangleright$	$x + 2 \in \mathbf{N}$	;
E :	Block $\triangleright$	End	;
A :	Repetition $\triangleright$ D $\triangleright$	$x \in \mathbf{N} \rightarrow x + 2 \in \mathbf{N}$	]

The only remaining task is to prove $[\, 2 \in \mathbf{N} \,]^{\cdot}$. $[\, 2 \in \mathbf{N} \,]^{\cdot}$ is an instance of TypeNumeralInN, so I can now complete the proof:

[**Mac proof of** L13.4.1**:**

B :	Block $\rhd$	Begin	;
C :	Hypothesis $\rhd$	$\quad \mathsf{x} \in \mathbf{N}$	;
G :	TypeNumeralInN $\rhd$	$\quad 2 \in \mathbf{N}$	;
D :	Type$\mathbf{N}$+$\mathbf{N}$ $\rhd$ C $\rhd$ G $\rhd$	$\quad \mathsf{x} + 2 \in \mathbf{N}$	;
E :	Block $\rhd$	End	;
A :	Repetition $\rhd$ D $\rhd$	$\mathsf{x} \in \mathbf{N} \to \mathsf{x} + 2 \in \mathbf{N}$	]

To polish the proof, I renumber it:

[**Mac proof of** L13.4.1**:**

$\mathsf{L1}$:	Block $\rhd$	Begin	;
$\mathsf{L2}$:	Hypothesis $\rhd$	$\quad \mathsf{x} \in \mathbf{N}$	;
$\mathsf{L3}$:	TypeNumeralInN $\rhd$	$\quad 2 \in \mathbf{N}$	;
$\mathsf{L4}$:	Type$\mathbf{N}$+$\mathbf{N}$ $\rhd$ $\mathsf{L2}$ $\rhd$ $\mathsf{L3}$ $\rhd$	$\quad \mathsf{x} + 2 \in \mathbf{N}$	;
$\mathsf{L5}$:	Block $\rhd$	End	;
$\mathsf{L6}$:	Repetition $\rhd$ $\mathsf{L4}$ $\rhd$	$\mathsf{x} \in \mathbf{N} \to \mathsf{x} + 2 \in \mathbf{N}$	]

I can also write the proof a bit shorter:

[**Mac proof of** L13.4.1**:**

$\mathsf{L1}$:	Hypothesis $\rhd$	$\mathsf{x} \in \mathbf{N}$	;
$\mathsf{L2}$:	TypeNumeralInN $\rhd$	$2 \in \mathbf{N}$	;
$\mathsf{L3}$:	Type$\mathbf{N}$+$\mathbf{N}$ $\rhd$ $\mathsf{L1}$ $\rhd$ $\mathsf{L2}$ $\rhd$	$\mathsf{x} + 2 \in \mathbf{N}$	]

The proof is a correct proof of $[\, \mathsf{x} \in \mathbf{N} \to \mathsf{x} + 2 \in \mathbf{N} \,]^{\cdot}$ because the hypothesis in Line $[\, \mathsf{L1} \,]^{\cdot}$ is in effect in Line $[\, \mathsf{L3} \,]^{\cdot}$ so that Line $[\, \mathsf{L3} \,]^{\cdot}$ reads $[\, \mathsf{x} \in \mathbf{N} \to \mathsf{x} + 2 \in \mathbf{N} \,]^{\cdot}$ when seen from outside the proof.

The secret behind short and/or elegant proofs are: Virtually nobody can just dream up an elegant proof. What mathematicians really do is that they first construct ugly proofs as best they can. Once a good mathematician has found a proof of something interesting, he or she may spend quite some time rewriting the proof over and over again, making improvements to it, before the proof is published. After that, other mathematicians may go on and publish further improvements with more clever definitions, better formulations of theorems, and better proofs. When you read a mathematics book on e.g. mathematical analysis, you are confronted with the result of the improvements made by several generations of mathematicians. There is no reason to get depressed if your proofs are less elegant than those you find in a book on e.g. mathematical analysis.

Exercise 13.4.2 Prove the following:

[**Mac lemma L13.4.3** : $\mathsf{x} \in \mathbf{N} \to \mathsf{x} \cdot 2 + 3 \in \mathbf{N}$]

[**Mac lemma L13.4.4** : $\mathsf{x} \in \mathbf{N} \to \mathsf{y} \in \mathbf{N} \to \mathsf{x} \cdot \mathsf{y} + \mathsf{x} \cdot \mathsf{y} \in \mathbf{N}$]

13.5 Algebra

You have seen two kinds of proofs in this book: algebraic proofs and derivation proofs. I now merge the two together in that I allow algebraic proofs to occur inside derivation proofs. Here you have an example:

[**Mac lemma L13.5.1** : $(n = 0) \in \mathbf{F} \to n! \equiv n \cdot (n-1)!$]

[**Mac proof of** L13.5.1**:**

L01 :	Block ▷	Begin	;
L02 :	Hypothesis ▷	$(n = 0) \in \mathbf{F}$	;
L03 :	Block ▷	Begin	;
L04 :	Algebra ▷	$n!$	;
L05 :	Definition ▷	$\mathrm{if}(n = 0, 1, n \cdot (n-1)!)$	;
L06 :	IfF ▷ L2 ▷	$n \cdot (n-1)!$	;
L07 :	Block ▷	End	;
L08 :	Repetition ▷ L6 ▷	$n! \equiv n \cdot (n-1)!$	;
L09 :	Block ▷	End	;
L10 :	Repetition ▷ L8 ▷	$(n = 0) \in \mathbf{F} \to n! \equiv n \cdot (n-1)!$	]

In the proof, Line [L3]˙ to Line [L7]˙ constitute an *algebraic block*. An algebraic block has form

[A :	Block ▷	Begin	;
B_0 :	Algebra ▷	$\mathcal{A}_0$	;
B_1 :	$\mathcal{D}_1$ ▷	$\mathcal{A}_1$	;
B_2 :	$\mathcal{D}_2$ ▷	$\mathcal{A}_2$	;
B_3 :	$\mathcal{D}_3$ ▷	$\mathcal{A}_3$	;
⋮	⋮	⋮	
B_{n-1} :	$\mathcal{D}_{n-1}$ ▷	$\mathcal{A}_{n-1}$	;
B_n :	$\mathcal{D}_n$ ▷	$\mathcal{A}_n$	;
C :	Block ▷	End	]°

In the block, [$\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_n$]° must be terms and [$\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_n$]° must be argumentations. When refering to Line [B_j]˙ from outside the block, Line [B_j]˙ reads [$\mathcal{A}_0 \equiv \mathcal{A}_j$]˙. As an example, in the proof of L13.5.1, Line [L8]˙ refers to Line [L6]˙. From the point of view of Line [L8]˙, Line [L6]˙ reads [$n! \equiv n \cdot (n-1)!$]˙ where the left hand side of the equal sign comes from Line [L4]˙ and the right hand side comes from Line [L6]˙.

The argumentation [$\mathcal{D}_j$]˙ must verify [$\mathcal{A}_{j-1} \equiv \mathcal{A}_j$]˙. You can see how that works in the proof of L13.5.1: The argumentation [Definition]˙ in Line [L5]˙ verifies

[$n! \equiv \mathrm{if}(n = 0, 1, n \cdot (n-1)!)$]˙.

algebraic block

The argumentation [IfF ▷ L2] in Line [L6] verifies

$$[\,\text{if}(n = 0, 1, n \cdot (n - 1)!) \equiv n \cdot (n - 1)!\,].$$

Exercise 13.5.2 Prove

[**Mac lemma L13.5.3** : $n \in \mathbf{N} \rightarrow \langle n, n, n, n \rangle$ tail head $\in \mathbf{N}$]

Exercise 13.5.4 Find the errors (if any) in the following proof:

[**Mac lemma L13.5.5** : $x \in \mathbf{F} \rightarrow f_9(x)$]

[**Mac proof of** L13.5.5:

L01 :	Block ▷	Begin	;
L02 :	Hypothesis ▷	$x \in \mathbf{F}$	;
L03 :	Type¬**F** ▷ L2 ▷	$\neg x \in \mathbf{T}$	;
L04 :	Block ▷	Begin	;
L05 :	Algebra ▷	$f_9(x)$	;
L06 :	Definition ▷	$\text{if}(x, T, f_9(\neg x))$	;
L07 :	IfF ▷ L2 ▷	$f(\neg x)$	;
L08 :	Definition ▷	$\text{if}(\neg x, T, f_9(\neg\neg x))$	;
L09 :	IfT ▷ L3 ▷	T	;
L10 :	Block ▷	End	;
L11 :	Block ▷	End	;
L12 :	Repetition ▷ L9 ▷	$x \in \mathbf{F} \rightarrow f_9(x)$	]

13.6 Answers

Answer 13.3.1 Rule a, and b are suited for backchaining. Rule c is unsuited because [y] in the premises does not occur in the conclusion.

Answer 13.3.2 [$n \in \mathbf{N} \rightarrow (n = 0) \in \mathbf{T} \rightarrow n! \in \mathbf{N}; n \in \mathbf{N} \rightarrow (n = 0) \in \mathbf{F} \rightarrow (n - 1)! \in \mathbf{N} \rightarrow n! \in \mathbf{N}$].

Answer 13.4.2 Direct application of backchaining gives:

[**Mac proof of** L13.4.3:

L1 :	Block ▷	Begin	;
L2 :	Hypothesis ▷	$x \in \mathbf{N}$	;
L3 :	TypeNumeralInN ▷	$2 \in \mathbf{N}$	;
L4 :	Type**N**·**N** ▷ L2 ▷ L3 ▷	$x \cdot 2 \in \mathbf{N}$	;
L5 :	TypeNumeralInN ▷	$3 \in \mathbf{N}$	;
L6 :	Type**N**+**N** ▷ L4 ▷ L5 ▷	$x \cdot 2 + 3 \in \mathbf{N}$	;
L7 :	Block ▷	End	;
L8 :	Repetition ▷ L6 ▷	$x \in \mathbf{N} \rightarrow x \cdot 2 + 3 \in \mathbf{N}$	]

A shorter version reads:

[**Mac proof of** L13.4.3:

L1 :	Hypothesis ▷	$x \in \mathbf{N}$	;
L2 :	TypeNumeralInN ▷	$2 \in \mathbf{N}$	;
L3 :	Type**N**·**N** ▷ L1 ▷ L2 ▷	$x \cdot 2 \in \mathbf{N}$	;
L4 :	TypeNumeralInN ▷	$3 \in \mathbf{N}$	;
L5 :	Type**N**+**N** ▷ L3 ▷ L4 ▷	$x \cdot 2 + 3 \in \mathbf{N}$	]

For L13.4.4, idiotic application of backchaining could give something like:

[**Mac proof of** L13.4.4:

L1 :	Block ▷	Begin	;
L2 :	Hypothesis ▷	$x \in \mathbf{N}$	;
L3 :	Block ▷	Begin	;
L4 :	Hypothesis ▷	$y \in \mathbf{N}$	;
L5 :	Type**N**·**N** ▷ L2 ▷ L4 ▷	$x \cdot y \in \mathbf{N}$	;
L6 :	Type**N**·**N** ▷ L2 ▷ L4 ▷	$x \cdot y \in \mathbf{N}$	;
L7 :	Type**N**+**N** ▷ L5 ▷ L6 ▷	$x \cdot y + x \cdot y \in \mathbf{N}$	;
L8 :	Block ▷	End	;
L9 :	Repetition ▷ L7 ▷	$y \in \mathbf{N} \to x \cdot y + x \cdot y \in \mathbf{N}$	;
L10 :	Block ▷	End	;
L11 :	Repetition ▷ L9 ▷	$x{\in}\mathbf{N}{\to}y{\in}\mathbf{N}{\to}x{\cdot}y{+}x{\cdot}y{\in}\mathbf{N}$	]

In the proof, Line [L5] and Line [L6] are identical, which is of course superflous. It is obvious to merge the two lines into one. In addition to merging of identical lines, one may go on and minimise the number of blocks:

[**Mac proof of** L13.4.4:

L1 :	Hypothesis ▷	$x \in \mathbf{N}$	;
L2 :	Hypothesis ▷	$y \in \mathbf{N}$	;
L3 :	Type**N**·**N** ▷ L1 ▷ L2 ▷	$x \cdot y \in \mathbf{N}$	;
L4 :	Type**N**+**N** ▷ L3 ▷ L3 ▷	$x \cdot y + x \cdot y \in \mathbf{N}$	]

Answer 13.5.2 Here is one possible proof:

[**Mac proof of** L13.5.3:

L1 :	Hypothesis ▷	$n \in \mathbf{N}$	;
L2 :	Block ▷	Begin	;
L3 :	Algebra ▷	$\langle n, n, n, n \rangle$ tail head	;
L4 :	TailPair ▷	$\langle n, n, n \rangle$ head	;
L5 :	HeadPair ▷	n	;
L6 :	Block ▷	End	;
L7 :	Commutativity ▷ L5 ▷	$n \equiv \langle n, n, n, n \rangle$ tail head	;
L8 :	Replace' ▷ L7 ▷ L1 ▷	$\langle n, n, n, n \rangle$ tail head $\in \mathbf{N}$	]

Answer 13.5.4 There are no errors in the proof. In particular, Line [L12] is a correct repetition of Line [L9]: Seen from the outside, Line [L9] reads [$x \in \mathbf{F} \to f_9(x) \equiv \mathsf{T}$]. Line [L12] reads [$x \in \mathbf{F} \to f_9(x)$] which is shorthand for [$x \in \mathbf{F} \to f_9(x) \equiv \mathsf{T}$].

Chapter 14

Logic

14.1 Proof by cases

$[\mathcal{A} \equiv \mathcal{B}]^{\cdot}$ is true for all $[\mathsf{x}]^{\cdot}$ that represent truth values if $[\mathcal{A} \equiv \mathcal{B}]^{\cdot}$ is true for all $[\mathsf{x}]^{\cdot}$ that represent truth as well as all $[\mathsf{x}]^{\cdot}$ that represent falsehood:

[**Mac rule Cases** : $\mathsf{x} \in \mathbf{T} \to \mathcal{A} \equiv \mathcal{B} \vdash \mathsf{x} \in \mathbf{F} \to \mathcal{A} \equiv \mathcal{B} \vdash \mathsf{x} \in \mathbf{B} \to \mathcal{A} \equiv \mathcal{B}$]

Here you have an example of use:

[**Mac lemma L14.1.1** : $\mathsf{x} \in \mathbf{B} \to \mathsf{x} \vee \neg\mathsf{x} \in \mathbf{T}$]

[**Mac proof of** L14.1.1**:**

L01 :	Block ▷	Begin	;
L02 :	Hypothesis ▷	$\mathsf{x} \in \mathbf{T}$	;
L03 :	Type$\neg\mathbf{T}$ ▷ L2 ▷	$\neg\mathsf{x} \in \mathbf{F}$	;
L04 :	Type$\mathbf{T}\vee\mathbf{F}$ ▷ L2 ▷ L3 ▷	$\mathsf{x} \vee \neg\mathsf{x} \in \mathbf{T}$	;
L05 :	Block ▷	End	;
L06 :	Block ▷	Begin	;
L07 :	Hypothesis ▷	$\mathsf{x} \in \mathbf{F}$	;
L08 :	Type$\neg\mathbf{F}$ ▷ L7 ▷	$\neg\mathsf{x} \in \mathbf{T}$	;
L09 :	Type$\mathbf{F}\vee\mathbf{T}$ ▷ L7 ▷ L8 ▷	$\mathsf{x} \vee \neg\mathsf{x} \in \mathbf{T}$	;
L10 :	Block ▷	End	;
L11 :	Cases ▷ L4 ▷ L9 ▷	$\mathsf{x} \in \mathbf{B} \to \mathsf{x} \vee \neg\mathsf{x} \in \mathbf{T}$	]

I constructed the proof above by backchaining. Here you have how I did:

Cases and Deduction are both suited for backchaining of $[\mathsf{x} \in \mathbf{B} \to \mathsf{x} \vee \neg\mathsf{x}]^{\cdot}$. Backward application of deduction leads nowhere, but backward application of Cases to $[\mathsf{x} \in \mathbf{B} \to \mathsf{x} \vee \neg\mathsf{x}]^{\cdot}$ gives

$$[\mathsf{x} \in \mathbf{T} \to \mathsf{x} \vee \neg\mathsf{x}; \mathsf{x} \in \mathbf{F} \to \mathsf{x} \vee \neg\mathsf{x}]^{\cdot}.$$

This is what I did in Line $[\mathsf{L11}]^{\cdot}$.

Then I applied deduction backwards to [$x \in \mathbf{T} \to x \vee \neg x$]˙ and backchained until I got Line [L1]˙ to Line [L5]˙. Finally, I applied deduction backwards to [$x \in \mathbf{F} \to x \vee \neg x$]˙ and backchained until I got Line [L6]˙ to Line [L10]˙.

Exercise 14.1.2 Prove the following:

[**Mac lemma L14.1.3** : $x \in \mathbf{B} \to \mathrm{if}(x, 2, 3) \in \mathbf{N}$]

[**Mac lemma L14.1.4** : $m \in \mathbf{N} \to n \in \mathbf{N} \to \mathrm{if}(m < n, m, n) \in \mathbf{N}$]

14.2 Tautologies

The following lemma says that [$x \wedge y \Leftrightarrow y \wedge x$]˙ is true for all truth values [x]˙ and [y]˙:

[**Mac lemma L14.2.1** : $x \in \mathbf{B} \to y \in \mathbf{B} \to x \wedge y \Leftrightarrow y \wedge x$]

The proof is long but not particularly complicated:

[**Mac proof of** L14.2.1:

L01 :	Block $\rhd$	Begin	;
L02 :	Hypothesis $\rhd$	$x \in \mathbf{T}$	;
L03 :	Block $\rhd$	Begin	;
L04 :	Hypothesis $\rhd$	$y \in \mathbf{T}$	;
L05 :	Type$\mathbf{T}\wedge\mathbf{T}\rhd$L2$\rhd$L4 $\rhd$	$(x \wedge y) \in \mathbf{T}$	;
L06 :	Type$\mathbf{T}\wedge\mathbf{T}\rhd$L4$\rhd$L2 $\rhd$	$(y \wedge x) \in \mathbf{T}$	;
L07 :	Type$\mathbf{T}\Leftrightarrow\mathbf{T}\rhd$L5$\rhd$L6 $\rhd$	$(x \wedge y \Leftrightarrow y \wedge x) \in \mathbf{T}$	;
L08 :	Block $\rhd$	End	;
L09 :	Block $\rhd$	Begin	;
L10 :	Hypothesis $\rhd$	$y \in \mathbf{F}$	;
L11 :	Type$\mathbf{T}\wedge\mathbf{F}\rhd$L2$\rhd$L10 $\rhd$	$(x \wedge y) \in \mathbf{F}$	;
L12 :	Type$\mathbf{F}\wedge\mathbf{T}\rhd$L10$\rhd$L2 $\rhd$	$(y \wedge x) \in \mathbf{F}$	;
L13 :	Type$\mathbf{F}\Leftrightarrow\mathbf{F}\rhd$L11$\rhd$L12 $\rhd$	$(x \wedge y \Leftrightarrow y \wedge x) \in \mathbf{T}$	;
L14 :	Block $\rhd$	End	;
L15 :	Cases $\rhd$ L7 $\rhd$ L13 $\rhd$	$y \in \mathbf{B} \rightarrow x \wedge y \Leftrightarrow y \wedge x \in \mathbf{T}$	;
L16 :	Block $\rhd$	End	;
L17 :	Block $\rhd$	Begin	;
L18 :	Hypothesis $\rhd$	$x \in \mathbf{F}$	;
L19 :	Block $\rhd$	Begin	;
L20 :	Hypothesis $\rhd$	$y \in \mathbf{T}$	;
L21 :	Type$\mathbf{F}\wedge\mathbf{T}\rhd$L18$\rhd$L20 $\rhd$	$(x \wedge y) \in \mathbf{F}$	;
L22 :	Type$\mathbf{T}\wedge\mathbf{F}\rhd$L20$\rhd$L18 $\rhd$	$(y \wedge x) \in \mathbf{F}$	;
L23 :	Type$\mathbf{F}\Leftrightarrow\mathbf{F}\rhd$L21$\rhd$L22 $\rhd$	$(x \wedge y \Leftrightarrow y \wedge x) \in \mathbf{T}$	;
L24 :	Block $\rhd$	End	;
L25 :	Block $\rhd$	Begin	;
L26 :	Hypothesis $\rhd$	$y \in \mathbf{F}$	;
L27 :	Type$\mathbf{F}\wedge\mathbf{F}\rhd$L18$\rhd$L26 $\rhd$	$(x \wedge y) \in \mathbf{F}$	;
L28 :	Type$\mathbf{F}\wedge\mathbf{F}\rhd$L26$\rhd$L18 $\rhd$	$(y \wedge x) \in \mathbf{F}$	;
L29 :	Type$\mathbf{F}\Leftrightarrow\mathbf{F}\rhd$L27$\rhd$L28 $\rhd$	$(x \wedge y \Leftrightarrow y \wedge x) \in \mathbf{T}$	;
L30 :	Block $\rhd$	End	;
L31 :	Cases $\rhd$ L23 $\rhd$ L29 $\rhd$	$y \in \mathbf{B} \rightarrow x \wedge y \Leftrightarrow y \wedge x \in \mathbf{T}$	;
L32 :	Block $\rhd$	End	;
L33 :	Cases $\rhd$ L15 $\rhd$ L31 $\rhd$	$x \in \mathbf{B} \rightarrow y \in \mathbf{B} \rightarrow$ $x \wedge y \Leftrightarrow y \wedge x \in \mathbf{T}$	]

Here you have a tabular arrangement of some of the proof lines:

L02:$x \in \mathbf{T}$	L04:$y \in \mathbf{T}$	L05:$(x \wedge y) \in \mathbf{T}$	L06:$(y \wedge x) \in \mathbf{T}$	L07:$(x \wedge y \Leftrightarrow y \wedge x) \in \mathbf{T}$
L02:$x \in \mathbf{T}$	L10:$y \in \mathbf{F}$	L11:$(x \wedge y) \in \mathbf{F}$	L12:$(y \wedge x) \in \mathbf{F}$	L13:$(x \wedge y \Leftrightarrow y \wedge x) \in \mathbf{T}$
L18:$x \in \mathbf{F}$	L20:$y \in \mathbf{T}$	L21:$(x \wedge y) \in \mathbf{F}$	L22:$(y \wedge x) \in \mathbf{F}$	L23:$(x \wedge y \Leftrightarrow y \wedge x) \in \mathbf{T}$
L18:$x \in \mathbf{F}$	L26:$y \in \mathbf{F}$	L27:$(x \wedge y) \in \mathbf{F}$	L28:$(y \wedge x) \in \mathbf{F}$	L29:$(x \wedge y \Leftrightarrow y \wedge x) \in \mathbf{T}$

The idea in the proof is to prove $[\, x \in \mathbf{B} \rightarrow y \in \mathbf{B} \rightarrow x \wedge y \Leftrightarrow y \wedge x \,]^{\cdot}$ by proving $[\, x \wedge y \Leftrightarrow y \wedge x \,]^{\cdot}$ in the four cases where $[\, x \,]^{\cdot}$ and $[\, y \,]^{\cdot}$ are true and false independently of each other. Here you have a more compact arrangement of the table above:

x	y	x ∧ y	y ∧ x	x ∧ y ⇔ y ∧ x
T	**T**	**T**	**T**	**T**
T	**F**	**F**	**F**	**T**
F	**T**	**F**	**F**	**T**
F	**F**	**F**	**F**	**T**

The literature calls such a table a *truth table*. You can prove [$x \in \mathbf{B} \to y \in \mathbf{B} \to x \wedge y \Leftrightarrow y \wedge x$]˙ by constructing a truth table like the one above. The table above is as good a proof as the 33 line derivation proof because it is possible to reconstruct the derivation proof from the truth table in a purely mechanical manner (i.e. you can program a computer to do it, so why bother do it yourself).

Here you have an even more compact arrangement of the table:

x	∧	y	⇔	y	∧	x
T	**T**	**T**	**T**	**T**	**T**	**T**
T	**F**	**F**	**T**	**F**	**F**	**T**
F	**F**	**T**	**T**	**T**	**F**	**F**
F	**F**	**F**	**T**	**F**	**F**	**F**

In the table, Column 1 and 7 indicate the value of [x]˙ and Column 3 and 5 indicate the value of [y]˙. Column 2 indicates the value of [$x \wedge y$]˙, Column 6 indicates the value of [$y \wedge x$]˙, and Column 4 indicates the value of [$x \wedge y \Leftrightarrow y \wedge x$]˙. The lemma holds because all entries in column 4 read [**T**]˙.

A term is a *tautology* if, within the truth table, the column under the principal operator contains nothing but [**T**]˙.

If you need a tautology in a proof, then you can state the tautology in the proof using Tautology as argumentation:

[**Mac rule Tautology** : **if** x is a tautology **then** x]

From a formal point of view, if you use Tautology in a proof, then it is up to the one who checks the correctness of the proof to construct the truth table to see that the given formula actually is a tautology. Whenever you use Tautology in a proof, however, you may construct the truth table as a service to the one who has to check the proof. Whenever you solve exercises related to this course, you are expected to construct a truth table for each use of Tautology.

The use of Tautology certainly can make proofs shorter:

[**Mac proof of** L14.2.1:
L1 : Tautology ▷ $x \in \mathbf{B} \to y \in \mathbf{B} \to (x \wedge y \Leftrightarrow y \wedge x) \in \mathbf{T}$]

Exercise 14.2.2 Prove the following:

[**Mac lemma L14.2.3** : $x \in \mathbf{B} \to x \Rightarrow x \vee x$]

[**Mac lemma L14.2.4** : $x \in \mathbf{B} \to y \in \mathbf{B} \to \neg(x \wedge y) \Leftrightarrow \neg x \vee \neg y$]

---

truth table
tautology

[**Mac lemma L14.2.5** : $x \in \mathbf{B} \to y \in \mathbf{B} \to z \in \mathbf{B} \to x \wedge (y \vee z) \Leftrightarrow x \wedge y \vee x \wedge z$]

[**Mac lemma L14.2.6** : $x \in \mathbf{B} \to y \in \mathbf{B} \to (x \Rightarrow y) \Leftrightarrow \neg x \vee y$]

[**Mac lemma L14.2.7** : $x \in \mathbf{B} \to y \in \mathbf{B} \to (x \Leftrightarrow y) \Leftrightarrow (x \Rightarrow y) \wedge (y \Rightarrow x)$]

L14.2.4 and the related rule [$x \in \mathbf{B} \to y \in \mathbf{B} \to \neg(x \vee y) \Leftrightarrow \neg x \wedge \neg y$] were discovered by *de Morgan*.

L14.2.6 was discovered by an Albanian railroad engineer (remember: a train crash is more dramatic than a computer crash, so railroad administrations tend to use more logic than computer business in verifying the correctness of systems).

14.3 Answers

Answer 14.1.2

[**Mac proof of** L14.1.3:

L01 :	Block ▷	Begin	;
L02 :	Hypothesis ▷	$x \in \mathbf{T}$	;
L03 :	IfT ▷ L2 ▷	$\mathrm{if}(x, 2, 3) \equiv 2$	;
L04 :	TypeNumeralInN ▷	$2 \in \mathbf{N}$	;
L05 :	Reverse' ▷ L3 ▷ L4 ▷	$\mathrm{if}(x, 2, 3) \in \mathbf{N}$	;
L06 :	Block ▷	End	;
L07 :	Block ▷	Begin	;
L08 :	Hypothesis ▷	$x \in \mathbf{F}$	;
L09 :	IfF ▷ L8 ▷	$\mathrm{if}(x, 2, 3) \equiv 3$	;
L10 :	TypeNumeralInN ▷	$3 \in \mathbf{N}$	;
L11 :	Reverse' ▷ L9 ▷ L10 ▷	$\mathrm{if}(x, 2, 3) \in \mathbf{N}$	;
L12 :	Block ▷	End	;
L13 :	Cases ▷ L5 ▷ L11 ▷	$x \in \mathbf{B} \to \mathrm{if}(x, 2, 3) \in \mathbf{N}$	]

[**Mac proof of** L14.1.4:

L01 :	Hypothesis ▷	$m \in \mathbf{N}$	;
L02 :	Hypothesis ▷	$n \in \mathbf{N}$	;
L03 :	Type**N**<**N** ▷ L1 ▷ L2 ▷	$(m < n) \in \mathbf{B}$	;
L04 :	Block ▷	Begin	;
L05 :	Hypothesis ▷	$(m < n) \in \mathbf{T}$	;
L06 :	IfT ▷ L5 ▷	$\mathrm{if}(m < n, m, n) \equiv m$	;
L07 :	Reverse' ▷ L1 ▷ L6 ▷	$\mathrm{if}(m < n, m, n) \in \mathbf{N}$	;
L08 :	Block ▷	End	;
L09 :	Block ▷	Begin	;
L10 :	Hypothesis ▷	$(m < n) \in \mathbf{F}$	;
L11 :	IfF ▷ L10 ▷	$\mathrm{if}(m < n, m, n) \equiv n$	;
L12 :	Reverse' ▷ L11 ▷ L2 ▷	$\mathrm{if}(m < n, m, n) \in \mathbf{N}$	;
L13 :	Block ▷	End	;
L14 :	Cases ▷ L7 ▷ L12 ▷	$(m < n) \in \mathbf{B} \to \mathrm{if}(m < n, m, n) \in \mathbf{N}$	;
L15 :	L14 ⊵ L3 ▷	$\mathrm{if}(m < n, m, n) \in \mathbf{N}$	]

de Morgan

Answer 14.2.2

[**Mac proof of** L14.2.3:
L1 : Tautology ▷ $x \in \mathbf{B} \to x \Rightarrow x \vee x$]

x	⇒	x	∨	x
T	**T**	**T**	**T**	**T**
F	**T**	**F**	**F**	**F**

[**Mac proof of** L14.2.4:
L1 : Tautology ▷ $x \in \mathbf{B} \to y \in \mathbf{B} \to \neg(x \wedge y) \Leftrightarrow \neg x \vee \neg y$]

¬	(x	∧	y)	⇔	¬	x	∨	¬	y
F	**T**	**T**	**T**	**T**	**F**	**T**	**F**	**F**	**T**
T	**T**	**F**	**F**	**T**	**F**	**T**	**T**	**T**	**F**
T	**F**	**F**	**T**	**T**	**T**	**F**	**T**	**F**	**T**
T	**F**	**F**	**F**	**T**	**T**	**F**	**T**	**T**	**F**

[**Mac proof of** L14.2.5:
L1 : Tautology ▷ $x\in\mathbf{B}\to y\in\mathbf{B}\to z\in\mathbf{B} \to x \wedge (y \vee z) \Leftrightarrow x \wedge y \vee x \wedge z$]

x	∧	(y	∨	z)	⇔	x	∧	y	∨	x	∧	z
T	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**
T	**T**	**T**	**T**	**F**	**T**	**T**	**T**	**T**	**T**	**T**	**F**	**F**
T	**T**	**F**	**T**	**T**	**T**	**T**	**F**	**F**	**T**	**T**	**T**	**T**
T	**F**	**F**	**F**	**F**	**T**	**T**	**F**	**F**	**F**	**T**	**F**	**F**
F	**F**	**T**	**T**	**T**	**T**	**F**	**F**	**T**	**F**	**F**	**F**	**T**
F	**F**	**T**	**T**	**F**	**T**	**F**	**F**	**T**	**F**	**F**	**F**	**F**
F	**F**	**F**	**T**	**T**	**T**	**F**	**F**	**F**	**F**	**F**	**F**	**T**
F	**F**	**F**	**F**	**F**	**T**	**F**	**F**	**F**	**F**	**F**	**F**	**F**

[**Mac proof of** L14.2.6:
L1 : Tautology ▷ $x \in \mathbf{B} \to y \in \mathbf{B} \to (x \Rightarrow y) \Leftrightarrow \neg x \vee y$]

(x	⇒	y)	⇔	¬	x	∨	y
T	**T**	**T**	**T**	**F**	**T**	**T**	**T**
T	**F**	**F**	**T**	**F**	**T**	**F**	**F**
F	**T**	**T**	**T**	**T**	**F**	**T**	**T**
F	**T**	**F**	**T**	**T**	**F**	**T**	**F**

[**Mac proof of** L14.2.7:
L1 : Tautology ▷ $x \in \mathbf{B} \to y \in \mathbf{B} \to (x \Leftrightarrow y) \Leftrightarrow (x \Rightarrow y) \wedge (y \Rightarrow x)$]

(x	⇔	y)	⇔	(x	⇒	y)	∧	(y	⇒	x)
T	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**	**T**
T	**F**	**F**	**T**	**T**	**F**	**F**	**F**	**F**	**T**	**T**
F	**F**	**T**	**T**	**F**	**T**	**T**	**F**	**T**	**F**	**F**
F	**T**	**F**	**T**	**F**	**T**	**F**	**T**	**F**	**T**	**F**

Chapter 15

Natural numbers

15.1 The Peano axioms

Peano (Italian mathematician, 1858–1932) and Dedekind (German mathematician, 1831–1916) were the first to publish axioms for the natural numbers. They did so independently, Dedekind in 1888 and Peano in 1889. Peano did a lot to promote the axiomatic method, and that explains why the axioms are now known as the Peano axioms even though Dedekind was first. Peano also worked on artificial languages like Volapük and Latino Sine Flexione (Latin without grammar). Peano was the one who invented the sign $[\in]^\circ$ to stand for membership.

At the time of Peano and Dedekind, 0 was not regarded as a natural number. Later, it has turned out to simplify matters to count 0 among the natural numbers. I have chosen to follow the trend and count 0 among the natural numbers for the following reason: I constantly need to refer to the set of integers $[\mathsf{x}]^\cdot$ for which $[\mathsf{x} > 0]^\cdot$ as well as the set of integers for which $[\mathsf{x} \geq 0]^\cdot$. Now, the term 'positive integer' covers the first kind of integers. If I did not count 0 among the natural numbers then 'natural number' and 'positive integer' would mean the same which is a waste of a good name.

Before Dedekind and Peano, it was a saying that "the natural numbers are the creation of God, all other is the creation of man". What this really meant was that mathematicians spent time on defining concepts like negative numbers, fractions, decimal fraction, real numbers, complex numbers, quarternions, and so on, and they formulated and proved lemmas about these "advanced" numbers. But mathematicians did not formulate and prove lemmas about natural numbers simply because they had no rules for proving such lemmas.

Peano formulated five rules about natural numbers. In my formulation they read:

1. Zero is a natural number. (Actually, Peano did not regard zero as a natural number, so his claim was that One was a natural number).

2. For any natural number [x] there is a natural number one greater than [x]. This is known as the *successor* of [x] and I denote it [x^+].
3. If two natural numbers have the same successor, then they are equal.
4. For any natural number [x] different from zero, there is a natural number whose successor equals [x]. Actually, according to (3) above, there is only one such number. I call the number the *predecessor* of [x] and denote it [x^-].
5. The principle of *induction* (which I state later).

Here you have more formal version of the first four rules (where I have taken the liberty to split the fourth Peano rule into two):

[**Mac rule PeanoA** : $0 \in \mathbf{N}$]

[**Mac rule PeanoB** : $x \in \mathbf{N} \vdash x^+ \in \mathbf{N}$]

[**Mac rule PeanoC** : $x \in \mathbf{N} \vdash y \in \mathbf{N} \vdash x^+ = y^+ \vdash x = y$]

[**Mac rule PeanoD** : $x \in \mathbf{N} \vdash (x = 0) \in \mathbf{F} \vdash x^- \in \mathbf{N}$]

[**Mac rule PeanoE** : $x \in \mathbf{N} \vdash (x = 0) \in \mathbf{F} \vdash x^{-+} = x$]

You will not need these rules for the exercises in this book, but you will need a rule which is just a reformulation of PeanoD:

[**Mac rule PeanoD'** : $x \in \mathbf{N} \vdash (x = 0) \in \mathbf{F} \vdash x - 1 \in \mathbf{N}$]

15.2 Induction

The fifth and most complicated rule among the Peano axioms is an *induction principle*, now known as *Peano induction*. In my formulation, Peano induction reads:

[**Mac rule Induction** :
$n \in \mathbf{N} \to (n = 0) \in \mathbf{T} \to \mathcal{A} \vdash$
$n \in \mathbf{N} \to (n = 0) \in \mathbf{F} \to \langle \mathcal{A} \mid n{:=}n - 1 \rangle \to \mathcal{A} \vdash$
$n \in \mathbf{N} \to \mathcal{A}$]

To master induction, you need to do two things:

---

successor
[x^+] x successor
predecessor
[x^-] x predecessor
induction
induction principle
Peano induction

(a) You must learn to use the proof technique in practice.

(b) You must understand why Induction holds.

If you have difficulty with (b) you may find it useful to concentrate on (a) first.

Before you can to anything with induction, you must master implication, deduction, and substitution, so make sure you are really done with those topics.

You merely need (a) above to use Peano induction to prove theorems. You need (b) whenever you encounter new situations where you need other kinds of induction. Once you really understand induction, you can easily formulate your own induction rules.

15.3 Using induction

I will now prove a lemma using induction:

[**Mac lemma L15.3.1** : $n \in \mathbf{N} \to n! \in \mathbf{N}$]

The lemma says that [n!]˙ is a natural number whenever [n]˙ is a natural number. In particular, the lemma says that a computer can compute [n!]˙ in finite time whenever [n]˙ is a natural number. This is because the lemma says, among other, that [n!]˙ differs from [⊥]˙. Since the definition of [n!]˙ merely involves computable constructs and since [n!]˙ differs from [⊥]˙, a computer can compute [n!]˙ in finite time.

I will prove [$n \in \mathbf{N} \to n! \in \mathbf{N}$]˙ by backchaining. The lemma has the form of an implication, so the most obvious thing to do is to apply deduction backwards:

[**Mac proof of** L15.3.1:

L1 :	Block ▷	Begin	;
L2 :	Hypothesis ▷	$n \in \mathbf{N}$	;
L3 :	⋯ ▷	$n! \in \mathbf{N}$	;
L4 :	Block ▷	End	;
L5 :	Repetition ▷ L3 ▷	$n \in \mathbf{N} \to n! \in \mathbf{N}$	]˙

This approach leads nowhere, however. Now that the lemma has survived an attack by deduction, I increase the fire by a backward application of Induction:

[**Mac proof of** L15.3.1:

x :	⋯ ▷	$n \in \mathbf{N} \to (n = 0) \in \mathbf{T} \to n! \in \mathbf{N}$	;
y :	⋯ ▷	$n \in \mathbf{N} \to (n = 0) \in \mathbf{F} \to$ $(n - 1)! \in \mathbf{N} \to n! \in \mathbf{N}$	;
z :	Induction ▷ x ▷ y ▷	$n \in \mathbf{N} \to n! \in \mathbf{N}$	]˙

The premisses of induction have the form of implications, so I try to apply deduction backwards.

[**Mac proof of** L15.3.1:

	Block ▷	Begin	;
	Hypothesis ▷	$n \in \mathbf{N}$	;
	Hypothesis ▷	$(n = 0) \in \mathbf{T}$	;
x :	⋯ ▷	$n! \in \mathbf{N}$	;
	Block ▷	End	;
	Block ▷	Begin	;
	Hypothesis ▷	$n \in \mathbf{N}$	;
	Hypothesis ▷	$(n = 0) \in \mathbf{F}$	;
	Hypothesis ▷	$(n - 1)! \in \mathbf{N}$	;
y :	⋯ ▷	$n! \in \mathbf{N}$	;
	Block ▷	End	;
z :	Induction ▷ x ▷ y ▷	$n \in \mathbf{N} \rightarrow n! \in \mathbf{N}$	]

I attack the two blocks separately. The first block reads:

[**Mac proof of** L15.3.1:

L1 :	Block ▷	Begin	;
L2 :	Hypothesis ▷	$n \in \mathbf{N}$	;
L3 :	Hypothesis ▷	$(n = 0) \in \mathbf{T}$	;
x :	⋯ ▷	$n! \in \mathbf{N}$	;
	Block ▷	End	]

There is nothing to backchain here, so it is time to throw in some new information in the proof. The thing to do is to compute the value of [$n!$] using the definition of [$n!$]:

[**Mac proof of** L15.3.1:

L01 :	Block ▷	Begin	;
L02 :	Hypothesis ▷	$n \in \mathbf{N}$	;
L03 :	Hypothesis ▷	$(n = 0) \in \mathbf{T}$	;
L04 :	Block ▷	Begin	;
L05 :	Algebra ▷	$n!$	;
L06 :	Definition ▷	$\mathrm{if}(n = 0, 1, n \cdot (n - 1)!)$	;
L07 :	IfT ▷ L3 ▷	1	;
L08 :	Block ▷	End	;
x :	⋯ ▷	$n! \in \mathbf{N}$	;
	Block ▷	End	]

At this point we have [$n! \equiv 1$] (Line [L7]) and have to prove [$n! \in \mathbf{N}$]. The final move in this part of the proof is to insert a line that says [$1 \in \mathbf{N}$]:

[**Mac proof of** L15.3.1:

L01 :	Block ▷	Begin	;
L02 :	Hypothesis ▷	$\quad n \in \mathbf{N}$	;
L03 :	Hypothesis ▷	$\quad (n = 0) \in \mathbf{T}$	;
L04 :	Block ▷	$\quad$ Begin	;
L05 :	Algebra ▷	$\qquad n!$	;
L06 :	Definition ▷	$\qquad \text{if}(n = 0, 1, n \cdot (n-1)!)$	;
L07 :	IfT ▷ L3 ▷	$\qquad 1$	;
L08 :	Block ▷	$\quad$ End	;
L09 :	TypeNumeralInN ▷	$\quad 1 \in \mathbf{N}$	;
L10 :	Reverse' ▷ L7 ▷ L9 ▷	$\quad n! \in \mathbf{N}$	;
L11 :	Block ▷	End	]

Next, I attack the second block:

[**Mac proof of** L15.3.1:

L12 :	Block ▷	Begin	;
L13 :	Hypothesis ▷	$\quad n \in \mathbf{N}$	;
L14 :	Hypothesis ▷	$\quad (n = 0) \in \mathbf{F}$	;
L15 :	Hypothesis ▷	$\quad (n-1)! \in \mathbf{N}$	;
y :	$\cdots$ ▷	$\quad n! \in \mathbf{N}$	;
	Block ▷	End	]

Now I am in the same situation as in the first block: Nothing seems to apply. Again, the thing to do is to compute the value of [$n!$] (to the extent computation is possible) using the definition of [$n!$]:

[**Mac proof of** L15.3.1:

L12 :	Block ▷	Begin	;
L13 :	Hypothesis ▷	$\quad n \in \mathbf{N}$	;
L14 :	Hypothesis ▷	$\quad (n = 0) \in \mathbf{F}$	;
L15 :	Hypothesis ▷	$\quad (n-1)! \in \mathbf{N}$	;
L16 :	Block ▷	$\quad$ Begin	;
L17 :	Algebra ▷	$\qquad n!$	;
L18 :	Definition ▷	$\qquad \text{if}(n = 0, 1, n \cdot (n-1)!)$	;
L19 :	IfF ▷ L14 ▷	$\qquad n \cdot (n-1)!$	;
L20 :	Block ▷	$\quad$ End	;
z :	$\cdots$ ▷	$\quad n \cdot (n-1)! \in \mathbf{N}$	;
y :	Reverse' ▷ L19 ▷ z ▷	$\quad n! \in \mathbf{N}$	;
	Block ▷	End	]

In Line [z] of the proof above I took the liberty to state that [$n \cdot (n-1)! \in \mathbf{N}$] is true. I prove that by backchaining. Backward application of [Type$\mathbf{N}\cdot\mathbf{N}$] on [$n \cdot (n-1)! \in \mathbf{N}$] gives

$$[\, n \in \mathbf{N}; (n-1)! \in \mathbf{N} \,].$$

Both of these happen to be in the proof already, so the second block reads:

[**Mac proof of** L15.3.1:

L12 :	Block $\rhd$	Begin	;
L13 :	Hypothesis $\rhd$	$\mathsf{n} \in \mathbf{N}$	;
L14 :	Hypothesis $\rhd$	$(\mathsf{n} = 0) \in \mathbf{F}$	;
L15 :	Hypothesis $\rhd$	$(\mathsf{n} - 1)! \in \mathbf{N}$	;
L16 :	Block $\rhd$	Begin	;
L17 :	Algebra $\rhd$	$\mathsf{n}!$	;
L18 :	Definition $\rhd$	$\mathrm{if}(\mathsf{n}{=}0, 1, \mathsf{n}{\cdot}(\mathsf{n}{-}1)!)$	;
L19 :	IfF $\rhd$ L14 $\rhd$	$\mathsf{n} \cdot (\mathsf{n} - 1)!$	;
L20 :	Block $\rhd$	End	;
L21 :	Type$\mathbf{N}{\cdot}\mathbf{N}$$\rhd$L13$\rhd$L15 $\rhd$	$\mathsf{n} \cdot (\mathsf{n} - 1)! \in \mathbf{N}$	;
L22 :	Reverse' $\rhd$ L19 $\rhd$ L21 $\rhd$	$\mathsf{n}! \in \mathbf{N}$	;
L23 :	Block $\rhd$	End	]

Putting everything together, I have the proof:

[**Mac proof of** L15.3.1:

L01 :	Block $\rhd$	Begin	;
L02 :	Hypothesis $\rhd$	$\mathsf{n} \in \mathbf{N}$	;
L03 :	Hypothesis $\rhd$	$(\mathsf{n} = 0) \in \mathbf{T}$	;
L04 :	Block $\rhd$	Begin	;
L05 :	Algebra $\rhd$	$\mathsf{n}!$	;
L06 :	Definition $\rhd$	$\mathrm{if}(\mathsf{n} = 0, 1, \mathsf{n} \cdot (\mathsf{n} - 1)!)$	;
L07 :	IfT $\rhd$ L3 $\rhd$	1	;
L08 :	Block $\rhd$	End	;
L09 :	TypeNumeralInN $\rhd$	$1 \in \mathbf{N}$	;
L10 :	Reverse' $\rhd$ L7 $\rhd$ L9 $\rhd$	$\mathsf{n}! \in \mathbf{N}$	;
L11 :	Block $\rhd$	End	;
L12 :	Block $\rhd$	Begin	;
L13 :	Hypothesis $\rhd$	$\mathsf{n} \in \mathbf{N}$	;
L14 :	Hypothesis $\rhd$	$(\mathsf{n} = 0) \in \mathbf{F}$	;
L15 :	Hypothesis $\rhd$	$(\mathsf{n} - 1)! \in \mathbf{N}$	;
L16 :	Block $\rhd$	Begin	;
L17 :	Algebra $\rhd$	$\mathsf{n}!$	;
L18 :	Definition $\rhd$	$\mathrm{if}(\mathsf{n}{=}0, 1, \mathsf{n}{\cdot}(\mathsf{n}{-}1)!)$	;
L19 :	IfF $\rhd$ L14 $\rhd$	$\mathsf{n} \cdot (\mathsf{n} - 1)!$	;
L20 :	Block $\rhd$	End	;
L21 :	Type$\mathbf{N}{\cdot}\mathbf{N}$$\rhd$L13$\rhd$L15 $\rhd$	$\mathsf{n} \cdot (\mathsf{n} - 1)! \in \mathbf{N}$	;
L22 :	Reverse' $\rhd$ L19 $\rhd$ L21 $\rhd$	$\mathsf{n}! \in \mathbf{N}$	;
L23 :	Block $\rhd$	End	;
L24 :	Induction $\rhd$ L10 $\rhd$ L22 $\rhd$	$\mathsf{n} \in \mathbf{N} \rightarrow \mathsf{n}! \in \mathbf{N}$	]

15.4 Analysis of the induction proof

The proof of L15.3.1 merely contains one application of induction. The rest of the proof uses other proof techniques. This is very typical for induction proofs: Induction is the key to the proof, but all the hard work is in proving the premisses of induction.

The proof of L15.3.1 proves

$$[\, \mathsf{n} \in \mathbf{N} \to \mathsf{n}! \in \mathbf{N} \,]^{\cdot}$$

by induction from

$$[\, \mathsf{n} \in \mathbf{N} \to (\mathsf{n} = 0) \in \mathbf{T} \to \mathsf{n}! \in \mathbf{N} \,]^{\cdot}$$

and

$$[\, \mathsf{n} \in \mathbf{N} \to (\mathsf{n} = 0) \in \mathbf{F} \to \langle \mathsf{n}! \in \mathbf{N} \mid \mathsf{n}{:=}\mathsf{n} - 1 \rangle \to \mathsf{n}! \in \mathbf{N} \,]^{\cdot}.$$

The two premisses are equivalent to the following lemmas:

[**Mac lemma L15.4.1** : $\mathsf{n} \in \mathbf{N} \vdash (\mathsf{n} = 0) \in \mathbf{T} \vdash \mathsf{n}! \in \mathbf{N}$]

[**Mac proof of** L15.4.1**:**

L02 :	Premise ▷	$\mathsf{n} \in \mathbf{N}$	;
L03 :	Premise ▷	$(\mathsf{n} = 0) \in \mathbf{T}$	;
L04 :	Block ▷	Begin	;
L05 :	Algebra ▷	$\mathsf{n}!$	;
L06 :	Definition ▷	$\mathrm{if}(\mathsf{n} = 0, 1, \mathsf{n} \cdot (\mathsf{n} - 1)!)$	;
L07 :	IfT ▷ L3 ▷	1	;
L08 :	Block ▷	End	;
L09 :	TypeNumeralInN ▷	$1 \in \mathbf{N}$	;
L10 :	Reverse' ▷ L7 ▷ L9 ▷	$\mathsf{n}! \in \mathbf{N}$	]

[**Mac lemma L15.4.2** : $\mathsf{n} \in \mathbf{N} \vdash (\mathsf{n} = 0) \in \mathbf{F} \vdash \langle \mathsf{n}! \in \mathbf{N} \mid \mathsf{n}{:=}\mathsf{n} - 1 \rangle \vdash \mathsf{n}! \in \mathbf{N}$]

[**Mac proof of** L15.4.2**:**

L13 :	Premise ▷	$\mathsf{n} \in \mathbf{N}$	;
L14 :	Premise ▷	$(\mathsf{n} = 0) \in \mathbf{F}$	;
L15 :	Premise ▷	$(\mathsf{n} - 1)! \in \mathbf{N}$	;
L16 :	Block ▷	Begin	;
L17 :	Algebra ▷	$\mathsf{n}!$	;
L18 :	Definition ▷	$\mathrm{if}(\mathsf{n} = 0, 1, \mathsf{n} \cdot (\mathsf{n} - 1)!)$	;
L19 :	IfF ▷ L14 ▷	$\mathsf{n} \cdot (\mathsf{n} - 1)!$	;
L20 :	Block ▷	End	;
L21 :	Type**N**·**N**▷L13▷L15 ▷	$\mathsf{n} \cdot (\mathsf{n} - 1)! \in \mathbf{N}$	;
L22 :	Reverse' ▷ L19 ▷ L21 ▷	$\mathsf{n}! \in \mathbf{N}$	]

It is interesting to see how much it is possible to prove from these two lemmas without induction and without using the definition of $[\, \mathsf{n}! \,]^{\cdot}$:

[**Mac lemma L15.4.3** : $0! \in \mathbf{N}$]

[**Mac proof of** L15.4.3**:**

L1 :	TypeNumeralInN $\rhd$	$0 \in \mathbf{N}$	;
L2 :	TypeTInT $\rhd$	$\mathsf{T} \in \mathbf{T}$	;
L3 :	Reverse' $\rhd$ $0 = 0 \equiv \mathsf{T}$ $\rhd$ L2 $\rhd$	$(0 = 0) \in \mathbf{T}$	;
L4 :	L15.4.1 $\rhd$ L1 $\rhd$ L3 $\rhd$	$0! \in \mathbf{N}$	]

[**Mac lemma L15.4.4** : $1! \in \mathbf{N}$]

[**Mac proof of** L15.4.4**:**

L1 :	TypeNumeralInN $\rhd$	$1 \in \mathbf{N}$	;
L2 :	TypeFInF $\rhd$	$\mathsf{F} \in \mathbf{F}$	;
L3 :	Reverse' $\rhd$ $1 = 0 \equiv \mathsf{F}$ $\rhd$ L2 $\rhd$	$(1 = 0) \in \mathbf{F}$	;
L4 :	L15.4.3 $\rhd$	$0! \in \mathbf{N}$	;
L5 :	Reverse' $\rhd$ $(1 - 1) \equiv 0$ $\rhd$ L4 $\rhd$	$(1 - 1)! \in \mathbf{N}$	;
L6 :	L15.4.2 $\rhd$ L1 $\rhd$ L3 $\rhd$ L5 $\rhd$	$1! \in \mathbf{N}$	]

[**Mac lemma L15.4.5** : $2! \in \mathbf{N}$]

[**Mac proof of** L15.4.5**:**

L1 :	TypeNumeralInN $\rhd$	$2 \in \mathbf{N}$	;
L2 :	TypeFInF $\rhd$	$\mathsf{F} \in \mathbf{F}$	;
L3 :	Reverse' $\rhd$ $2 = 0 \equiv \mathsf{F}$ $\rhd$ L2 $\rhd$	$(2 = 0) \in \mathbf{F}$	;
L4 :	L15.4.4 $\rhd$	$1! \in \mathbf{N}$	;
L5 :	Reverse' $\rhd$ $(2 - 1) \equiv 1$ $\rhd$ L4 $\rhd$	$(2 - 1)! \in \mathbf{N}$	;
L6 :	L15.4.2 $\rhd$ L1 $\rhd$ L3 $\rhd$ L5 $\rhd$	$2! \in \mathbf{N}$	]

[**Mac lemma L15.4.6** : $3! \in \mathbf{N}$]

[**Mac proof of** L15.4.6**:**

L1 :	TypeNumeralInN $\rhd$	$3 \in \mathbf{N}$	;
L2 :	TypeFInF $\rhd$	$\mathsf{F} \in \mathbf{F}$	;
L3 :	Reverse' $\rhd$ $3 = 0 \equiv \mathsf{F}$ $\rhd$ L2 $\rhd$	$(3 = 0) \in \mathbf{F}$	;
L4 :	L15.4.5 $\rhd$	$2! \in \mathbf{N}$	;
L5 :	Reverse' $\rhd$ $(3 - 1) \equiv 2$ $\rhd$ L4 $\rhd$	$(3 - 1)! \in \mathbf{N}$	;
L6 :	L15.4.2 $\rhd$ L1 $\rhd$ L3 $\rhd$ L5 $\rhd$	$3! \in \mathbf{N}$	]

Have you got the pattern? If not, prove [$117! \in \mathbf{N}$]˙ from L15.4.1 and L15.4.2 without using induction and without using the definition of [$\mathsf{n}!$]˙.

Now that you have proved [$117! \in \mathbf{N}$]˙ I assume you agree that it is possible to prove [$1000000! \in \mathbf{N}$]˙ and [$1000000000! \in \mathbf{N}$]˙ this way. Or, said another way, if [n]˙ is a numeral that denotes a natural number, then anyone can prove [$\mathsf{n}! \in \mathbf{N}$]˙ using L15.4.1 and L15.4.2.

Without induction you can prove infinitely many lemmas one by one, namely the lemmas [$0! \in \mathbf{N}$]˙, [$1! \in \mathbf{N}$]˙, [$2! \in \mathbf{N}$]˙, [$3! \in \mathbf{N}$]˙, and so on. With induction you can kill infinitely many birds with one stone in that you can prove [$\mathsf{n} \in \mathbf{N} \to \mathsf{n}! \in \mathbf{N}$]˙.

15.5 Understanding induction

As I mentioned in Section 15.2, Peano induction reads:

[**Mac rule Induction** :
$\mathsf{n} \in \mathbf{N} \to (\mathsf{n} = 0) \in \mathbf{T} \to \mathcal{A} \vdash$
$\mathsf{n} \in \mathbf{N} \to (\mathsf{n} = 0) \in \mathbf{F} \to \langle \mathcal{A} \mid \mathsf{n}{:=}\mathsf{n} - 1 \rangle \to \mathcal{A} \vdash$
$\mathsf{n} \in \mathbf{N} \to \mathcal{A}$]

As I also mentioned in Section 15.2, if you really want to understand this rule, you should first make sure that you understand implication, deduction, and substitution.

After that you should go back and look at how I represent numbers. I represent numbers such that each number has many representations. As an example, zero has many representations. The premise [$\mathsf{n} \in \mathbf{N} \to (\mathsf{n} = 0) \in \mathbf{T} \to \mathcal{A}$]˙ says that [$\mathcal{A}$]˙ is true for all [n]˙ that represent zero. The conclusion [$\mathsf{n} \in \mathbf{N} \to \mathcal{A}$]˙ says that [$\mathcal{A}$]˙ is true for all [n]˙ that represent natural numbers.

To demonstrate you understand what impact the representation has, give a counterproof to the following antilemma (c.f. the exercise below, see Chapter 16 for a general discussion of antilemmas):

[**Mac antilemma L15.5.1** : $\langle \mathcal{A} \mid \mathsf{n}{:=}0 \rangle \vdash \mathsf{n} \in \mathbf{N} \to \mathcal{A} \to \langle \mathcal{A} \mid \mathsf{n}{:=}\mathsf{n} + 1 \rangle \vdash \mathsf{n} \in \mathbf{N} \to \mathcal{A}$]

The rule above fails because each natural number has more than one representation in my framework. If each natural number had exactly one representation, then the rule above would hold. Therefore you may find the rule above in other mathematics books that use other frameworks.

After disproving L15.5.1 and after carefully reading Section 15.4 you may be so lucky that it is clear to you that Induction holds. If not, you may attack Induction by double negation: Try to disprove Induction. In other words, try to find a term [$\mathcal{A}$]˙ and a term [m]˙ such that

> [$\mathsf{n} \in \mathbf{N} \to (\mathsf{n} = 0) \in \mathbf{T} \to \mathcal{A}$]˙ holds, and
>
> [$\mathsf{n} \in \mathbf{N} \to (\mathsf{n} = 0) \in \mathbf{F} \to \langle \mathcal{A} \mid \mathsf{n}{:=}\mathsf{n} - 1 \rangle \to \mathcal{A}$]˙ holds, and
>
> [$\mathsf{m} \in \mathbf{N}$]˙ holds, and
>
> [$\langle \mathcal{A} \mid \mathsf{n}{:=}\mathsf{m} \rangle$]˙ fails.

If you fail to disprove Induction, then you may end up understanding why it holds.

In the end, nobody but you can make you understand anything. I have done my best to give you some clues. The rest is up to you.

Exercise 15.5.2 Disprove [L15.5.1]˙, i.e. find an [$\mathcal{A}$]˙ and an [m]˙ such that [$\langle \mathcal{A} \mid \mathsf{n}{:=}0 \rangle \equiv \mathsf{T}$]˙ holds and [$\mathsf{n} \in \mathbf{N} \to \mathcal{A} \to \langle \mathcal{A} \mid \mathsf{n}{:=}\mathsf{n} + 1 \rangle$]˙ holds, and [$\mathsf{m} \in \mathbf{N}$]˙ holds, and [$\langle \mathcal{A} \mid \mathsf{n}{:=}\mathsf{m} \rangle$]˙ fails.

15.6 Answers

Answer 15.5.2 I define

$$\left[\ \mathrm{normalise}(\mathsf{x}) \doteq \mathsf{x} = 0 \left\{ \begin{array}{l} 0 \\ \mathrm{normalise}(\mathsf{x}-1)+1 \end{array} \right.\ \right]^{\cdot}.$$

If $[\ \mathsf{x}\ ]^{\cdot}$ is a weak representation of a natural number, then $[\ \mathrm{normalise}(\mathsf{x})\ ]^{\cdot}$ is the strong representation of the same number. Here you have an example:

$$[\ \mathrm{normalise}(\langle\langle \mathsf{N}, \underline{\mathsf{E}}, \langle \oplus, \underline{0}\rangle^{\cdot}, \oplus, \underline{1}\rangle^{\cdot}\rangle^{\cdot}) \equiv \langle\langle \mathsf{N}, \underline{\mathsf{E}}, \langle \oplus, \underline{0}\rangle^{\cdot}, \oplus, \underline{0}\rangle^{\cdot}\rangle^{\cdot}\ ]^{\cdot}.$$

The relation $[\ \mathsf{x} \mathrel{\ddot{\tilde{=}}} \mathsf{y}\ ]^{\cdot}$ compares structures that are built up from $[\ \mathsf{N}\ ]^{\cdot}$ and $[\ \mathsf{x} \mathrel{.\!\!\because} \mathsf{y}\ ]^{\cdot}$, so you have e.g.:

$$[\ \langle\langle \mathsf{N}, \underline{\mathsf{E}}, \langle \oplus, \underline{0}\rangle^{\cdot}, \oplus, \underline{1}\rangle^{\cdot}\rangle^{\cdot}) \mathrel{\ddot{\tilde{=}}} \langle\langle \mathsf{N}, \underline{\mathsf{E}}, \langle \oplus, \underline{0}\rangle^{\cdot}, \oplus, \underline{0}\rangle^{\cdot}\rangle^{\cdot} \equiv \mathsf{F}\ ]^{\cdot}.$$

You have $[\ \mathrm{normalise}(\mathsf{n}) \equiv \mathsf{n}\ ]^{\cdot}$ whenever $[\ \mathsf{n}\ ]^{\cdot}$ is a strong representation of a natural number. Therefore you have

$$[\ \mathrm{normalise}(\mathsf{n}) \mathrel{\ddot{\tilde{=}}} \mathsf{n}\ ]^{\cdot}$$

for all strong representations of natural numbers. Now let $[\ \mathcal{A}\ ]^{\cdot}$ be

$$[\ \mathrm{normalise}(\mathsf{n}) \mathrel{\ddot{\tilde{=}}} \mathsf{n}\ ]^{\cdot}$$

and let $[\ \mathsf{m}\ ]^{\cdot}$ be

$$[\ \langle\langle \mathsf{N}, \underline{\mathsf{E}}, \langle \oplus, \underline{0}\rangle^{\cdot}, \oplus, \underline{1}\rangle^{\cdot}\rangle^{\cdot}\ ]^{\cdot}.$$

You have:

$[\ \langle \mathrm{normalise}(\mathsf{n}) \mathrel{\ddot{\tilde{=}}} \mathsf{n} \mid \mathsf{n}{:=}0\rangle \equiv \mathsf{T}\ ]^{\cdot}$ holds

because $[\ 0\ ]^{\cdot}$ strongly represents a natural number.

$[\ \mathsf{n} \in \mathbf{N} \to \mathrm{normalise}(\mathsf{n}) \mathrel{\ddot{\tilde{=}}} \mathsf{n} \to \langle \mathrm{normalise}(\mathsf{n}) \mathrel{\ddot{\tilde{=}}} \mathsf{n} \mid \mathsf{n}{:=}\mathsf{n}+1\rangle\ ]^{\cdot}$ holds

because $[\ \mathsf{n}+1\ ]^{\cdot}$ strongly represents a natural number whenever $[\ \mathsf{n} \in \mathbf{N}\ ]^{\cdot}$.

$[\ \mathsf{m} \in \mathbf{N}\ ]^{\cdot}$ holds

because $[\ \mathsf{m}\ ]^{\cdot}$ is classical and weakly represents a natural number.

$[\ \langle \mathrm{normalise}(\mathsf{n}) \mathrel{\ddot{\tilde{=}}} \mathsf{n} \mid \mathsf{n}{:=}\mathsf{m}\rangle\ ]^{\cdot}$ fails

because $[\ \mathrm{normalise}(\mathsf{m})\ ]^{\cdot}$ equals the strong representation of zero which is

$$[\ \langle\langle \mathsf{N}, \underline{\mathsf{E}}, \langle \oplus, \underline{0}\rangle^{\cdot}, \oplus, \underline{0}\rangle^{\cdot}\rangle^{\cdot}\ ]^{\cdot}$$

whereas $[\ \mathsf{m}\ ]^{\cdot}$ equals

$$[\ \langle\langle \mathsf{N}, \underline{\mathsf{E}}, \langle \oplus, \underline{0}\rangle^{\cdot}, \oplus, \underline{1}\rangle^{\cdot}\rangle^{\cdot}\ ]^{\cdot}.$$

Chapter 16

Counterproofs

16.1 Introduction

$[\,\mathsf{T} \equiv \mathsf{F}\,]^{\cdot}$ fails. You can prove that $[\,\mathsf{T} \equiv \mathsf{F}\,]^{\cdot}$ fails by proving that the contradiction $[\,\bot \equiv \mathsf{T}\,]^{\cdot}$ follows from $[\,\mathsf{T} \equiv \mathsf{F}\,]^{\cdot}$:

[**Mac lemma L16.1.1** : $\mathsf{T} \equiv \mathsf{F} \vdash \bot \equiv \mathsf{T}$]

[**Mac proof of** L16.1.1**:**

L1 :	Premise $\rhd$	$\mathsf{T} \equiv \mathsf{F}$	;
L2 :	Block $\rhd$	Begin	;
L3 :	Algebra $\rhd$	$\bot$	;
L4 :	Reverse $\rhd$ IfTrue $\rhd$	$\text{if}(\mathsf{T}, \bot, \mathsf{T})$	;
L5 :	Replace $\rhd$ L1 $\rhd$	$\text{if}(\mathsf{F}, \bot, \mathsf{T})$	;
L6 :	Replace $\rhd$ IfFalse $\rhd$	T	;
L7 :	Block $\rhd$	End	;
L8 :	Repetition $\rhd$ L6 $\rhd$	$\bot \equiv \mathsf{T}$	]

I now introduce the notion of an *antilemma*:

[**Mac antilemma TUnequalF** : $\mathsf{T} \equiv \mathsf{F}$]

[**Mac proof of** TUnequalF**:**

L1 :	Antilemma $\rhd$	$\mathsf{T} \equiv \mathsf{F}$	;
L2 :	Block $\rhd$	Begin	;
L3 :	Algebra $\rhd$	$\bot$	;
L4 :	Reverse $\rhd$ IfTrue $\rhd$	$\text{if}(\mathsf{T}, \bot, \mathsf{T})$	;
L5 :	Replace $\rhd$ L1 $\rhd$	$\text{if}(\mathsf{F}, \bot, \mathsf{T})$	;
L6 :	Replace $\rhd$ IfFalse $\rhd$	T	;
L7 :	Block $\rhd$	End	;
L8 :	Repetition $\rhd$ L6 $\rhd$	$\bot \equiv \mathsf{T}$	]

$[\,\text{TUnequalF}\,]^{\cdot}$ antilemma rule TUnequalF

I will refer to the proof of $[\,\text{TUnequalF}\,]^{\cdot}$ as a *counterproof* of $[\,\mathsf{T} \equiv \mathsf{F}\,]^{\cdot}$, and I will say that the proof *disproves* $[\,\mathsf{T} \equiv \mathsf{F}\,]^{\cdot}$.

The counterproof of $[\,\mathsf{T} \equiv \mathsf{F}\,]^{\cdot}$ merely differs from the proof of $[\,\mathsf{T} \equiv \mathsf{F} \vdash \bot \equiv \mathsf{T}\,]^{\cdot}$ in that $[\,\text{Premise}\,]^{\cdot}$ is replaced by $\left[\,\boxed{\text{Antilemma}}\,\right]^{\cdot}$ in the argumentation in the first line.

In general, if $[\,\mathcal{A}\,]^{\cdot}$ is an equation, then a proof of $[\,\mathcal{A} \vdash \bot \equiv \mathsf{T}\,]^{\cdot}$ can be converted into a counterproof of $[\,\mathcal{A}\,]^{\cdot}$ merely by changing $[\,\text{Premise}\,]^{\cdot}$ to $[\,\text{Antilemma}\,]^{\cdot}$. If $[\,\mathcal{A}\,]^{\cdot}$ contains the character $[\,\vdash\,]^{\circ}$, then counterproofs have some extra features which I present later.

I now disprove $[\,3 \equiv 4\,]^{\cdot}$:

[**Mac antilemma L16.1.3** : $3 \equiv 4$]

[**Mac proof of** L16.1.3**:**

L1 :	Antilemma $\rhd$	$3 \equiv 4$	;
L2 :	Block $\rhd$	Begin	;
L3 :	Algebra $\rhd$	T	;
L4 :	Reverse $\rhd$ $3 = 3 \equiv \mathsf{T}$ $\rhd$	$3 = 3$	;
L5 :	Replace $\rhd$ L1 $\rhd$	$3 = 4$	;
L6 :	Replace $\rhd$ $3 = 4 \equiv \mathsf{F}$ $\rhd$	F	;
L7 :	Block $\rhd$	End	;
L8 :	TUnequalF $\rhd$ L6 $\rhd$	$\bot$	]

Note that the term $[\,\bot\,]^{\cdot}$ in Line $[\,\mathsf{L8}\,]^{\cdot}$ occurs in a position where an equation is expected, so Line $[\,\mathsf{L8}\,]^{\cdot}$ reads $[\,\bot \equiv \mathsf{T}\,]^{\cdot}$.

In Line $[\,\mathsf{L8}\,]^{\cdot}$ I use the antilemma $[\,\text{TUnequalF}\,]^{\cdot}$ as the rule $[\,\mathsf{T} \equiv \mathsf{F} \vdash \bot \equiv \mathsf{T}\,]^{\cdot}$. In general, whenever $[\,\mathcal{A}\,]^{\cdot}$ is an equation, the antilemma $[\,\mathcal{A}\,]^{\cdot}$ is equivalent to the lemma $[\,\mathcal{A} \vdash \bot\,]^{\cdot}$.

In the proof above, I prove $[\,\mathsf{T} \equiv \mathsf{F}\,]^{\cdot}$ from $[\,3 \equiv 4\,]^{\cdot}$, and then I use that $[\,\mathsf{T} \equiv \mathsf{F}\,]^{\cdot}$ fails according to $[\,\text{TUnequalF}\,]^{\cdot}$. This is a quite common structure for a counterproof.

$[\,\mathsf{x} + 2 \equiv 4\,]^{\cdot}$ fails because $[\,\mathsf{x} + 2 \equiv 4\,]^{\cdot}$ fails for at least one value of $[\,\mathsf{x}\,]^{\cdot}$:

[**Mac antilemma L16.1.4** : $\mathsf{x} + 2 \equiv 4$]

[**Mac proof of** L16.1.4**:**

L1 :	Antilemma $\rhd$	$\mathsf{x} + 2 \equiv 4$	;
L2 :	Block $\rhd$	Begin	;
L3 :	Algebra $\rhd$	3	;
L4 :	Reverse $\rhd$ $1 + 2 \equiv 3$ $\rhd$	$1 + 2$	;
L5 :	Replace $\rhd$ L1 $\rhd$	4	;
L6 :	Block $\rhd$	End	;
L7 :	L16.1.3 $\rhd$ L5 $\rhd$	$\bot$	]

$[\,\text{Antilemma}\,]^{\cdot}$ counterproof disproves antilemma

In a proof of $[\, x + 2 \equiv 4 \vdash \bot \,]^{\cdot}$ you have to state $[\, x + 2 \equiv 4 \,]^{\cdot}$ verbatim as a premise. A counterproof gives a bit more freedom in that in a counterproof of $[\, x + 2 \equiv 4 \,]^{\cdot}$ you may use $[\, x + 2 \equiv 4 \,]^{\cdot}$ as a new rule. Among other, you may replace $[\, x \,]^{\cdot}$ by any term:

[**Mac proof of** L16.1.4**:**

L1 :	Antilemma ▷	$1 + 2 \equiv 4$	;
L2 :	Block ▷	Begin	;
L3 :	Algebra ▷	3	;
L4 :	Reverse ▷ $1 + 2 \equiv 3$ ▷	$1 + 2$	;
L5 :	Repetition ▷ L1 ▷	4	;
L6 :	Block ▷	End	;
L7 :	L16.1.3 ▷ L5 ▷	$\bot$	]

In general, you may use $[$ Antilemma $]^{\cdot}$ anywhere a rule may occur:

[**Mac proof of** L16.1.4**:**

L2 :	Block ▷	Begin	;
L3 :	Algebra ▷	3	;
L4 :	Reverse ▷ $1 + 2 \equiv 3$ ▷	$1 + 2$	;
L5 :	Antilemma ▷	4	;
L6 :	Block ▷	End	;
L7 :	L16.1.3 ▷ L5 ▷	$\bot$	]

Exercise 16.1.5 Prove or disprove the following:

(a) $[\, \langle x, y, z \rangle$ tail head $\equiv x \,]^{\cdot}$.

(b) $[\, \langle x, y, z \rangle$ tail head $\equiv y \,]^{\cdot}$.

(c) $[\, \langle x, y, z \rangle$ tail head $\equiv z \,]^{\cdot}$.

16.2 A useful rule

The following is convenient for counterproofs:

[**Mac lemma CounterTF** : $x \equiv \mathsf{T} \vdash x \equiv \mathsf{F} \vdash \bot$]

[**Mac proof of** CounterTF**:**

L1 :	Premise ▷	$x \equiv \mathsf{T}$	;
L2 :	Premise ▷	$x \equiv \mathsf{F}$	;
L3 :	Commutativity ▷ L1 ▷	$\mathsf{T} \equiv x$	;
L4 :	Transitivity ▷ L3 ▷ L2 ▷	$\mathsf{T} \equiv \mathsf{F}$	;
L5 :	TUnequalF ▷ L4 ▷	$\bot$	]

As an example, here you have a counterproof of $[\, 2 \leq 1 \,]^{\cdot}$:

[CounterTF]˙ rule CounterTF

[**Mac antilemma L16.2.2** : $2 \leq 1$]

[**Mac proof of** L16.2.2:

L1 :	Antilemma ⊳	$2 \leq 1$	;
L2 :	Replace ⊳ $2 \leq 1 \equiv \mathsf{F}$ ⊳	$2 \leq 1 \equiv \mathsf{F}$	;
L3 :	CounterTF ⊳ L1 ⊳ L2 ⊳	$\perp$	]

16.3 Counterexamples

[$x + y - y \equiv x$]˙ fails:

[**Mac antilemma L16.3.1** : $x + y - y \equiv x$]

To prove that [$x + y - y \equiv x$]˙ fails, you just need to give one value for [x]˙ and one value for [y]˙ for which [$x + y - y \equiv y$]˙ fails. As an example, [$x + y - y \equiv y$]˙ fails for

[$x \equiv 0.1F; y \equiv 1F$]˙

since [$x + y - y \equiv 1F$]˙ for these values of [x]˙ and [y]˙. Here you have a formal proof:

[**Mac proof of** L16.3.1:

L1 :	Block ⊳	Begin	;
L2 :	Algebra ⊳	T	;
L3 :	Reverse ⊳ $0.1F + 1F - 1F = 1 \equiv \mathsf{T}$ ⊳	$0.1F + 1F - 1F = 1$	;
L4 :	Replace ⊳ Antilemma ⊳	$0.1F = 1$	;
L5 :	Replace ⊳ $0.1F = 1 \equiv \mathsf{F}$ ⊳	F	;
L6 :	Block ⊳	End	;
L7 :	TUnequalF ⊳ L5 ⊳	$\perp$	]

I will refer to

[$x \equiv 0.1F; y \equiv 1F$]˙

as a *counterexample* to [$x + y - y \equiv x$]˙.

Exercise 16.3.2 Give counterexamples to the following:

(a) [$x :: y \equiv y :: x$]˙.

(b) [$n! \equiv n \cdot (n - 1)!$]˙.

(c) [$\text{if}(x, x, x) \equiv \text{if}(x, x, \mathsf{F})$]˙.

---
counterexample

16.4 Counterexamples to terms

$[\, \mathsf{x} \leq \mathsf{x} \,]^{\cdot}$ fails:

[**Mac antilemma L16.4.1** : $\mathsf{x} \leq \mathsf{x}$]

[**Mac proof of** L16.4.1:

$$\begin{array}{lll}
\mathsf{L1}: & \text{Algebra} \rhd & \bot \quad ; \\
\mathsf{L2}: & \text{StrictWeaklyLessThanX} \rhd & \bot \leq \bot \quad ; \\
\mathsf{L3}: & \text{Replace} \rhd \text{Antilemma} \rhd & \mathsf{T} \quad]
\end{array}$$

When I state that $[\, \mathsf{x} \leq \mathsf{x} \,]^{\cdot}$ fails, $[\, \mathsf{x} \leq \mathsf{x} \,]^{\cdot}$ occurs in a position where an equation is expected. Therefore, $[\, \mathsf{x} \leq \mathsf{x} \,]^{\cdot}$ is shorthand for $[\, \mathsf{x} \leq \mathsf{x} \equiv \mathsf{T} \,]^{\cdot}$. Hence, a counterexample to $[\, \mathsf{x} \leq \mathsf{x} \,]^{\cdot}$ is a value of $[\, \mathsf{x} \,]^{\cdot}$ for which $[\, \mathsf{x} \leq \mathsf{x} \,]^{\cdot}$ differs from $[\, \mathsf{T} \,]^{\cdot}$. In the proof above I used

$$[\, \mathsf{x} \equiv \bot \,]^{\cdot}$$

as a counterexample.

Exercise 16.4.2 Give counterexamples to the following:

(a) $[\, \mathsf{x} + \mathsf{y} = \mathsf{y} = \mathsf{x} \,]^{\cdot}$.

(b) $[\, \mathsf{x} \vee \neg \mathsf{x} \,]^{\cdot}$.

16.5 Counterexamples to implications

[**Mac antilemma L16.5.1** : $\mathsf{x} \in \mathbf{Z} \rightarrow \mathsf{y} \in \mathbf{N} \rightarrow \mathsf{x} \leq \mathsf{y}$]

[**Mac proof of** L16.5.1:

$$\begin{array}{lll}
\mathsf{L1}: & \text{Antilemma} \rhd & 2 \in \mathbf{Z} \rightarrow 1 \in \mathbf{N} \rightarrow 2 \leq 1 \quad ; \\
\mathsf{L2}: & \text{TypeNumeralInZ} \rhd & 2 \in \mathbf{Z} \quad ; \\
\mathsf{L3}: & \mathsf{L1} \unrhd \mathsf{L2} \rhd & 1 \in \mathbf{N} \rightarrow 2 \leq 1 \quad ; \\
\mathsf{L4}: & \text{TypeNumeralInN} \rhd & 1 \in \mathbf{N} \quad ; \\
\mathsf{L5}: & \mathsf{L3} \unrhd \mathsf{L4} \rhd & 2 \leq 1 \quad ; \\
\mathsf{L6}: & \text{Computation} \rhd & 2 \leq 1 \equiv \mathsf{F} \quad ; \\
\mathsf{L7}: & \text{CounterTF} \rhd \mathsf{L5} \rhd \mathsf{L6} \rhd & \bot \quad]
\end{array}$$

The counterproof above builds on the following counterexample:

$$[\, \mathsf{x} \equiv 2; \mathsf{y} \equiv 1 \,]^{\cdot}$$

The statement $[\, \mathsf{x} \in \mathbf{Z} \rightarrow \mathsf{y} \in \mathbf{N} \rightarrow \mathsf{x} \leq \mathsf{y} \,]^{\cdot}$ has two hypotheses, namely

$$[\, \mathsf{x} \in \mathbf{Z} \,]^{\cdot}$$

and

$$[\, \mathsf{y} \in \mathbf{N} \,]^{\cdot}$$

and one consequence, namely

$$[\, \mathsf{x} \le \mathsf{y} \,]^{\cdot}.$$

The counterexample $[\, \mathsf{x} \equiv 2; \mathsf{y} \equiv 1 \,]^{\cdot}$ has the property that it is a counterexample to the consequence and, at the same time, satisfies the hypotheses. You should note carefully that a counterexample to an implication is something that makes the hypotheses hold and the consequence fail.

Exercise 16.5.2

(a) Is $[\, \mathsf{x} \equiv 0 \,]^{\cdot}$ a counterexample to $[\, \mathsf{x} \in \mathbf{Z} \to \mathsf{x} \ne 0 \,]^{\cdot}$?

(b) Is $[\, \mathsf{x} \equiv \bullet \,]^{\cdot}$ a counterexample to $[\, \mathsf{x} \in \mathbf{Z} \to \mathsf{x} \ne 0 \,]^{\cdot}$?

(c) Is $[\, \mathsf{x} \equiv 1 \,]^{\cdot}$ a counterexample to $[\, \mathsf{x} \in \mathbf{Z} \to \mathsf{x} \ne 0 \,]^{\cdot}$?

Exercise 16.5.3 Prove the following or give counterexamples:

(a) $[\, \mathsf{x} \in \mathbf{B} \to \mathsf{y} \in \mathbf{B} \to \mathsf{x} \vee \neg \mathsf{x} \wedge \mathsf{y} \Leftrightarrow \mathsf{x} \vee \mathsf{y} \,]^{\cdot}$.

(b) $[\, \mathsf{x} \in \mathbf{B} \to \mathsf{y} \in \mathbf{B} \to (\mathsf{x} \vee \mathsf{y}) \wedge \mathsf{x} \Leftrightarrow \mathsf{x} \wedge \mathsf{y} \,]^{\cdot}$.

16.6 Answers

Answer 16.1.5

[**Mac lemma L16.6.1** : $\langle \mathsf{x}, \mathsf{y}, \mathsf{z} \rangle$ tail head $\equiv \mathsf{y}$]

[**Mac proof of** L16.6.1**:**

L1 :	Algebra ▷	$\langle \mathsf{x}, \mathsf{y}, \mathsf{z} \rangle$ tail head	;
L2 :	Replace ▷ TailPair ▷	$\langle \mathsf{y}, \mathsf{z} \rangle$ head	;
L3 :	Replace ▷ HeadPair ▷	y	]

[**Mac antilemma L16.6.2** : $\langle \mathsf{x}, \mathsf{y}, \mathsf{z} \rangle$ tail head $\equiv \mathsf{x}$]

[**Mac proof of** L16.6.2**:**

L1 :	Algebra ▷	$\bot$	;
L2 :	Reverse ▷ Antilemma ▷	$\langle \bot, \mathsf{T}, 117 \rangle$ tail head	;
L3 :	L16.6.1 ▷	T	]

[**Mac antilemma L16.6.3** : $\langle \mathsf{x}, \mathsf{y}, \mathsf{z} \rangle$ tail head $\equiv \mathsf{z}$]

[**Mac proof of** L16.6.3**:**

L1 :	Algebra ▷	$\bot$	;
L2 :	Reverse ▷ Antilemma ▷	$\langle 117, \mathsf{T}, \bot \rangle$ tail head	;
L3 :	L16.6.1 ▷	T	]

Answer 16.3.2 (a) $[\, \mathsf{x} \equiv 117; \mathsf{y} \equiv \bullet \,]^{\cdot}$. (b) $[\, \mathsf{x} \equiv 0 \,]^{\cdot}$. (c) $[\, \mathsf{x} \equiv \mathsf{N} \therefore \mathsf{N} \therefore \mathsf{N} \,]^{\cdot}$ (which is a weak representation of falsehood).

Answer 16.4.2 (a) $[\,\mathsf{x} \equiv \bot; \mathsf{y} \equiv \bot\,]^{\cdot}$. (b) $[\,\mathsf{x} \equiv \bot\,]^{\cdot}$.
Answer 16.5.2 (a) Yes. (b) No. (c) No.
Answer 16.5.3 (a)

[**Mac lemma L16.6.4** : $\mathsf{x} \in \mathbf{B} \to \mathsf{y} \in \mathbf{B} \to \mathsf{x} \lor \neg\mathsf{x} \land \mathsf{y} \Leftrightarrow \mathsf{x} \lor \mathsf{y}$]

[**Mac proof of** L16.6.4**:**
L1 : Tautology ▷ $\mathsf{x} \in \mathbf{B} \to \mathsf{y} \in \mathbf{B} \to \mathsf{x} \lor \neg\mathsf{x} \land \mathsf{y} \Leftrightarrow \mathsf{x} \lor \mathsf{y}$]

x	∨	¬	x	∧	y	⇔	x	∨	y
T	**T**	**F**	**T**	**F**	**T**	**T**	**T**	**T**	**T**
F	**T**	**T**	**F**	**T**	**T**	**T**	**F**	**T**	**T**
T	**T**	**F**	**T**	**F**	**F**	**T**	**T**	**T**	**F**
F	**F**	**T**	**F**	**F**	**F**	**T**	**F**	**F**	**F**

(b) Counterexample: $[\,\mathsf{x} \equiv \mathsf{T}; \mathsf{y} \equiv \mathsf{F}\,]^{\cdot}$.

Chapter 17

Information

17.1 Properties

If $[\ \mathsf{f}\ '\ \mathsf{x} \equiv \mathsf{T}\ ]^{\cdot}$ then I will say that $[\ \mathsf{x}\ ]^{\cdot}$ has the $[\ \mathsf{f}\ ]^{\cdot}$ *property*. As an example, I will refer to

$$[\ \lambda \mathsf{x}.\mathsf{x} = 2\ ]^{\cdot}$$

as the "equals two" property. You have

$$[\ (\lambda \mathsf{x}.\mathsf{x} = 2)\ '\ 2.0\mathtt{F} \equiv \mathsf{T}\ ]^{\cdot}$$

so you have that $[\ 2.0\mathtt{F}\ ]^{\cdot}$ has the "equals two" property. You also have

$$[\ (\lambda \mathsf{x}.\mathsf{x} = 2)\ '\ 2 \equiv \mathsf{T}\ ]^{\cdot}$$

so $[\ 2\ ]^{\cdot}$ also has the "equals two" property. You have that

$[\ (\lambda \mathsf{x}.\mathsf{x} = 2)\ '\ 3 \equiv \mathsf{T}\ ]^{\cdot}$ fails and

$[\ (\lambda \mathsf{x}.\mathsf{x} = 2)\ '\ \bot \equiv \mathsf{T}\ ]^{\cdot}$ fails

so neither $[\ 3\ ]^{\cdot}$ nor $[\ \bot\ ]^{\cdot}$ has the "equals two" property.

I now introduce the "equals three" property:

$$[\ \lambda \mathsf{x}.\mathsf{x} = 3\ ]^{\cdot}$$

and the "precision two" property:

$$[\ \lambda \mathsf{x}.\#\mathsf{x} = 2\ ]^{\cdot}.$$

Here you have a table that shows which of the properties $[\ 2\ ]^{\cdot}$, $[\ 2.0\mathtt{F}\ ]^{\cdot}$, $[\ 3\ ]^{\cdot}$, and $[\ \bot\ ]^{\cdot}$ have:

---

property

	2	2.0F	3	$\bot$
$\lambda x.x = 2$	Yes	Yes	No	No
$\lambda x.x = 3$	No	No	Yes	No
$\lambda x.\#x = 2$	No	Yes	No	No

Two entities are equal if they have the same properties and they differ if they do not. As an example, $[\,2\,]^{\cdot}$ and $[\,3\,]^{\cdot}$ differ because $[\,2\,]^{\cdot}$ has the "equals two" property which $[\,3\,]^{\cdot}$ does not have. $[\,2\,]^{\cdot}$ and $[\,2.0\mathrm{F}\,]^{\cdot}$ differ because $[\,2.0\mathrm{F}\,]^{\cdot}$ has the "precision two" property which $[\,2\,]^{\cdot}$ does not have. $[\,2\,]^{\cdot}$ and $[\,\bot\,]^{\cdot}$ differ because $[\,2\,]^{\cdot}$ has the "equals two" property which $[\,\bot\,]^{\cdot}$ does not have. $[\,2+2\,]^{\cdot}$ and $[\,4\,]^{\cdot}$ are equal because $[\,f\,'\,(2+2) \equiv \mathsf{T}\,]^{\cdot}$ if and only if $[\,f\,'\,4 \equiv \mathsf{T}\,]^{\cdot}$.

Exercise 17.1.1 Does $[\,2+2\,]^{\cdot}$ have the $[\,\lambda x.\#x = \infty\,]^{\cdot}$ property?

17.2 Maps

Now recall from Section 9.4 that I represent everything by maps, and recall from Section 9.5 that there are three kinds of maps, namely $[\,\mathsf{T}\,]^{\cdot}$, $[\,\bot\,]^{\cdot}$, and functions. Maps satisfy the following axioms:

[**Mac rule ApplyT** : $\mathsf{T}\,'\,y \equiv \mathsf{T}$]

[**Mac rule ApplyLambda** : $(\lambda u.v)\,'\,y \equiv \langle v \mid u{:=}y \rangle$]

[**Mac rule ApplyBottom** : $\bot\,'\,y \equiv \bot$]

[**Mac rule CaseT** : $\mathrm{case}(\mathsf{T}, y, z) \equiv y$]

[**Mac rule CaseLambda** : $\mathrm{case}(\lambda u.v, y, z) \equiv z$]

[**Mac rule CaseBottom** : $\mathrm{case}(\bot, y, z) \equiv \bot$]

The property $[\,\lambda x.\mathsf{T}\,]^{\cdot}$ is *universal* in the sense that any map $[\,x\,]^{\cdot}$ has the $[\,\lambda x.\mathsf{T}\,]^{\cdot}$ property:

$[\,(\lambda x.\mathsf{T})\,'\,2 \equiv \mathsf{T}\,]^{\cdot}$,

$[\,(\lambda x.\mathsf{T})\,'\,2.0\mathrm{F} \equiv \mathsf{T}\,]^{\cdot}$,

$[\,(\lambda x.\mathsf{T})\,'\,3 \equiv \mathsf{T}\,]^{\cdot}$, and

$[\,(\lambda x.\mathsf{T})\,'\,\bot \equiv \mathsf{T}\,]^{\cdot}$.

The property $[\,\mathsf{T}\,]^{\cdot}$ is also universal:

$[\,\mathsf{T}\,'\,2 \equiv \mathsf{T}\,]^{\cdot}$,

universal property

$[\, \mathsf{T}\, '\, 2.0\mathtt{F} \equiv \mathsf{T} \,]^{\cdot}$,

$[\, \mathsf{T}\, '\, 3 \equiv \mathsf{T} \,]^{\cdot}$, and

$[\, \mathsf{T}\, '\, \bot \equiv \mathsf{T} \,]^{\cdot}$.

$[\, \lambda \mathsf{x}.\mathsf{T} \,]^{\cdot}$ and $[\, \mathsf{T} \,]^{\cdot}$ are the only universal properties. To see that, proceed as follows: A property is a map. Hence, a property is $[\, \mathsf{T} \,]^{\cdot}$, $[\, \bot \,]^{\cdot}$, or a function. $[\, \mathsf{T} \,]^{\cdot}$ is a universal property and $[\, \bot \,]^{\cdot}$ is not. If $[\, \mathsf{f} \,]^{\cdot}$ is a function and if $[\, \mathsf{f} \,]^{\cdot}$ is universal, then $[\, \mathsf{f}\, '\, \mathsf{x} \equiv \mathsf{T} \,]^{\cdot}$ for all maps $[\, \mathsf{x} \,]^{\cdot}$. Hence, $[\, \mathsf{f}\, '\, \mathsf{x} \equiv (\lambda \mathsf{x}.\mathsf{T})\, '\, \mathsf{x} \,]^{\cdot}$ for all $[\, \mathsf{x} \,]^{\cdot}$. Hence, $[\, \mathsf{f} \equiv \lambda \mathsf{x}.\mathsf{T} \,]^{\cdot}$ according to Section 8.7. In conclusion, any universal property equals $[\, \mathsf{T} \,]^{\cdot}$ or equals $[\, \lambda \mathsf{x}.\mathsf{T} \,]^{\cdot}$.

The object $[\, \bot \,]^{\cdot}$ is special in that it has very few properties. Actually, $[\, \bot \,]^{\cdot}$ merely has the two universal ones.

Exercise 17.2.1

(a) Does $[\, \lambda \mathsf{x}.\bot \,]^{\cdot}$ have the $[\, \lambda \mathsf{y}.\mathrm{case}(\mathsf{y}, \bot, \mathsf{T}) \,]^{\cdot}$ property?

(b) Does $[\, \lambda \mathsf{x}.\bot \equiv \bot \,]^{\cdot}$ hold?

17.3 Information contents

I refer to the properties of an entity $[\, \mathsf{x} \,]^{\cdot}$ as the *information contents* of $[\, \mathsf{x} \,]^{\cdot}$. as an example, $[\, 2.0\mathtt{F} \,]^{\cdot}$ contains the information that it equals two, that it has precision two, that it is a decimal fraction, and so on. $[\, 3 \,]^{\cdot}$ contains the information that it equals three, has precision $[\, \infty \,]^{\cdot}$, is a decimal fraction, and so on.

$[\, \bot \,]^{\cdot}$ contains very little infomation. Actually $[\, \bot \,]^{\cdot}$ merely contains two pieces of infomation, namely $[\, \mathsf{T} \,]^{\cdot}$ and $[\, \lambda \mathsf{x}.\mathsf{T} \,]^{\cdot}$.

17.4 Information comparison

I use

$$\left[\, \boxed{\mathsf{x} \preceq \mathsf{y}} \,\right]^{\cdot}$$

to denote that $[\, \mathsf{y} \,]^{\cdot}$ contains all the information that $[\, \mathsf{x} \,]^{\cdot}$ contains plus possibly some more. Here you have an example:

$[\, \bot \preceq 2 \,]^{\cdot}$

holds because $[\, 2 \,]^{\cdot}$ has all the properties that $[\, \bot \,]^{\cdot}$ has. To see that, proceed as follows: $[\, \bot \,]^{\cdot}$ merely has two properties, namely $[\, \lambda \mathsf{x}.\mathsf{T} \,]^{\cdot}$ and $[\, \mathsf{T} \,]^{\cdot}$. You

---

information contents
$[\, \mathsf{x} \preceq \mathsf{y} \,]^{\cdot}$ x weakly less information y

have already seen that $[\, 2 \,]^{\cdot}$ has both of these properties (plus a lot more), so $[\, \bot \preceq 2 \,]^{\cdot}$ holds. Here you have another example:

$$[\, \bot \preceq 3 \,]^{\cdot}.$$

The argument is the same: $[\, \bot \,]^{\cdot}$ has properties $[\, \lambda \mathsf{x}.\mathsf{T} \,]^{\cdot}$ and $[\, \mathsf{T} \,]^{\cdot}$, and $[\, 3 \,]^{\cdot}$ has both of them (plus more). Here is yet another example:

$$[\, \bot \preceq \bot \,]^{\cdot}.$$

The leftmost $[\, \bot \,]^{\cdot}$ has two properties, namely $[\, \lambda \mathsf{x}.\mathsf{T} \,]^{\cdot}$ and $[\, \mathsf{T} \,]^{\cdot}$, and the rightmost $[\, \bot \,]^{\cdot}$ has both of them (plus no more, but that is ok).

Here you have an example of the opposite:

$$[\, 2 \not\preceq 3 \,]^{\cdot}$$

holds because $[\, 2 \,]^{\cdot}$ has a property that $[\, 3 \,]^{\cdot}$ does not have (just think of the "equals two" property). You also have

$$[\, 3 \not\preceq 2 \,]^{\cdot}.$$

This is so because $[\, 3 \,]^{\cdot}$ has a property that $[\, 2 \,]^{\cdot}$ does not have (just think of the "equals three" property).

Here you have some further examples:

$$[\, 2 \not\preceq \bot \,]^{\cdot},$$

$$[\, 3 \not\preceq \bot \,]^{\cdot},$$

$$[\, 2 \preceq 2 \,]^{\cdot}, \text{ and}$$

$$[\, 3 \preceq 3 \,]^{\cdot}.$$

If $[\, \mathsf{x} \,]^{\cdot}$ and $[\, \mathsf{y} \,]^{\cdot}$ are terms, then $[\, \mathsf{x} \preceq \mathsf{y} \,]^{\cdot}$ is shorthand for some equation. I formally define $[\, \mathsf{x} \preceq \mathsf{y} \,]^{\cdot}$ later. $[\, \mathsf{x} \not\preceq \mathsf{y} \,]^{\cdot}$ is shorthand for $[\, \mathsf{x} \preceq \mathsf{y} \vdash \bot \,]^{\cdot}$.

Exercise 17.4.1 Which of the following hold?

(a) $[\, \bot \preceq \mathsf{T} \,]^{\cdot}$.

(b) $[\, \mathsf{T} \preceq \bot \,]^{\cdot}$.

(c) $[\, \mathsf{T} \preceq \mathsf{T} \,]^{\cdot}$.

(d) $[\, \mathsf{T} \preceq \mathsf{F} \,]^{\cdot}$.

(e) $[\, \mathsf{F} \preceq \mathsf{T} \,]^{\cdot}$.

17.5 Separability

If $[\, x \,]^{\cdot}$ has a property that $[\, y \,]^{\cdot}$ does not have and vice versa, then I say that $[\, x \,]^{\cdot}$ and $[\, y \,]^{\cdot}$ are *separable* and write $\left[\, \boxed{x \# y} \,\right]^{\cdot}$. As an example, you have

$$[\, 2 \# 3 \,]^{\cdot}.$$

You have $[\, \bot \preceq 2 \,]^{\cdot}$ so $[\, \bot \,]^{\cdot}$ and $[\, 2 \,]^{\cdot}$ are *inseparable*.

Separability is important in computer applications. Suppose you want a computer to do some computation, and suppose you expect the result of the computation to be either $[\, 2 \,]^{\cdot}$ or $[\, 3 \,]^{\cdot}$. Further suppose you want to know the result of the computation. In this situation you can equipe the computer with a green and a red lamp and program the computer to turn on the green lamp if the result is $[\, 2 \,]^{\cdot}$ and to turn on the red one if the result is $[\, 3 \,]^{\cdot}$. You can program the computer to do that because $[\, 2 \,]^{\cdot}$ and $[\, 3 \,]^{\cdot}$ are separable.

Now suppose you want the same computer to make another computation and suppose you expect the result of the computation to be either $[\, 2 \,]^{\cdot}$ or $[\, \bot \,]^{\cdot}$. You can easily program the computer to turn on the green lamp if the result of the computation is $[\, 2 \,]^{\cdot}$, but you cannot program the computer to turn on the red lamp if the result is $[\, \bot \,]^{\cdot}$.

The morale is that if a computation has several possible outcomes and you want to know which one, then you must ensure that the possible outcomes are separable.

Exercise 17.5.1 Which of the following hold?

(a) $[\, \bot \# \mathsf{T} \,]^{\cdot}$.

(b) $[\, \bot \# \mathsf{F} \,]^{\cdot}$.

(c) $[\, \mathsf{T} \# \mathsf{F} \,]^{\cdot}$.

17.6 Relation to implication

If $[\, x \,]^{\cdot}$ and $[\, y \,]^{\cdot}$ are terms, if $[\, f \,]^{\cdot}$ is a variable, and if $[\, f \,]^{\cdot}$ does not occur free in $[\, x \,]^{\cdot}$ and $[\, y \,]^{\cdot}$, then

$$[\, x \preceq y \,]^{\cdot}$$

holds if and only if

$$[\, f \,'\, x \rightarrow f \,'\, y \,]^{\cdot}$$

holds. $[\, f \,'\, x \rightarrow f \,'\, y \,]^{\cdot}$ says that if $[\, f \,]^{\cdot}$ is a property of $[\, x \,]^{\cdot}$ then $[\, f \,]^{\cdot}$ is also a property of $[\, y \,]^{\cdot}$. Since $[\, f \,]^{\cdot}$ is a variable, $[\, f \,'\, x \rightarrow f \,'\, y \,]^{\cdot}$ holds if $[\, f \,'\, x \,]^{\cdot}$ implies $[\, f \,'\, y \,]^{\cdot}$ for all properties $[\, f \,]^{\cdot}$.

Stated formally you have:

---

separable
$[\, x \# y \,]^{\cdot}$ x separable y
inseparable

[**Mac rule InfoImply** : $x \preceq y \vdash f \,' x \rightarrow f \,' y$]

[**Mac rule ImplyInfo** : **if** notfree(f, x) ∧ notfree(f, y) **then** $f \,' x \rightarrow f \,' y \vdash x \preceq y$]

I have taken the liberty to omit the side condition [notfree(f, x) ∧ notfree(f, y)]˙ in [InfoImply]˙ because [InfoImply]˙ holds even when that condition is not satisfied.

Here you have a formal proof of [$2 \not\preceq 3$]˙:

[**Mac lemma L17.6.1** : $2 \preceq 3 \vdash \bot$]

[**Mac proof of** L17.6.1**:**

L1 :	Premise ▷	$2 \preceq 3$	;
L2 :	InfoImply ▷ L1 ▷	$f \,' 2 \rightarrow f \,' 3$	;
L3 :	Replace ▷ L2 ▷	$(\lambda x.x = 2) \,' 2 \rightarrow (\lambda x.x = 2) \,' 3$	;
L4 :	Computation ▷	$(\lambda x.x = 2) \,' 2$	;
L5 :	L3 ⊵ L4 ▷	$(\lambda x.x = 2) \,' 3$	;
L6 :	Computation ▷	$(\lambda x.x = 2) \,' 3 \equiv F$	;
L7 :	CounterTF ▷ L5 ▷ L6 ▷	$\bot$	]

Exercise 17.6.2 Prove [$T \not\preceq F$]˙.

17.7 Properties of [$x \preceq y$]˙

[x]˙ has all properties that [x]˙ has:

[**Mac lemma InfoReflexive** : $x \preceq x$]

[**Mac proof of** InfoReflexive**:**

L1 :	Block ▷	Begin	;
L2 :	Hypothesis ▷	$f \,' x$	;
L3 :	Block ▷	End	;
L4 :	ImplyInfo ▷ L2 ▷	$x \preceq x$	]

If [x]˙ has all properties that [y]˙ has and vice versa, then [x]˙ and [y]˙ have the same properties, so in this case they are equal:

[**Mac rule InfoAntiSymmetry** : $x \preceq y \vdash y \preceq x \vdash x \equiv y$]

If [z]˙ has all properties that [y]˙ has and [y]˙ has all properties that [x]˙ has, then [z]˙ has all properties that [x]˙ has:

[**Mac lemma InfoTransitivity** : $x \preceq y \vdash y \preceq z \vdash x \preceq z$]

[InfoReflexive]˙ rule InfoReflexive
[InfoTransitivity]˙ rule InfoTransitivity

[**Mac proof of** InfoTransitivity:

L01 :	Premise ▷	$x \preceq y$	;
L02 :	Premise ▷	$y \preceq z$	;
L03 :	InfoImply ▷ L1 ▷	f ' x → f ' y	;
L04 :	InfoImply ▷ L2 ▷	f ' y → f ' z	;
L05 :	Block ▷	Begin	;
L06 :	Hypothesis ▷	f ' x	;
L07 :	L3 ⊵ L6 ▷	f ' y	;
L08 :	L4 ⊵ L7 ▷	f ' z	;
L09 :	Block ▷	End	;
L10 :	ImplyInfo ▷ L8 ▷	$x \preceq z$	]

[$\perp$]˙ has fewer properties than anything else:

[**Mac rule InfoBottom** : $\perp \preceq x$]

An object [b]˙ that satisfies [$b \preceq x$]˙ for some ordering relation [$\preceq$]° and all [x]˙ is known as a *bottom* object. [$\perp$]˙ is the bottom object of [$\preceq$]° within the universe of maps (this is where [$\perp$]˙ got its name from). As some other examples, [0]˙ is the bottom object for [$\leq$]° within [**N**]˙. [F]˙ is the bottom object for [$\leq$]˙ within [**B**]˙. [$\leq$]° has no bottom object within [**Z**]˙.

17.8 Monotonicity

[**Mac lemma Monotonicity** : $x \preceq y \vdash$ g ' x $\preceq$ g ' y]

[**Mac proof of** Monotonicity:

L1 :	Premise ▷	$x \preceq y$	;
L2 :	InfoImply ▷ L1 ▷	f ' x → f ' y	;
L3 :	Replace ▷ L2 ▷	(λu.f ' (g ' u)) ' x → (λu.f ' (g ' u)) ' y	;
L4 :	ApplyLambda ▷	(λu.f ' (g ' u)) ' x ≡ f ' (g ' x)	;
L5 :	Replace' ▷ L4 ▷ L3 ▷	f ' (g ' x) → (λu.f ' (g ' u)) ' y	;
L6 :	ApplyLambda ▷	(λu.f ' (g ' u)) ' y ≡ f ' (g ' y)	;
L7 :	Replace' ▷ L6 ▷ L5 ▷	f ' (g ' x) → f ' (g ' y)	;
L8 :	ImplyInfo ▷ L7 ▷	g ' x $\preceq$ g ' y	]

The lemma above says something very central to computer science: if you take any function [g]˙, then the more information you give [g]˙ as input, the more information will [g]˙ give as output. This result has absolutely no counterpart in classical mathematics (since in classical mathematics, any two different entities are separable).

In the treatment above I have proved [Monotonicity]˙ from [InfoImply]˙ and [ImplyInfo]˙. Later on (in Appendix B) you will see that it may be more

---

bottom
[Monotonicity]˙ rule Monotonicity

convenient to do it the other way round, i.e. to take [Monotonicity]˙ as a fundamental rule and to prove [InfoImply]˙ and [ImplyInfo]˙ from the fundamental rules and definitions.

Here you have another formulation of monotonicity:

[**Mac lemma Monotonicity'** : $\mathsf{x} \preceq \mathsf{y} \vdash \langle \mathcal{A} \mid \mathsf{u{:=}x} \rangle \preceq \langle \mathcal{A} \mid \mathsf{u{:=}y} \rangle$]

[**Mac proof of** Monotonicity'**:**

L1 :	Premise ⊳	$\mathsf{x} \preceq \mathsf{y}$	;
L2 :	Monotonicity ⊳ L1 ⊳	$(\lambda \mathsf{u}.\mathcal{A})\ '\ \mathsf{x} \preceq (\lambda \mathsf{u}.\mathcal{A})\ '\ \mathsf{y}$	;
L3 :	ApplyLambda ⊳	$(\lambda \mathsf{u}.\mathcal{A})\ '\ \mathsf{x} \equiv \langle \mathcal{A} \mid \mathsf{u{:=}x} \rangle$	;
L4 :	Replace' ⊳ L3 ⊳ L2 ⊳	$\langle \mathcal{A} \mid \mathsf{u{:=}x} \rangle \preceq (\lambda \mathsf{u}.\mathcal{A})\ '\ \mathsf{y}$	;
L5 :	ApplyLambda ⊳	$(\lambda \mathsf{u}.\mathcal{A})\ '\ \mathsf{y} \equiv \langle \mathcal{A} \mid \mathsf{u{:=}y} \rangle$	;
L6 :	Replace' ⊳ L5 ⊳ L4 ⊳	$\langle \mathcal{A} \mid \mathsf{u{:=}x} \rangle \preceq \langle \mathcal{A} \mid \mathsf{u{:=}y} \rangle$	]

Exercise 17.8.3 Prove the following:

[**Mac lemma MonotonicityHead** : $\mathsf{x} \preceq \mathsf{y} \vdash \mathsf{x}$ head $\preceq \mathsf{y}$ head]

[**Mac lemma MonotonicityTail** : $\mathsf{x} \preceq \mathsf{y} \vdash \mathsf{x}$ tail $\preceq \mathsf{y}$ tail]

[**Mac lemma MonotonicityHead'** : $\mathsf{x} :: \mathsf{y} \preceq \mathsf{u} :: \mathsf{v} \vdash \mathsf{x} \preceq \mathsf{u}$]

[**Mac lemma MonotonicityTail'** : $\mathsf{x} :: \mathsf{y} \preceq \mathsf{u} :: \mathsf{v} \vdash \mathsf{y} \preceq \mathsf{v}$]

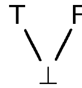
[**Mac lemma MonotonicityHead"** : $\mathsf{x} \preceq \mathsf{y} \vdash \mathsf{x} :: \mathsf{z} \preceq \mathsf{y} :: \mathsf{z}$]

[**Mac lemma MonotonicityTail"** : $\mathsf{y} \preceq \mathsf{z} \vdash \mathsf{x} :: \mathsf{y} \preceq \mathsf{x} :: \mathsf{z}$]

[**Mac lemma MonotonicityPair** : $\mathsf{x} \preceq \mathsf{u} \vdash \mathsf{y} \preceq \mathsf{v} \vdash \mathsf{x} :: \mathsf{y} \preceq \mathsf{u} :: \mathsf{v}$]

17.9 Diagrams of [$\mathsf{x} \preceq \mathsf{y}$]˙

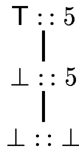
Here you have a *diagram* of [$\mathsf{x} \preceq \mathsf{y}$]˙ applied to [⊥]˙, [T]˙ and [F]˙.



[Monotonicity']˙	rule Monotonicity'
[MonotonicityHead]˙	rule MonotonicityHead
[MonotonicityTail]˙	rule MonotonicityTail
[MonotonicityHead']˙	rule MonotonicityHead'
[MonotonicityTail']˙	rule MonotonicityTail'
[MonotonicityHead"]˙	rule MonotonicityHead"
[MonotonicityTail"]˙	rule MonotonicityTail"
[MonotonicityPair]˙	rule MonotonicityPair
	diagram

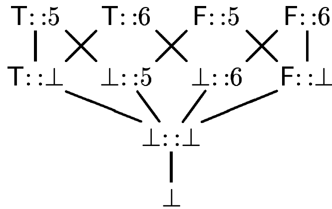
In the diagram, there is an *edge* from $[\,\bot\,]^{\cdot}$ up to $[\,\mathsf{T}\,]^{\cdot}$. That edge indicates that $[\,\bot \preceq \mathsf{T}\,]^{\cdot}$ holds. There is also an edge from $[\,\bot\,]^{\cdot}$ up to $[\,\mathsf{F}\,]^{\cdot}$. That edge indicates that $[\,\bot \preceq \mathsf{F}\,]^{\cdot}$ holds. There is no edge between $[\,\mathsf{T}\,]^{\cdot}$ and $[\,\mathsf{F}\,]^{\cdot}$. This absence of an edge indicates that $[\,\mathsf{T}\,]^{\cdot}$ and $[\,\mathsf{F}\,]^{\cdot}$ are separable.

Here you have a diagram of $[\,\mathsf{x} \preceq \mathsf{y}\,]^{\cdot}$ applied to $[\,\bot :: \bot\,]^{\cdot}$, $[\,\bot :: 5\,]^{\cdot}$ and $[\,\mathsf{T} :: 5\,]^{\cdot}$:



The two edges indicate that $[\,\bot :: \bot \preceq \bot :: 5\,]^{\cdot}$ and $[\,\bot :: 5 \preceq \mathsf{T} :: 5\,]^{\cdot}$ both hold. From $[\,\bot :: \bot\,]^{\cdot}$ you can move up to $[\,\mathsf{T} :: 5\,]^{\cdot}$ by following first the edge from $[\,\bot :: \bot\,]^{\cdot}$ to $[\,\bot :: 5\,]^{\cdot}$ and then the edge from $[\,\bot :: 5\,]^{\cdot}$ to $[\,\mathsf{T} :: 5\,]^{\cdot}$. Therefore, the diagram also says that $[\,\bot :: \bot \preceq \mathsf{T} :: 5\,]^{\cdot}$ holds even though there is no edge directly from $[\,\bot :: \bot\,]^{\cdot}$ to $[\,\mathsf{T} :: 5\,]^{\cdot}$.

Here you have a diagram of $[\,\mathsf{x} \preceq \mathsf{y}\,]^{\cdot}$ applied to ten mathematical entities:



As an example, you can move from $[\,\bot :: \bot\,]^{\cdot}$ up to $[\,\mathsf{F} :: 5\,]^{\cdot}$, so the diagram says that $[\,\bot :: \bot \preceq \mathsf{F} :: 5\,]^{\cdot}$ holds. Actually, you can move from $[\,\bot :: \bot\,]^{\cdot}$ up to $[\,\mathsf{F} :: 5\,]^{\cdot}$ in two different ways: you can go via $[\,\bot :: 5\,]^{\cdot}$ or via $[\,\mathsf{F} :: \bot\,]^{\cdot}$.

In the diagram, you cannot move from $[\,\mathsf{T} :: \bot\,]^{\cdot}$ up to $[\,\bot :: 5\,]^{\cdot}$, so the diagram says that $[\,\mathsf{T} :: \bot\,]^{\cdot}$ and $[\,\bot :: 5\,]^{\cdot}$ are separable.

Exercise 17.9.1 Draw a diagram of $[\,\preceq\,]^{\circ}$ applied to $[\,\langle\rangle\,]^{\cdot}$, $[\,\langle 1\rangle\,]^{\cdot}$, $[\,\langle 1,2\rangle\,]^{\cdot}$, $[\,\bot\,]^{\cdot}$, $[\,1 :: \bot\,]^{\cdot}$, and $[\,1 :: 2 :: \bot\,]^{\cdot}$.

17.10 Finite approximations

You have

$$\begin{aligned}
[\,\infty\text{-list}(1) &\equiv 1 :: \infty\text{-list}(2)\,]^{\cdot}, \\
[\,\infty\text{-list}(1) &\equiv 1 :: 2 :: \infty\text{-list}(3)\,]^{\cdot}, \\
[\,\infty\text{-list}(1) &\equiv 1 :: 2 :: 3 :: \infty\text{-list}(4)\,]^{\cdot},
\end{aligned}$$

edge

and so on. Since $[\, \bot \preceq \mathsf{x} \,]^{\cdot}$ holds for all $[\, \mathsf{x} \,]^{\cdot}$, you have in particular

$$[\, \bot \preceq \infty\text{-list}(\mathsf{n}) \,]^{\cdot}$$

for all $[\, \mathsf{n} \,]^{\cdot}$. In consequence, you have

$$\begin{array}{lll} [\, 1 :: \bot & \preceq & 1 :: \infty\text{-list}(2) \,]^{\cdot} \\ [\, 1 :: 2 :: \bot & \preceq & 1 :: 2 :: \infty\text{-list}(3) \,]^{\cdot} \\ [\, 1 :: 2 :: 3 :: \bot & \preceq & 1 :: 2 :: 3 :: \infty\text{-list}(4) \,]^{\cdot} \end{array}$$

In other words, you have

$$\begin{array}{lll} [\, 1 :: \bot & \preceq & \infty\text{-list}(1) \,]^{\cdot} \\ [\, 1 :: 2 :: \bot & \preceq & \infty\text{-list}(1) \,]^{\cdot} \\ [\, 1 :: 2 :: 3 :: \bot & \preceq & \infty\text{-list}(1) \,]^{\cdot} \end{array}$$

and so on. Here you have a diagram of $[\, \mathsf{x} \preceq \mathsf{y} \,]^{\cdot}$ applied to $[\, \bot \,]^{\cdot}$, $[\, 1 :: \bot \,]^{\cdot}$, $[\, 1 :: 2 :: \bot \,]^{\cdot}$, $[\, 1 :: 2 :: 3 :: \bot \,]^{\cdot}$, and $[\, \infty\text{-list}(1) \,]^{\cdot}$:

$$\begin{array}{c} \infty\text{-list}(1) \\ | \\ 1::2::3::\bot \\ | \\ 1::2::\bot \\ | \\ 1::\bot \\ | \\ \bot \end{array}$$

You can go on and prove $[\, 1 :: 2 :: 3 :: 4 :: \bot \preceq \infty\text{-list}(1) \,]^{\cdot}$, $[\, 1 :: 2 :: 3 :: 4 :: 5 :: \bot \preceq \infty\text{-list}(1) \,]^{\cdot}$, and so on. This leads to an infinitely large diagram that looks like this:

$$\begin{array}{c} \infty\text{-list}(1) \\ | \\ \vdots \\ | \\ 1::2::3::4::5::\bot \\ | \\ 1::2::3::4::\bot \\ | \\ 1::2::3::\bot \\ | \\ 1::2::\bot \\ | \\ 1::\bot \\ | \\ \bot \end{array}$$

As you can see, $[\,\bot\,]^{\cdot}$, $[\,1 :: \bot\,]^{\cdot}$, $[\,1 :: 2 :: \bot\,]^{\cdot}$, $[\,1 :: 2 :: 3 :: \bot\,]^{\cdot}$, and so on form an *increasing sequence* of values. $[\,\infty\text{-list}(1)\,]^{\cdot}$ is said to be the *supremum* of the sequence. The values $[\,\bot\,]^{\cdot}$, $[\,1 :: \bot\,]^{\cdot}$, $[\,1 :: 2 :: \bot\,]^{\cdot}$, $[\,1 :: 2 :: 3 :: \bot\,]^{\cdot}$, and so on are known as *finite approximations* of $[\,\infty\text{-list}(1)\,]^{\cdot}$.

In literature on theoretical computer science, you may find other words for these concepts. As an example, it is common practice to use "monotonic sequence" as a synonym for "(weakly) increasing sequence". It is also common practice to say that $[\,\bot\,]^{\cdot}$, $[\,1 :: \bot\,]^{\cdot}$, $[\,1 :: 2 :: \bot\,]^{\cdot}$, $[\,1 :: 2 :: 3 :: \bot\,]^{\cdot}$, and so on is "convergent" and has "limit value" $[\,\infty\text{-list}(1)\,]^{\cdot}$. When using "convergent" and "limit value" this way it is common practice to say that convergence and limit value is "with respect to the Scott-topology induced by $[\,\mathsf{x} \preceq \mathsf{y}\,]^{\cdot}$".

Here you have a very important property of computable functions:

Fact 17.10.1 If $[\,\mathsf{f}, \mathsf{a}_1, \mathsf{a}_2, \mathsf{a}_3, \ldots\,]^{\circ}$ are computable by machine and if

$$[\,\mathsf{a}_1, \mathsf{a}_2, \mathsf{a}_3, \ldots\,]^{\circ}$$

is an increasing sequence with supremum $[\,\mathsf{a}\,]^{\cdot}$, then

$$[\,f\,\text{'}\,\mathsf{a}_1, f\,\text{'}\,\mathsf{a}_2, f\,\text{'}\,\mathsf{a}_3, \ldots\,]^{\circ}$$

is an increasing sequence with supremum $[\,f\,\text{'}\,\mathsf{a}\,]^{\cdot}$.

The property is known as *continuity*. In the literature you may see the property described as "continuity with respect to the Scott-topology".

In the formulation above, I tacitly assume $[\,\mathsf{f}\,]^{\cdot}$, $[\,\mathsf{a}_1\,]^{\cdot}$, and so on to be maps. The fact, however, is a general property of computation and holds for suitable definitions of $[\,\preceq\,]^{\circ}$ for all kinds of computations.

In the fact above, I assume $[\,\mathsf{f}\,]^{\cdot}$, $[\,\mathsf{a}_1\,]^{\cdot}$, and so on to be *computable*. A term $[\,\mathcal{A}\,]^{\cdot}$ denotes a computable map if the term contains no so-called quantifiers. I introduce quantifiers in Chapter 18.

The map $[\,\bot\,]^{\cdot}$ is computable. If you ask a computer to compute $[\,\bot\,]^{\cdot}$, the computer starts computing and continues to compute forever. Being computable does not require the computation to end in finite time.

17.11 Information contents of functions

For all terms $[\,\mathcal{A}\,]^{\cdot}$ you have $[\,\bot \preceq \mathsf{T}\,]^{\cdot}$, $[\,\bot \preceq \lambda \mathsf{x}.\mathcal{A}\,]^{\cdot}$, and $[\,\mathsf{T} \# \lambda \mathsf{x}.\mathcal{A}\,]^{\cdot}$ (c.f. Exercise 17.11.2):

```
T  λx.A
 \ /
  ⊥
```

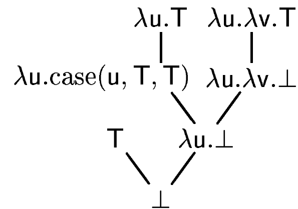
---

increasing sequence
supremum
finite approximations
continuity
computable

If $[\, \mathsf{g} \,]^{\cdot}$ and $[\, \mathsf{h} \,]^{\cdot}$ are functions, then $[\, \mathsf{g} \preceq \mathsf{h} \,]^{\cdot}$ if and only if $[\, \mathsf{g}\,\text{'}\,\mathsf{x} \preceq \mathsf{h}\,\text{'}\,\mathsf{x} \,]^{\cdot}$ for all $[\, \mathsf{x} \,]^{\cdot}$. More formally, you have:

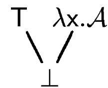
$[$ **Mac rule InfoLambda** $: \mathcal{A} \preceq \mathcal{B} \vdash \lambda \mathsf{x}.\mathcal{A} \preceq \lambda \mathsf{x}.\mathcal{B} \,]$

The diagram below gives some examples.



Exercise 17.11.1 Does the diagram above state that $[\, \mathsf{T} \preceq \lambda \mathsf{u}.\text{case}(\mathsf{u}, \mathsf{T}, \mathsf{T}) \,]^{\cdot}$?

Exercise 17.11.2 Verify



In other words: State $[\, \bot \preceq \mathsf{T} \,]^{\cdot}$, $[\, \bot \preceq \lambda \mathsf{x}.\mathcal{A} \,]^{\cdot}$, and $[\, \mathsf{T} \# \lambda \mathsf{x}.\mathcal{A} \,]^{\cdot}$ as lemmas and prove the lemmas.

17.12 Minimal fixed points

In Section 8.18 I defined $[\, \mathsf{Y} \,]^{\cdot}$ and proved that for any $[\, \mathsf{f} \,]^{\cdot}$,

$$[\, \mathsf{Y}\,\text{'}\,\mathsf{f} \,]^{\cdot}$$

is a solution to

$$[\, \mathsf{f}\,\text{'}\,\mathsf{x} \equiv \mathsf{x} \,]^{\cdot}.$$

I also gave an example where I took $[\, \mathsf{f} \,]^{\cdot}$ to be $[\, \lambda \mathsf{x}.\mathsf{x} + 1 \,]^{\cdot}$. If I take $[\, \mathsf{f} \,]^{\cdot}$ to be $[\, \lambda \mathsf{x}.\mathsf{x} + 1 \,]^{\cdot}$, then

$$[\, \mathsf{f}\,\text{'}\,\mathsf{x} \equiv \mathsf{x} \,]^{\cdot}$$

becomes

$$[\, \mathsf{x} + 1 \equiv \mathsf{x} \,]^{\cdot}.$$

That equation had infinitely many solutions, four of which were:

$$[\, \bot + 1 \equiv \bot \,]^{\cdot},$$

$[\bullet + 1 \equiv \bullet]^{\cdot}$,

$[1\emptyset\mathrm{F} + 1 \equiv 1\emptyset\mathrm{F}]^{\cdot}$, and

$[2\emptyset\mathrm{F} + 1 \equiv 2\emptyset\mathrm{F}]^{\cdot}$.

As I mentioned in Section 8.18, $[\mathsf{Y}\,\text{'}\,\lambda\mathsf{x}.\mathsf{x} + 1]^{\cdot}$ equals $[\bot]^{\cdot}$. In general, if an equation of form

$[\mathsf{f}\,\text{'}\,\mathsf{x} \equiv \mathsf{x}]^{\cdot}$

has more than one solution, then one of the solutions will contain less information than any of the other solutions. I will refer to that solution as the *minimal solution* of $[\mathsf{f}\,\text{'}\,\mathsf{x} \equiv \mathsf{x}]^{\cdot}$ or as the *minimal fixed point* of $[\mathsf{f}]^{\cdot}$. For any $[\mathsf{f}]^{\cdot}$, $[\mathsf{Y}\,\text{'}\,\mathsf{f}]^{\cdot}$ computes the minimal fixed point of $[\mathsf{f}]^{\cdot}$:

[**Mac rule MinimalY** : $\mathsf{f}\,\text{'}\,\mathsf{x} \equiv \mathsf{x} \vdash \mathsf{Y}\,\text{'}\,\mathsf{f} \preceq \mathsf{x}$]

Actually, $[\mathsf{f}\,\text{'}\,\mathsf{x} \preceq \mathsf{x}]^{\cdot}$ is enough to conclude $[\mathsf{Y}\,\text{'}\,\mathsf{f} \preceq \mathsf{x}]^{\cdot}$:

[**Mac rule Minimal** : $\mathsf{f}\,\text{'}\,\mathsf{x} \preceq \mathsf{x} \vdash \mathsf{Y}\,\text{'}\,\mathsf{f} \preceq \mathsf{x}$]

17.13 Uses of minimality

You can use the minimality of $[\mathsf{Y}]^{\cdot}$ to prove lemmas like the following:

[**Mac lemma L17.13.1** : $\mathsf{Y}\,\text{'}\,\lambda\mathsf{x}.\mathsf{x} + 1 \equiv \bot$]

I first prove an auxiliary lemma:

[**Mac lemma InfoLessBottom** : $\mathsf{x} \preceq \bot \vdash \mathsf{x} \equiv \bot$]

[**Mac proof of** InfoLessBottom**:**

L1 :	Premise ▷	$\mathsf{x} \preceq \bot$	;
L2 :	InfoBottom ▷	$\bot \preceq \mathsf{x}$	;
L3 :	InfoAntiSymmetry ▷ L1 ▷ L2 ▷	$\mathsf{x} \equiv \bot$	]

Then I prove L17.13.1:

[**Mac proof of** L17.13.1**:**

L1 :	ApplyLambda ▷	$(\lambda\mathsf{x}.\mathsf{x} + 1)\,\text{'}\,\bot \equiv \bot + 1$	;
L2 :	StrictPlusX ▷	$\bot + 1 \equiv \bot$	;
L3 :	Transitivity ▷ L1 ▷ L2 ▷	$(\lambda\mathsf{x}.\mathsf{x} + 1)\,\text{'}\,\bot \equiv \bot$	;
L4 :	MinimalY ▷ L3 ▷	$\mathsf{Y}\,\text{'}\,\lambda\mathsf{x}.\mathsf{x} + 1 \preceq \bot$	;
L5 :	InfoLessBottom ▷ L4 ▷	$\mathsf{Y}\,\text{'}\,\lambda\mathsf{x}.\mathsf{x} + 1 \equiv \bot$	]

If you ever have to prove that some computation never ends, you should consider to use [Minimal]˙ or [MinimalY]˙.

---

minimal solution
minimal fixed point
[InfoLessBottom]˙ rule InfoLessBottom

17.14 Minimality of definitions

I now give another example. I define

$$\left[\ \boxed{f_7(\mathsf{x})} \doteq \mathrm{case}(\mathsf{x}, f_7(\mathsf{T}), \mathsf{T})\ \right], \text{ and}$$

$$\left[\ \boxed{f_8} \doteq \mathsf{Y}\ ’\ \lambda\mathsf{f}.\lambda\mathsf{x}.\mathrm{case}(\mathsf{x}, \mathsf{f}\ ’\ \mathsf{T}, \mathsf{T})\ \right].$$

Map defines $[\ f_7(\mathsf{x})\ ]^{\cdot}$ to be the minimal construct that satisfies

$$[\ f_7(\mathsf{x}) \equiv \mathrm{case}(\mathsf{x}, f_7(\mathsf{T}), \mathsf{T})\ ]^{\cdot}.$$

In other words, Map defines $[\ f_7(\mathsf{x})\ ]^{\cdot}$ such that

$$[\ f_7(\mathsf{x}) \equiv f_8\ ’\ \mathsf{x}\ ]^{\cdot}.$$

$[\ f_8\ ]^{\cdot}$ is more convenient than $[\ f_7(\mathsf{x})\ ]^{\cdot}$ in connection with applications of minimality. As an example, minimality allows to prove $[\ f_8\ ’\ \mathsf{T} \equiv \bot\ ]^{\cdot}$, which essentially says $[\ f_7(\mathsf{T}) \equiv \bot\ ]^{\cdot}$. To prove $[\ f_8\ ’\ \mathsf{T} \equiv \bot\ ]^{\cdot}$ I first prove $[\ f_8 \preceq \lambda\mathsf{x}.\mathrm{case}(\mathsf{x}, \bot, \mathsf{T})\ ]^{\cdot}$:

[**Mac lemma L17.14.1** : $f_8 \preceq \lambda\mathsf{x}.\mathrm{case}(\mathsf{x}, \bot, \mathsf{T})$]

[**Mac proof of** L17.14.1**:**

L1 :	Block ▷	Begin	;
L2 :	Algebra ▷	$(\lambda\mathsf{f}.\lambda\mathsf{x}.\mathrm{case}(\mathsf{x}, \mathsf{f}\ ’\ \mathsf{T}, \mathsf{T}))\ ’$ $\lambda\mathsf{x}.\mathrm{case}(\mathsf{x}, \bot, \mathsf{T})$	;
L3 :	ApplyLambda ▷	$\lambda\mathsf{x}.\mathrm{case}(\mathsf{x}, (\lambda\mathsf{u}.\mathrm{case}(\mathsf{u}, \bot, \mathsf{T}))\ ’\ \mathsf{T}, \mathsf{T})$	;
L4 :	Replace ▷ ApplyLambda ▷	$\lambda\mathsf{x}.\mathrm{case}(\mathsf{x}, \mathrm{case}(\mathsf{T}, \bot, \mathsf{T}), \mathsf{T})$	;
L5 :	Replace ▷ CaseT ▷	$\lambda\mathsf{x}.\mathrm{case}(\mathsf{x}, \bot, \mathsf{T})$	;
L6 :	Block ▷	End	;
L7 :	MinimalY ▷ L5 ▷	$\mathsf{Y}\ ’\ \lambda\mathsf{f}.\lambda\mathsf{x}.\mathrm{case}(\mathsf{x}, \mathsf{f}\ ’\ \mathsf{T}, \mathsf{T}) \preceq$ $\lambda\mathsf{x}.\mathrm{case}(\mathsf{x}, \bot, \mathsf{T})$	;
L8 :	Definition ▷	$f_8 \equiv \mathsf{Y}\ ’\ \lambda\mathsf{f}.\lambda\mathsf{x}.\mathrm{case}(\mathsf{x}, \mathsf{f}\ ’\ \mathsf{T}, \mathsf{T})$	;
L9 :	Reverse’ ▷ L8 ▷ L7 ▷	$f_8 \preceq \lambda\mathsf{x}.\mathrm{case}(\mathsf{x}, \bot, \mathsf{T})$	]

The lemma above allow to prove $[\ f_8\ ’\ \mathsf{T} \equiv \bot\ ]^{\cdot}$:

[**Mac lemma L17.14.2** : $f_8\ ’\ \mathsf{T} \equiv \bot$]

$[\ f_7(\mathsf{x})\ ]^{\cdot}$	f seven of x end
$[\ f_8\ ]^{\cdot}$	f eight

[**Mac proof of** L17.14.2:

L1 :	L17.14.1 $\rhd$	$f_8 \preceq \lambda \mathsf{x}.\mathrm{case}(\mathsf{x}, \bot, \mathsf{T})$	;
L2 :	Monotonicity' $\rhd$ L1 $\rhd$	f_8 ' $\mathsf{T} \preceq (\lambda \mathsf{x}.\mathrm{case}(\mathsf{x}, \bot, \mathsf{T}))$ ' T	;
L3 :	Block $\rhd$	Begin	;
L4 :	Algebra $\rhd$	$(\lambda \mathsf{x}.\mathrm{case}(\mathsf{x}, \bot, \mathsf{T}))$ ' T	;
L5 :	ApplyLambda $\rhd$	$\mathrm{case}(\mathsf{T}, \bot, \mathsf{T})$	;
L6 :	CaseT $\rhd$	$\bot$	;
L7 :	Block $\rhd$	End	;
L8 :	Replace' $\rhd$ L6 $\rhd$ L2 $\rhd$	f_8 ' $\mathsf{T} \preceq \bot$	;
L9 :	InfoLessBottom $\rhd$ L8 $\rhd$	f_8 ' $\mathsf{T} \equiv \bot$	]

The first example you saw of a value that was equal to $[\,\bot\,]^{\cdot}$ was $[\,(-1)!\,]^{\cdot}$. A proof of $[\,(-1)! \equiv \bot\,]^{\cdot}$ requires use of minimality, a detailed knowledge of the definition of constructs like $[\,\mathsf{x}\cdot\mathsf{y}\,]^{\cdot}$, and a rule that says how Map translates recursive definitions into definitions that use $[\,\mathsf{Y}\,]^{\cdot}$. I will not enter such technicalities here.

Exercise 17.14.3 I define

$[\, f_{10} \doteq \mathsf{x} + f_{10}(\mathsf{x}+1) \,]$, and

$[\, f_{11} \doteq \mathsf{Y}$ ' $\lambda \mathsf{f}.\lambda \mathsf{x}.\mathsf{x} + \mathsf{f}$ ' $(\mathsf{x}+1) \,]$.

You have $[\, f_{10}(\mathsf{x}) \equiv f_{11}$ ' $\mathsf{x} \,]^{\cdot}$, but $[\, f_{11} \,]^{\cdot}$ is easier to work with than $[\, f_{10}(\mathsf{x}) \,]^{\cdot}$ in connection with minimality, so I use $[\, f_{11} \,]^{\cdot}$ in this exercise: Prove $[\, f_{11}$ '$0 \equiv \bot \,]^{\cdot}$.

17.15 $[\,\mathsf{x} \equiv \mathsf{y}\,]^{\cdot}$ is a directive

I will now give an indication of why I have chosen to make $[\,\mathsf{x} \equiv \mathsf{y}\,]^{\cdot}$ a directive. First consider the following definition:

$[\, a \doteq a + 1 \,]$.

Computation of $[\, a \,]^{\cdot}$ goes thus:

$$\begin{array}{lll} a & \stackrel{+}{\rightarrow} & a+1 \\ & \stackrel{+}{\rightarrow} & a+1+1 \\ & \stackrel{+}{\rightarrow} & a+1+1+1 \\ & \stackrel{+}{\rightarrow} & \ldots \end{array}$$

As you can see, the computation never ends, so $[\, a \equiv \bot \,]^{\cdot}$.

Now suppose $[\,\mathsf{x} \equiv \mathsf{y}\,]^{\cdot}$ were an operator and suppose $[\,\mathsf{x} \equiv \mathsf{y}\,]^{\cdot}$ equals $[\,\mathsf{T}\,]^{\cdot}$ if $[\,\mathsf{x}\,]^{\cdot}$ equals $[\,\mathsf{y}\,]^{\cdot}$. Then consider the following definition:

$[\, b \doteq b \equiv \bot \,]^{\circ}$.

Computation of $[\, b \,]^{\cdot}$ goes thus:

$$\begin{array}{lll} b & \stackrel{+}{\to} & b \equiv \bot \\ & \stackrel{+}{\to} & (b \equiv \bot) \equiv \bot \\ & \stackrel{+}{\to} & (b \equiv \bot) \equiv \bot) \equiv \bot \\ & \stackrel{+}{\to} & \ldots \end{array}$$

As you can see, the computation never ends, so $[\, b \equiv \bot \,]^{\circ}$. From the definition $[\, b \doteq b \equiv \bot \,]^{\circ}$ you have

$$[\, b \equiv (b \equiv \bot) \,]^{\circ},$$

and from the computation above you have

$$[\, (b \equiv \bot) \equiv \mathsf{T} \,]^{\circ}.$$

From these two you have

$$[\, b \equiv \mathsf{T} \,]^{\circ}.$$

However, you also have

$$[\, b \equiv \bot \,]^{\circ}.$$

This is quite bad since if $[\, b \,]^{\cdot}$ is equal to both $[\, \mathsf{T} \,]^{\cdot}$ and $[\, \bot \,]^{\cdot}$, then $[\, \mathsf{T} \,]^{\cdot}$ must be equal to $[\, \bot \,]^{\cdot}$, and that is definitely not what I wanted. Actually, I have used $[\, \bot \equiv \mathsf{T} \,]^{\cdot}$ as contradiction in the $[\, \mathsf{Mac} \,]^{\cdot}$ system, so if I made $[\, \mathsf{x} \equiv \mathsf{y} \,]^{\cdot}$ a term, then the $[\, \mathsf{Mac} \,]^{\cdot}$ system would be inconsistent.

But let me continue with this thought experiment to the bitter end. I have assumed for a while that $[\, \mathsf{x} \equiv \mathsf{y} \,]^{\cdot}$ is a term, and $[\, \bot \equiv \mathsf{T} \,]^{\cdot}$ is the first, unexpected consequence. Now consider the following:

$[\, 117 \equiv \mathrm{if}(\mathsf{T}, 117, 118) \,]^{\cdot}$ by the properties of $[\, \mathrm{if}(\mathsf{x}, \mathsf{y}, \mathsf{z}) \,]^{\cdot}$,

$[\, \mathrm{if}(\mathsf{T}, 117, 118) \equiv \mathrm{if}(\bot, 117, 118) \,]^{\cdot}$ since $[\, \mathsf{T} \equiv \bot \,]^{\cdot}$, and

$[\, \mathrm{if}(\bot, 117, 118) \equiv \bot \,]^{\cdot}$ by the properties of $[\, \mathrm{if}(\mathsf{x}, \mathsf{y}, \mathsf{z}) \,]^{\cdot}$.

By combination of these three you can see that $[\, 117 \equiv \bot \,]^{\cdot}$. So it is not just $[\, \mathsf{T} \,]^{\cdot}$ that equals $[\, \bot \,]^{\cdot}$. Actually, all numbers are equal to $[\, \bot \,]^{\cdot}$ and, hence, all numbers are equal to each other. Or even worse, any mathematical concept is equal to any other mathematical concept. This of course simplifies mathematics, but it also makes it less useful.

As you can see, the consequences of making $[\, \mathsf{x} \equiv \mathsf{y} \,]^{\cdot}$ an operator are quite alarming. In this book I have chosen the simplest countermeasure I know of, and that is to make $[\, \mathsf{x} \equiv \mathsf{y} \,]^{\cdot}$ a statement.

Another countermeasure could be to ban recursive definitions and the $[\, \lambda \mathsf{x}.\mathsf{y} \,]^{\cdot}$ construct. That is what classical mathematics does with more or less success.

The problem with this approach is that recursive definitions are needed both in pure mathematics and in applications to computer science, so mathematics without recursive definitions is less straightforward than mathematics with recursive definitions.

To sum up: The decision to make $[\, x \equiv y \,]^{\cdot}$ a statement is one I have made for the sake of consistency.

The need to make $[\, x \equiv y \,]^{\cdot}$ a statement has something to do with monotonicity. If $[\, x \equiv y \,]^{\cdot}$ were a term and if $[\, x \equiv y \,]^{\cdot}$ had one value when $[\, x \,]^{\cdot}$ was equal to $[\, y \,]^{\cdot}$ and some other value when $[\, x \,]^{\cdot}$ differed from $[\, y \,]^{\cdot}$, then $[\, x \equiv y \,]^{\cdot}$ would be non-monotonic. In a system like $[\, \mathsf{Mac} \,]^{\cdot}$ that allows free use of $[\, \lambda x.y \,]^{\cdot}$, all operators must be monotonic.

17.16 Answers

Answer 17.1.1 Yes.

Answer 17.2.1 (a) Yes. (b) No, $[\, \bot \,]^{\cdot}$ does not have the $[\, \lambda y.\mathrm{case}(y, \bot, \mathsf{T}) \,]^{\cdot}$ property, so $[\, \lambda x.\bot \,]^{\cdot}$ and $[\, \bot \,]^{\cdot}$ have different properties.

Answer 17.4.1 (a) and (c).

Answer 17.5.1 (c).

Answer 17.6.2

[**Mac lemma L17.16.1** : $\mathsf{T} \preceq \mathsf{F} \vdash \bot$]

[**Mac proof of** L17.16.1**:**

L1 :	Premise $\rhd$	$\mathsf{T} \preceq \mathsf{F}$	;
L2 :	InfoImply $\rhd$ L1 $\rhd$	$f \,'\, \mathsf{T} \to f \,'\, \mathsf{F}$	;
L3 :	Replace $\rhd$ L2 $\rhd$	$(\lambda x.x) \,'\, \mathsf{T} \to (\lambda x.x) \,'\, \mathsf{F}$	;
L4 :	Computation $\rhd$	$(\lambda x.x) \,'\, \mathsf{T}$	;
L5 :	L3 $\unrhd$ L4 $\rhd$	$(\lambda x.x) \,'\, \mathsf{F}$	;
L6 :	Computation $\rhd$	$(\lambda x.x) \,'\, \mathsf{F} \equiv \mathsf{F}$	;
L7 :	CounterTF $\rhd$ L5 $\rhd$ L6 $\rhd$	$\bot$	]

Answer 17.8.3

[**Mac proof of** MonotonicityHead**:**

L1 :	Premise $\rhd$	$x \preceq y$	;
L2 :	Monotonicity' $\rhd$ L1 $\rhd$	x head $\preceq y$ head	]

[**Mac proof of** MonotonicityTail**:**

L1 :	Premise $\rhd$	$x \preceq y$	;
L2 :	Monotonicity' $\rhd$ L1 $\rhd$	x tail $\preceq y$ tail	]

[**Mac proof of** MonotonicityHead'**:**

L1 : Premise ▷ $x :: y \preceq u :: v$;
L2 : MonotonicityHead ▷ L1 ▷ $(x :: y)$ head $\preceq (u :: v)$ head ;
L3 : HeadPair ▷ $(x :: y)$ head $\equiv x$;
L4 : Replace' ▷ L3 ▷ L2 ▷ $x \preceq (u :: v)$ head ;
L5 : HeadPair ▷ $(u :: v)$ head $\equiv u$;
L6 : Replace' ▷ L5 ▷ L4 ▷ $x \preceq u$]

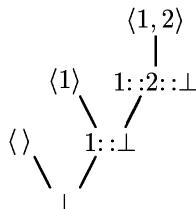
[**Mac proof of** MonotonicityTail'**:**
L1 : Premise ▷ $x :: y \preceq u :: v$;
L2 : MonotonicityHead ▷ L1 ▷ $(x :: y)$ tail $\preceq (u :: v)$ tail ;
L3 : HeadPair ▷ $(x :: y)$ tail $\equiv y$;
L4 : Replace' ▷ L3 ▷ L2 ▷ $y \preceq (u :: v)$ tail ;
L5 : HeadPair ▷ $(u :: v)$ tail $\equiv v$;
L6 : Replace' ▷ L5 ▷ L4 ▷ $y \preceq v$]

[**Mac proof of** MonotonicityHead"**:**
L1 : Premise ▷ $x \preceq y$;
L2 : Monotonicity' ▷ L1 ▷ $x :: z \preceq y :: z$]

[**Mac proof of** MonotonicityTail"**:**
L1 : Premise ▷ $y \preceq z$;
L2 : Monotonicity' ▷ L1 ▷ $x :: y \preceq x :: z$]

[**Mac proof of** MonotonicityPair**:**
L1 : Premise ▷ $x \preceq u$;
L2 : Premise ▷ $y \preceq v$;
L3 : MonotonicityHead" ▷ L1 ▷ $x :: y \preceq u :: y$;
L4 : MonotonicityTail" ▷ L2 ▷ $u :: y \preceq u :: v$;
L5 : InfoTransitivity ▷ L3 ▷ L4 ▷ $x :: y \preceq u :: v$]

Answer 17.9.1



Answer 17.11.1 No. There is no edge from $[\, \mathsf{T} \,]^{\cdot}$ up to $[\, \lambda u.\mathrm{case}(u, \mathsf{T}, \mathsf{T}) \,]^{\cdot}$.
Answer 17.11.2 $[\, \bot \preceq \mathsf{T} \,]^{\cdot}$ and $[\, \bot \preceq \lambda x.\mathcal{A} \,]^{\cdot}$ are easy:

[**Mac lemma L17.16.2** : $\bot \preceq \mathsf{T}$]

[**Mac proof of** L17.16.2**:**
L1 : InfoBottom ▷ $\bot \preceq \mathsf{T}$]

[**Mac lemma L17.16.3** : $\bot \preceq \lambda \mathsf{x}.\mathcal{A}$]

[**Mac proof of** L17.16.3**:**
L1 : InfoBottom ▷ $\bot \preceq \lambda \mathsf{x}.\mathcal{A}$]

[$\mathsf{T} \# \lambda \mathsf{x}.\mathcal{A}$] requires two lemmas:

[**Mac lemma L17.16.4** : $\mathsf{T} \preceq \lambda \mathsf{x}.\mathcal{A} \vdash \bot$]

[**Mac proof of** L17.16.4**:**

L01 :	Premise ▷	$\mathsf{T} \preceq \lambda \mathsf{x}.\mathcal{A}$	;
L02 :	InfoImply ▷ L1 ▷	$(\lambda \mathsf{u}.\text{case}(\mathsf{u}, \mathsf{T}, \mathsf{F}))\text{ ' }\mathsf{T} \rightarrow$ $(\lambda \mathsf{u}.\text{case}(\mathsf{u}, \mathsf{T}, \mathsf{F}))\text{ ' }\lambda \mathsf{x}.\mathcal{A}$	;
L03 :	ApplyLambda ▷	$(\lambda \mathsf{u}.\text{case}(\mathsf{u}, \mathsf{T}, \mathsf{F}))\text{ ' }\mathsf{T} \equiv \text{case}(\mathsf{T}, \mathsf{T}, \mathsf{F})$	;
L04 :	CaseT ▷	$\text{case}(\mathsf{T}, \mathsf{T}, \mathsf{F}) \equiv \mathsf{T}$	;
L05 :	Transitivity ▷ L3 ▷ L4 ▷	$(\lambda \mathsf{u}.\text{case}(\mathsf{u}, \mathsf{T}, \mathsf{F}))\text{ ' }\mathsf{T} \equiv \mathsf{T}$	;
L06 :	ApplyLambda ▷	$(\lambda \mathsf{u}.\text{case}(\mathsf{u}, \mathsf{T}, \mathsf{F}))\text{ ' }\lambda \mathsf{x}.\mathcal{A} \equiv$ $\text{case}(\lambda \mathsf{x}.\mathcal{A}, \mathsf{T}, \mathsf{F})$	;
L07 :	CaseLambda ▷	$\text{case}(\lambda \mathsf{x}.\mathcal{A}, \mathsf{T}, \mathsf{F}) \equiv \mathsf{F}$	;
L08 :	Transitivity ▷ L6 ▷ L7 ▷	$(\lambda \mathsf{u}.\text{case}(\mathsf{u}, \mathsf{T}, \mathsf{F}))\text{ ' }\lambda \mathsf{x}.\mathcal{A} \equiv \mathsf{F}$	;
L09 :	Replace' ▷ L5 ▷ L2 ▷	$\mathsf{T} \rightarrow (\lambda \mathsf{u}.\text{case}(\mathsf{u}, \mathsf{T}, \mathsf{F}))\text{ ' }\lambda \mathsf{x}.\mathcal{A}$	;
L10 :	Replace' ▷ L8 ▷ L9 ▷	$\mathsf{T} \rightarrow \mathsf{F}$	;
L11 :	Reflexivity ▷	T	;
L12 :	L10 ⊵ L11 ▷	F	;
L13 :	Commutativity ▷ L12 ▷	$\mathsf{T} \equiv \mathsf{F}$	;
L14 :	TUnequalF ▷	$\bot$	]

[**Mac lemma L17.16.5** : $\lambda \mathsf{x}.\mathcal{A} \preceq \mathsf{T} \vdash \bot$]

[**Mac proof of** L17.16.5**:**

L01 :	Premise ▷	$\lambda \mathsf{x}.\mathcal{A} \preceq \mathsf{T}$	;
L02 :	InfoImply ▷ L1 ▷	$(\lambda \mathsf{u}.\text{case}(\mathsf{u}, \mathsf{F}, \mathsf{T}))\text{ ' }\lambda \mathsf{x}.\mathcal{A} \rightarrow$ $(\lambda \mathsf{u}.\text{case}(\mathsf{u}, \mathsf{F}, \mathsf{T}))\text{ ' }\mathsf{T}$	;
L03 :	ApplyLambda ▷	$(\lambda \mathsf{u}.\text{case}(\mathsf{u}, \mathsf{F}, \mathsf{T}))\text{ ' }\mathsf{T} \equiv \text{case}(\mathsf{T}, \mathsf{F}, \mathsf{T})$	;
L04 :	CaseT ▷	$\text{case}(\mathsf{T}, \mathsf{F}, \mathsf{T}) \equiv \mathsf{F}$	;
L05 :	Transitivity ▷ L3 ▷ L4 ▷	$(\lambda \mathsf{u}.\text{case}(\mathsf{u}, \mathsf{T}, \mathsf{F}))\text{ ' }\mathsf{T} \equiv \mathsf{F}$	;
L06 :	ApplyLambda ▷	$(\lambda \mathsf{u}.\text{case}(\mathsf{u}, \mathsf{F}, \mathsf{T}))\text{ ' }\lambda \mathsf{x}.\mathcal{A} \equiv$ $\text{case}(\lambda \mathsf{x}.\mathcal{A}, \mathsf{F}, \mathsf{T})$	;
L07 :	CaseLambda ▷	$\text{case}(\lambda \mathsf{x}.\mathcal{A}, \mathsf{F}, \mathsf{T}) \equiv \mathsf{T}$	;
L08 :	Transitivity ▷ L6 ▷ L7 ▷	$(\lambda \mathsf{u}.\text{case}(\mathsf{u}, \mathsf{F}, \mathsf{T}))\text{ ' }\lambda \mathsf{x}.\mathcal{A} \equiv \mathsf{T}$	;
L09 :	Replace' ▷ L8 ▷ L2 ▷	$\mathsf{T} \rightarrow (\lambda \mathsf{u}.\text{case}(\mathsf{u}, \mathsf{T}, \mathsf{F}))\text{ ' }\lambda \mathsf{x}.\mathcal{A}$	;
L10 :	Replace' ▷ L5 ▷ L9 ▷	$\mathsf{T} \rightarrow \mathsf{F}$	;
L11 :	Reflexivity ▷	T	;
L12 :	L10 ⊵ L11 ▷	F	;
L13 :	Commutativity ▷ L12 ▷	$\mathsf{T} \equiv \mathsf{F}$	;
L14 :	TUnequalF ▷	$\bot$	]

Answer 17.14.3

[**Mac lemma L17.16.6** : $f_{11} \preceq \lambda x.\bot$]

[**Mac proof of** L17.16.6**:**

L1 :	Block ▷	Begin	;
L2 :	Algebra ▷	$(\lambda f.\lambda x.x + f\text{'}(x+1))\text{'}\lambda x.\bot$	;
L3 :	ApplyLambda ▷	$\lambda x.x + (\lambda x.\bot)\text{'}(x+1)$	;
L4 :	Replace ▷ ApplyLambda ▷	$\lambda x.x + \bot$	;
L5 :	Replace ▷ XPlusStrict ▷	$\lambda x.\bot$	;
L6 :	Block ▷	End	;
L7 :	MinimalY ▷ L5 ▷	$Y\text{'}\lambda f.\lambda x.x + f\text{'}(x+1) \preceq \lambda x.\bot$	;
L8 :	Definition ▷	$f_{11} \equiv Y\text{'}\lambda f.\lambda x.x + f\text{'}(x+1)$	;
L9 :	Reverse' ▷ L8 ▷ L7 ▷	$f_{11} \preceq \lambda x.\bot$	]

[**Mac lemma L17.16.7** : $f_{11}\text{'}0 \equiv \bot$]

[**Mac proof of** L17.16.7**:**

L1 :	L17.16.6 ▷	$f_{11} \preceq \lambda x.\bot$	;
L2 :	Monotonicity' ▷ L1 ▷	$f_{11}\text{'}0 \preceq (\lambda x.\bot)\text{'}0$	;
L3 :	ApplyLambda ▷	$(\lambda x.\bot)\text{'}0 \equiv \bot$	;
L4 :	Replace' ▷ L3 ▷ L2 ▷	$f_{11}\text{'}0 \preceq \bot$	;
L5 :	InfoLessBottom ▷ L4 ▷	$f_{11}\text{'}0 \equiv \bot$	]

Chapter 18

Quantifiers

18.1 Universal quantification

If $[\,\mathsf{S}\,]^{\cdot}$ is a set then $\left[\,\boxed{\forall \mathsf{x}{\in}\mathsf{S}{:}\,\mathcal{A}}\,\right]^{\cdot}$ has the following properties:

$[\,\forall \mathsf{x}{\in}\mathsf{S}{:}\,\mathcal{A} \equiv \mathsf{T}\,]^{\cdot}$	if $[\,\mathcal{A} \in \mathbf{T}\,]^{\cdot}$ for all $[\,\mathsf{x} \in \mathsf{S}\,]^{\cdot}$.
$[\,\forall \mathsf{x}{\in}\mathsf{S}{:}\,\mathcal{A} \equiv \mathsf{F}\,]^{\cdot}$	if $[\,\mathcal{A} \in \mathbf{B}\,]^{\cdot}$ for all $[\,\mathsf{x} \in \mathsf{S}\,]^{\cdot}$ and $[\,\mathcal{A} \in \mathbf{F}\,]^{\cdot}$ for some $[\,\mathsf{x} \in \mathsf{S}\,]^{\cdot}$.
$[\,\forall \mathsf{x}{\in}\mathsf{S}{:}\,\mathcal{A} \equiv \bullet\,]^{\cdot}$	if $[\,\mathcal{A}\,]^{\cdot}$ is classical for all $[\,\mathsf{x} \in \mathsf{S}\,]^{\cdot}$ and $[\,\mathcal{A} \notin \mathbf{B}\,]^{\cdot}$ for some $[\,\mathsf{x} \in \mathsf{S}\,]^{\cdot}$.
$[\,\forall \mathsf{x}{\in}\mathsf{S}{:}\,\mathcal{A} \equiv \bot\,]^{\cdot}$	if $[\,\mathcal{A}\,]^{\cdot}$ is non-classical for some $[\,\mathsf{x} \in \mathsf{S}\,]^{\cdot}$.

The literature refers to $[\,\forall \mathsf{x}{\in}\mathsf{S}{:}\,\mathcal{A}\,]^{\cdot}$ as a *universal quantification*.

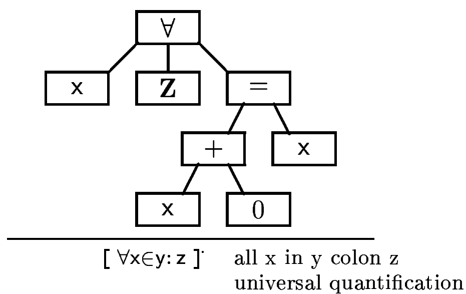
Here are some examples: $[\,\forall \mathsf{x}{\in}\mathbf{Z}{:}\,\mathsf{x}{+}0 = \mathsf{x}\,]^{\cdot}$ equals $[\,\mathsf{T}\,]^{\cdot}$ because $[\,\mathsf{x}{+}0 = \mathsf{x}\,]^{\cdot}$ is true for all integers $[\,\mathsf{x}\,]^{\cdot}$. $[\,\forall \mathsf{x}{\in}\mathbf{N}{:}\,2 \;//\; \mathsf{x} \in \mathbf{N}\,]^{\cdot}$ equals $[\,\mathsf{F}\,]^{\cdot}$ because $[\,2 \;//\; \mathsf{x} \equiv \bullet\,]^{\cdot}$ for $[\,\mathsf{x} \equiv 0\,]^{\cdot}$ and $[\,\bullet \in \mathbf{N}\,]^{\cdot}$ is false.

Exercise 18.1.1 What are the values of the following:

a $[\,\forall \mathsf{x}{\in}\mathbf{N}{:}\,\mathsf{x} \le \mathsf{x}!\,]^{\cdot}$.

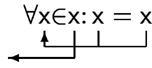
b $[\,\forall \mathsf{x}{\in}\mathbf{Z}{:}\,\mathsf{x} \le \mathsf{x}!\,]^{\cdot}$.

The syntax tree of $[\,\forall \mathsf{x}{\in}\mathbf{Z}{:}\,\mathsf{x} + 0 = \mathsf{x}\,]^{\cdot}$ reads:



$[\,\forall \mathsf{x}{\in}\mathsf{y}{:}\,\mathsf{z}\,]^{\cdot}$ all x in y colon z
universal quantification

The construct $[\,\forall x{\in}S\colon \mathcal{A}\,]^{\cdot}$ binds all free occurrences of $[\,x\,]^{\cdot}$ in $[\,\mathcal{A}\,]^{\cdot}$ but does not bind those in $[\,S\,]^{\cdot}$. As an example, the binding diagram of $[\,\forall x{\in}x\colon x = x\,]^{\cdot}$ is



and $[\,\forall x{\in}x\colon x = x\,]^{\cdot}$ means the same as $[\,\forall y{\in}x\colon y = y\,]^{\cdot}$.

Exercise 18.1.2 What are the values of the following:

a $[\,\langle \forall x{\in}x\colon x! > 0 \mid x{:=}\mathbf{N}\rangle\,]^{\cdot}$.

b $[\,\langle \forall x{\in}x\colon x! > 0 \mid x{:=}\mathbf{Z}\rangle\,]^{\cdot}$.

The universal quantifier has priority between $[\,x \vee y\,]^{\cdot}$ and $[\,x \Rightarrow y\,]^{\cdot}$:

$$[\,x \vee y \mathrel{\check{>}} \forall x{\in}y\colon z \mathrel{\check{>}} x \Rightarrow y\,].$$

The following term reduction rule is very handy:

$$\left[\,\forall x, y{\in}S\colon a \overset{\circ}{\rightarrow} \forall x{\in}S\colon \forall y{\in}S\colon a\,\right].$$

The universal quantifier is not computable by machine. You can prove properties about the universal quantifier, and you can use the universal quantifier in mathematical proofs, but you cannot implement the universal quantifier on a computer. Therefore I will call the universal quantifier *non-computable* as opposed to constructs like $[\,x + y\,]^{\cdot}$ and $[\,x - y\,]^{\cdot}$ which are *computable*.

18.2 The $[\,\mathrm{Gen}\,]^{\cdot}$ rule

In Section 18.1 I stated that if $[\,S\,]^{\cdot}$ is a set then $[\,\forall x{\in}S\colon \mathcal{A} \equiv \mathsf{T}\,]^{\cdot}$ if $[\,\mathcal{A} \in \mathbf{T}\,]^{\cdot}$ for all $[\,x \in S\,]^{\cdot}$. Here you have a formal version of that:

[**Mac rule Gen** : $S \in \mathbf{Set} \vdash x \in S \rightarrow \mathcal{A} \vdash \forall x{\in}S\colon \mathcal{A}$]

The litterature refers to $[\,\mathrm{Gen}\,]^{\cdot}$ as *generalisation*.

Exercise 18.2.1 Prove the following:

[**Mac lemma L18.2.2** : $\forall x \in \mathbf{D}^{\cdot}_{\infty}\colon x + 1 \in \mathbf{D}^{\cdot}_{\infty}$]

non-computable
computable
generalisation

18.3 The [ElimAll] rule

If [S] is a set and if [$\forall x{\in}S{:}\,\mathcal{A}$] is true then [$\mathcal{A}$] is true for all [$\mathcal{B} \in S$]:

[**Mac rule ElimAll** : $S \in \mathbf{Set} \vdash \forall x{\in}S{:}\,\mathcal{A} \vdash \mathcal{B} \in S \vdash \langle \mathcal{A} \mid x{:=}\mathcal{B} \rangle$]

Here you have an example of use:

[**Mac lemma L18.3.1** : $\forall y{\in}\mathbf{Z}{:}\, x > y \to x > 117$]

The lemma says that if [x] is greater than any integer, then [x] is greater than [117] ([x] could be e.g. [$+\infty$]).

[**Mac proof of** L18.3.1:

L1 :	SetZ ▷	$\mathbf{Z} \in \mathbf{Set}$	;
L2 :	Block ▷	Begin	;
L3 :	Hypothesis ▷	$\forall y{\in}\mathbf{Z}{:}\, x > y$	;
L4 :	TypeNumeralInZ ▷	$117 \in \mathbf{Z}$	;
L5 :	ElimAll ▷ L1 ▷ L3 ▷ L4 ▷	$x > 117$	;
L6 :	Block ▷	End	]

Exercise 18.3.2 Prove the following:

[**Mac lemma L18.3.3** : $\forall y{\in}\mathbf{Z}{:}\, x > y \to \forall y{\in}\mathbf{Z}{:}\, x > y + 1$]

18.4 Existential quantification

If [S] is a set then [$\boxed{\exists x{\in}S{:}\,\mathcal{A}}$] has the following properties:

[$\exists x{\in}S{:}\,\mathcal{A} \equiv F$]	if [$\mathcal{A} \in \mathbf{F}$] for all [$x \in S$].
[$\exists x{\in}S{:}\,\mathcal{A} \equiv T$]	if [$\mathcal{A} \in \mathbf{B}$] for all [$x \in S$] and [$\mathcal{A} \in \mathbf{T}$] for some [$x \in S$].
[$\exists x{\in}S{:}\,\mathcal{A} \equiv \bullet$]	if [$\mathcal{A}$] is classical for all [$x \in S$] and [$\mathcal{A} \notin \mathbf{B}$] for some [$x \in S$].
[$\exists x{\in}S{:}\,\mathcal{A} \equiv \bot$]	if [$\mathcal{A}$] is non-classical for some [$x \in S$].

The literature refers to [$\exists x{\in}S{:}\,\mathcal{A}$] as an *existential quantification*.

Here are some examples: [$\exists x{\in}\mathbf{Z}{:}\, x^2 = 4$] equals [T] because [$(x^2 = 4) \in \mathbf{B}$] for all integers [x] and, among other, [$x^2 = 4$] is true for [$x \equiv 2$]. [$\exists x{\in}\mathbf{Z}{:}\, x^2 = 3$] equals [F] because [$x^2 = 3$] is false for all integers [x].

The existential quantifier has the same priority as the universal quantifier:

[$\exists x{\in}S{:}\,\mathcal{A} \doteq \forall x{\in}S{:}\,\mathcal{A}$].

[$\exists x{\in}y{:}\, z$] exists x in y colon z
existential quantification

18.5 The $[\,\mathrm{IntroExists}\,]^{\cdot}$ rule

Here you have a law about the existential quantifier.

$[$ **Mac rule IntroExists** : $\mathsf{S} \in \mathbf{Set} \vdash \mathsf{x} \in \mathsf{S} \to \mathcal{A} \in \mathbf{B} \vdash \mathcal{B} \in \mathsf{S} \vdash \langle \mathcal{A} \mid \mathsf{x}{:=}\mathcal{B} \rangle \vdash \exists \mathsf{x}{\in}\mathsf{S}\colon \mathcal{A}\ ]$

The premise

$$[\,\mathsf{x} \in \mathsf{S} \to \mathcal{A} \in \mathbf{B}\,]^{\cdot}$$

ensures that $[\,\exists \mathsf{x}{\in}\mathsf{S}\colon \mathcal{A}\,]^{\cdot}$ equals $[\,\mathsf{T}\,]^{\cdot}$ or $[\,\mathsf{F}\,]^{\cdot}$ (rather than $[\,\bullet\,]^{\cdot}$ or $[\,\bot\,]^{\cdot}$). The third and fourth premise say that $[\,\mathcal{B}\,]^{\cdot}$ is a member of $[\,\mathsf{S}\,]^{\cdot}$ and that $[\,\mathcal{A}\,]^{\cdot}$ is true when $[\,\mathsf{x} \equiv \mathcal{B}\,]^{\cdot}$. The conclusion says that there exists an $[\,\mathsf{x}\,]^{\cdot}$ in $[\,\mathsf{S}\,]^{\cdot}$ such that $[\,\mathcal{A}\,]^{\cdot}$ is true. Here you have an example of use:

$[$ **Mac lemma L18.5.1** : $\exists \mathsf{x}{\in}\mathbf{Z}\colon \mathsf{x} \cdot \mathsf{x} = 4\ ]$

$[$ **Mac proof of** L18.5.1**:**

L00 :	SetZ $\rhd$	$\mathbf{Z} \in \mathbf{Set}$	;
L01 :	Block $\rhd$	Begin	;
L02 :	Hypothesis $\rhd$	$\mathsf{x} \in \mathbf{Z}$	;
L03 :	Type$\mathbf{Z}{\cdot}\mathbf{Z}$ $\rhd$ L2 $\rhd$ L2 $\rhd$	$\mathsf{x} \cdot \mathsf{x} \in \mathbf{Z}$	;
L04 :	TypeNumeralInZ $\rhd$	$4 \in \mathbf{Z}$	;
L05 :	Type$\mathbf{Z}{=}\mathbf{Z}$ $\rhd$ L3 $\rhd$ L4 $\rhd$	$(\mathsf{x} \cdot \mathsf{y} = 4) \in \mathbf{B}$	;
L06 :	Block $\rhd$	End	;
L07 :	TypeNumeralInZ $\rhd$	$2 \in \mathbf{Z}$	;
L08 :	Computation $\rhd$	$2 \cdot 2 = 4$	;
L09 :	Substitution $\rhd$ L8 $\rhd$	$\langle \mathsf{x} \cdot \mathsf{x} = 4 \mid \mathsf{x}{:=}2 \rangle$	;
L10 :	IntroExists $\rhd$ L0 $\rhd$ L5 $\rhd$ L7 $\rhd$ L9 $\rhd$	$\exists \mathsf{x}{\in}\mathbf{Z}\colon \mathsf{x} \cdot \mathsf{x} = 4$	]

In the proof above I proved $[\,\exists \mathsf{x}{\in}\mathbf{Z}\colon \mathsf{x} \cdot \mathsf{x} = 4\,]^{\cdot}$ by giving the example $[\,\mathsf{x} \equiv 2\,]^{\cdot}$. The example $[\,\mathsf{x} \equiv -2\,]^{\cdot}$ is just as good:

$[$ **Mac proof of** L18.5.1**:**

L00 :	SetZ $\rhd$	$\mathbf{Z} \in \mathbf{Set}$	;
L01 :	Block $\rhd$	Begin	;
L02 :	Hypothesis $\rhd$	$\mathsf{x} \in \mathbf{Z}$	;
L03 :	Type$\mathbf{Z}{\cdot}\mathbf{Z}$ $\rhd$ L2 $\rhd$ L2 $\rhd$	$\mathsf{x} \cdot \mathsf{x} \in \mathbf{Z}$	;
L04 :	TypeNumeralInZ $\rhd$	$4 \in \mathbf{Z}$	;
L05 :	Type$\mathbf{Z}{=}\mathbf{Z}$ $\rhd$ L3 $\rhd$ L4 $\rhd$	$\mathsf{x} \cdot \mathsf{y} = 4 \in \mathbf{B}$	;
L06 :	Block $\rhd$	End	;
L07 :	TypeNumeralInZ $\rhd$	$-2 \in \mathbf{Z}$	;
L08 :	Computation $\rhd$	$(-2) \cdot -2 = 4$	;
L09 :	Substitution $\rhd$ L8 $\rhd$	$\langle \mathsf{x} \cdot \mathsf{x} = 4 \mid \mathsf{x}{:=}{-2} \rangle$	;
L10 :	IntroExists $\rhd$ L0 $\rhd$ L5 $\rhd$ L7 $\rhd$ L9 $\rhd$	$\exists \mathsf{x}{\in}\mathbf{Z}\colon \mathsf{x} \cdot \mathsf{x} = 4$	]

18.6 Local definitions

Here you have a proof that uses a *local definition*:

[**Mac lemma L18.5.1** : $\exists x{\in}\mathbf{Z}{:}\, x \cdot x = 4$]

[**Mac proof of** L18.5.1:

L00 :	SetZ ⊳	$\mathbf{Z} \in \mathbf{Set}$	;
L01 :	Block ⊳	Begin	;
L02 :	Hypothesis ⊳	$x \in \mathbf{Z}$	;
L03 :	Type**Z**·**Z** ⊳ L2 ⊳ L2 ⊳	$x \cdot x \in \mathbf{Z}$	;
L04 :	TypeNumeralInZ ⊳	$4 \in \mathbf{Z}$	;
L05 :	Type**Z**=**Z** ⊳ L3 ⊳ L4 ⊳	$x \cdot x = 4 \in \mathbf{B}$	;
L06 :	Block ⊳	End	;
L07 :	Local ⊳	$x \equiv 2$	;
L08 :	TypeNumeralInZ ⊳	$x \in \mathbf{Z}$	;
L09 :	Computation ⊳	$x \cdot x = 4$	;
L10 :	IntroExists ⊳ L0 ⊳ L5 ⊳ L8 ⊳ L9 ⊳	$\exists x{\in}\mathbf{Z}{:}\, x \cdot x = 4$	]

Line [L7] makes a local definition. The local definition has the effect that all free occurrences of [x] from Line [L7] to Line [L10] are shorthand for [2].

Note that in Line [L8] I use [TypeNumeralInZ] to prove [$x \in \mathbf{Z}$]. That is legal because [x] is shorthand for the numeral [2]. In general, when verifying a proof one must replace locally defined variables by their definitions before checking that rules are applied correctly.

In general, a local definition is an equation whose left hand side is a variable. The right hand side can be any term.

A local definition has effect from the line where it is made and until the end of the smallest enclosing block, or until the end of the proof if there is no enclosing blocks.

A proof may contain more than one local definition of the same variable. If more than one local definition of the same variable is in effect at any one line, then the latest of the definitions count.

Local definitions have no effect on bound variables. As an example, the local definition of [x] in Line [L7] has no effect on the occurrences of [x] in Line [L10] because the occurrences of [x] in Line [L10] are not free.

A local definition like [$x \equiv T :: x$] where the defined variable occurs in the right hand side of the definition is a local, recursive definition. A local definition like [$x \equiv T :: x$] introduces [x] as shorthand for [$Y \,'\, \lambda x.T :: x$].

18.7 Choice

[$\boxed{\varepsilon x{\in}S{:}\, \mathcal{A}}$] is a construct that is quite different from any of the constructs you have seen so far. [$\varepsilon x{\in}S{:}\, \mathcal{A}$] is known as *Hilberts epsilon operator* or as the

[$\varepsilon x{\in}y{:}\, z$] local definition
choose x in y colon z
Hilberts epsilon operator

choice operator.

If $[\,\mathsf{S}\,]^{\cdot}$ is a set, if $[\,\mathcal{A} \in \mathbf{B}\,]^{\cdot}$ for all $[\,\mathsf{x} \in \mathsf{S}\,]^{\cdot}$, and if $[\,\mathcal{A} \in \mathbf{T}\,]^{\cdot}$ for some $[\,\mathsf{x} \in \mathsf{S}\,]^{\cdot}$, then $[\,\varepsilon\mathsf{x}{\in}\mathsf{S}\colon \mathcal{A}\,]^{\cdot}$ equals some $[\,\mathsf{x}\,]^{\cdot}$ such that $[\,\mathsf{x} \in \mathsf{S}\,]^{\cdot}$ and $[\,\mathcal{A}\,]^{\cdot}$ are true. Here you have an example:

$$[\,(\varepsilon\mathsf{x}{\in}\mathbf{N}\colon 2\cdot\mathsf{x}+3=23)=10\,]^{\cdot}.$$

In the example above, the epsilon operator finds the natural number $[\,\mathsf{x}\,]^{\cdot}$ for which $[\,2\cdot\mathsf{x}+3=23\,]^{\cdot}$.

If $[\,\mathsf{x} \in \mathsf{S}\,]^{\cdot}$ and $[\,\mathcal{A}\,]^{\cdot}$ hold for more than one $[\,\mathsf{x}\,]^{\cdot}$, then $[\,\varepsilon\mathsf{x}{\in}\mathsf{S}\colon \mathcal{A}\,]^{\cdot}$ selects one of them at random. As an example,

$$[\,\varepsilon\mathsf{x}{\in}\mathbf{Z}\colon \mathsf{x}^2=9\,]^{\cdot}$$

may be $[\,3\,]^{\cdot}$ or $[\,-3\,]^{\cdot}$. I cannot tell whether $[\,\varepsilon\mathsf{x}{\in}\mathbf{Z}\colon \mathsf{x}^2=9\,]^{\cdot}$ equals $[\,3\,]^{\cdot}$ or $[\,-3\,]^{\cdot}$, but I know that one of them holds:

$$[\,(\varepsilon\mathsf{x}{\in}\mathbf{Z}\colon \mathsf{x}^2=9)=3 \vee (\varepsilon\mathsf{x}{\in}\mathbf{Z}\colon \mathsf{x}^2=9)=-3\,]^{\cdot}.$$

However, $[\,\varepsilon\mathsf{x}{\in}\mathsf{S}\colon \mathcal{A}\,]^{\cdot}$ is not completely random: it consistently chooses the same $[\,\mathsf{x}\,]^{\cdot}$ each time it is applied to the same $[\,\mathsf{S}\,]^{\cdot}$ and $[\,\mathcal{A}\,]^{\cdot}$. Hence, $[\,\varepsilon\mathsf{x}{\in}\mathbf{Z}\colon \mathsf{x}^2=9\,]^{\cdot}$ is either always $[\,3\,]^{\cdot}$ or always $[\,-3\,]^{\cdot}$. It cannot be $[\,3\,]^{\cdot}$ sometimes and $[\,-3\,]^{\cdot}$ other times. As a consequence you have

$$[\,\varepsilon\mathsf{x}{\in}\mathbf{Z}\colon \mathsf{x}^2=9 \equiv \varepsilon\mathsf{x}{\in}\mathbf{Z}\colon \mathsf{x}^2=9\,]^{\cdot}$$

because the two occurrences of the epsilon operator choose the same number. This is fairly important, for otherwise the algebraic rule $[\,\mathcal{A} \equiv \mathcal{A}\,]^{\cdot}$ would fail.

Now look once more at

$$[\,\varepsilon\mathsf{x}{\in}\mathbf{N}\colon 2\cdot\mathsf{x}+3=23\,]^{\cdot}.$$

The epsilon operator chooses an element of $[\,\mathbf{N}\,]^{\cdot}$ for which $[\,2{\cdot}\mathsf{x}{+}3=23\,]^{\cdot}$. That element will be a classical $[\,\mathsf{x}\,]^{\cdot}$ which weakly represents "ten". $[\,\varepsilon\mathsf{x}{\in}\mathbf{N}\colon 2{\cdot}\mathsf{x}{+}3 = 23\,]^{\cdot}$ may or may not be the strong representation of "ten". You definitely have that

$$[\,(\varepsilon\mathsf{x}{\in}\mathbf{N}\colon 2\cdot\mathsf{x}+3=23)=10\,]^{\cdot}$$

is true. On the contrary, I cannot tell whether

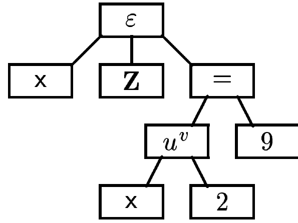
$$[\,(\varepsilon\mathsf{x}{\in}\mathbf{N}\colon 2\cdot\mathsf{x}+3=23)\equiv 10\,]^{\cdot}$$

holds or fails. That depends on epsilon.

The syntax tree of $[\,\varepsilon\mathsf{x}{\in}\mathbf{Z}\colon \mathsf{x}^2=9\,]^{\cdot}$ reads

---

choice operator



The first argument of $[\,\varepsilon\,]^{\circ}$ must be a variable. Here you have a binding diagram:

$$\left[\, \varepsilon x{\in}\mathbf{Z}{:}\, x^2 = 9 \,\right]^{\cdot}.$$

$[\,\varepsilon x{\in}S{:}\, \mathcal{A}\,]^{\cdot}$ binds all free occurrences of $[\,x\,]^{\cdot}$ in $[\,\mathcal{A}\,]^{\cdot}$ but does not bind those in $[\,S\,]^{\cdot}$. Here you have an example:

$$\left[\, \langle \varepsilon x{\in}x{:}\, x^2 = \langle x \mid x{:=}9\rangle \mid x{:=}\mathbf{Z}\rangle \,\right]^{\cdot}.$$

The epsilon operator is extremely handy because it can answer any mathematical question. Unfortunately, the epsilon operator is non-computable.

If $[\,S\,]^{\cdot}$ is a set and $[\,\mathcal{A}\,]^{\cdot}$ is non-classical for some $[\,x \in S\,]^{\cdot}$ then $[\,\varepsilon x{\in}S{:}\, \mathcal{A} \equiv \bot\,]^{\cdot}$. As an example you have

$$[\,\varepsilon x{\in}\mathbf{Z}{:}\, x! = 6 \equiv \bot\,]^{\cdot}$$

because $[\,x! = 6\,]^{\cdot}$ is non-classical for some $[\,x \in \mathbf{Z}\,]^{\cdot}$ (as an example, $[\,x! = 6\,]^{\cdot}$ equals $[\,\bot\,]^{\cdot}$ for $[\,x \equiv -1\,]^{\cdot}$).

If $[\,S\,]^{\cdot}$ is a set then $[\,\varepsilon x{\in}S{:}\, \mathcal{A}\,]^{\cdot}$ equals $[\,\bullet\,]^{\cdot}$ in all cases not covered above. As an example,

$$[\,(\varepsilon x{\in}\mathbf{Z}{:}\, x^2 = 10) \equiv \bullet\,]^{\cdot}$$

because no integer $[\,x\,]^{\cdot}$ satisfies $[\,x^2 = 10\,]^{\cdot}$.

Exercise 18.7.1 What are the possible values for the following expressions:

a $[\,\varepsilon x{\in}\mathbf{N}{:}\, x^2 = 9\,]^{\cdot}$.

b $[\,\varepsilon x{\in}\mathbf{T}{:}\, x^2 = 10\,]^{\cdot}$.

c $[\,\varepsilon x{\in}\mathbf{Z}{:}\, 2 \cdot x + 3 = 11\,]^{\cdot}$.

Exercise 18.7.2 Let $[\,\mathbf{R}\,]^{\cdot}$ denote the set of real numbers. For all non-negative real numbers $[\,x\,]^{\cdot}$, the square root $[\,\sqrt{x}\,]^{\cdot}$ is the unique, non-negative real number $[\,y\,]^{\cdot}$ for which $[\,y^2 = x\,]^{\cdot}$. For negative, real numbers $[\,x\,]^{\cdot}$, $[\,\sqrt{x} \equiv \bullet\,]^{\cdot}$. Use $[\,\varepsilon\,]^{\circ}$ to define $[\,\sqrt{x}\,]^{\cdot}$.

18.8 The [ElimExists] rules

If [S] is a set and if [$\exists x \in S: \mathcal{A}$] is true then [$\mathcal{A} \in \mathbf{B}$] for all [$x \in S$] and there exists an [$x \in S$] such that [$\mathcal{A}$] is true. I can easily state that [$\mathcal{A} \in \mathbf{B}$] for all [$x \in S$]:

[**Mac rule ElimExistsB** : $S \in \mathbf{Set} \vdash \exists x \in S: \mathcal{A} \vdash \mathcal{B} \in S \vdash \langle \mathcal{A} \mid x{:=}\mathcal{B} \rangle \in \mathbf{B}$]

I will now state that if [S] is a set and if [$\exists x \in S: \mathcal{A}$] then there exists an [$x \in S$] such that [$\mathcal{A}$] is true. To do so I need some term [$\mathcal{B}$] whose value belongs to [S] and which makes [$\mathcal{A}$] true. The epsilon operator allows me to form such a term:

[$\varepsilon x \in S: \mathcal{A}$].

The value of [$\varepsilon x \in S: \mathcal{A}$] belongs to [S] and [$\mathcal{A}$] becomes true when [x] is set to that value:

[**Mac rule ElimExistsS** : $S \in \mathbf{Set} \vdash \exists x \in S: \mathcal{A} \vdash (\varepsilon x \in S: \mathcal{A}) \in S$]

[**Mac rule ElimExists** : $S \in \mathbf{Set} \vdash \exists x \in S: \mathcal{A} \vdash \langle \mathcal{A} \mid x{:=}\varepsilon x \in S: \mathcal{A} \rangle$]

Here you have an example of use:

[**Mac lemma L18.8.1** : $z \in \mathbf{Z} \vdash \exists x \in \mathbf{Z}: 2 \cdot (x+1) = z \vdash \exists y \in \mathbf{Z}: 2 \cdot y = z$]

[**Mac proof of** L18.8.1**:**

L01 :	Premise ▷	$z \in \mathbf{Z}$	;
L02 :	Premise ▷	$\exists x \in \mathbf{Z}: 2 \cdot (x+1) = z$	;
L03 :	SetZ ▷	$\mathbf{Z} \in \mathbf{Set}$	;
L04 :	Block ▷	Begin	;
L05 :	Hypothesis ▷	$y \in \mathbf{Z}$	;
L06 :	TypeNumeralInZ ▷	$2 \in \mathbf{Z}$	;
L07 :	Type**Z**+**Z** ▷ L6 ▷ L5 ▷	$2 \cdot y \in \mathbf{Z}$	;
L08 :	Type**Z**=**Z** ▷ L7 ▷ L1 ▷	$(2 \cdot y = z) \in \mathbf{B}$	;
L09 :	Block ▷	End	;
L10 :	Local ▷	$u \equiv \varepsilon x \in \mathbf{Z}: 2 \cdot (x+1 = z)$	;
L11 :	Local ▷	$y \equiv u + 1$	;
L12 :	ElimExistsS ▷ L3 ▷ L2 ▷	$u \in \mathbf{Z}$	;
L13 :	TypeNumeralInZ ▷	$1 \in \mathbf{Z}$	;
L14 :	Type**Z**+**Z** ▷ L12 ▷ L13 ▷	$y \in \mathbf{Z}$	;
L15 :	ElimExists ▷ L3 ▷ L2 ▷	$2 \cdot (u+1) = z$	;
L16 :	Repetition ▷ L15 ▷	$2 \cdot y = z$	;
L17 :	IntroExists ▷ L3 ▷ L8 ▷ L14 ▷ L16 ▷	$\exists y \in \mathbf{Z}: 2 \cdot y = z$	]

18.9 Type rules for quantification

Here you have some type rules for quantification:

[**Mac rule TypeAll** : $x \in S \to \mathcal{A} \in \mathbf{B} \vdash (\forall x{\in}S\colon \mathcal{A}) \in \mathbf{B}$]

[**Mac rule TypeExists** : $x \in S \to \mathcal{A} \in \mathbf{B} \vdash (\exists x{\in}S\colon \mathcal{A}) \in \mathbf{B}$]

The reverse rules may also be useful:

[**Mac rule InverseTypeAll** : $(\forall x{\in}S\colon \mathcal{A}) \in \mathbf{B} \vdash \mathcal{B} \in S \vdash \langle \mathcal{A} \mid x{:=}\mathcal{B} \rangle \in \mathbf{B}$]

[**Mac rule InverseTypeExists** : $(\exists x{\in}S\colon \mathcal{A}) \in \mathbf{B} \vdash \mathcal{B} \in S \vdash \langle \mathcal{A} \mid x{:=}\mathcal{B} \rangle \in \mathbf{B}$]

18.10 Equality rules for quantification

Here you have some equality rules for quantification:

[**Mac rule EqualAll** : $x \in S \to \mathcal{A} \equiv \mathcal{B} \vdash \forall x{\in}S\colon \mathcal{A} \equiv \forall x{\in}S\colon \mathcal{B}$]

[**Mac rule EqualExists** : $x \in S \to \mathcal{A} \equiv \mathcal{B} \vdash \exists x{\in}S\colon \mathcal{A} \equiv \exists x{\in}S\colon \mathcal{B}$]

[**Mac rule EqualChoice** : $x \in S \to \mathcal{A} \equiv \mathcal{B} \vdash \varepsilon x{\in}S\colon \mathcal{A} \equiv \varepsilon x{\in}S\colon \mathcal{B}$]

18.11 Fundamental constructs

This entire book builds on a fairly simple derivation system which I call *map theory*. Map theory has the following six fundamental constructs:

$[\, \mathsf{N} \,]^{\cdot}$
$[\, \lambda x.y \,]^{\cdot}$
$[\, x\, \text{'}\, y \,]^{\cdot}$
$[\, \text{case}(x,y,z) \,]^{\cdot}$
$[\, \bar{\varepsilon} x \,]^{\cdot}$
$[\, \bar{\exists} x \,]^{\cdot}$

You have seen the four first constructs before, and I describe the last two in a moment.

Almost all constructs mathematicians have ever studied can be defined on basis of these six constructs alone.

All operators and numerals in this book can be defined from the six fundamental constructs above. As examples, $[\, x + y \,]^{\cdot}$ and $[\, 117\mathtt{F} \,]^{\cdot}$ and $[\, \forall x{\in}S\colon \mathcal{A} \,]^{\cdot}$ can be defined from the fundamental constructs.

The terms of map theory comprise all terms built from variables and the six fundamental constructs. For convenience, terms may also include defined

map theory

operators and numerals, which are defined on basis of the six fundamental constructs.

Lemmas and antilemmas of map theory have form $[\mathcal{A} \equiv \mathcal{B}]^{\cdot}$ where $[\mathcal{A}]^{\cdot}$ and $[\mathcal{B}]^{\cdot}$ are terms. All mac rules can be proved from the rules of map theory.

The construct $[\bar{\exists}\mathsf{x}]^{\cdot}$ has the following properties:

$[\bar{\exists}\mathsf{x} \equiv \mathsf{T}]^{\cdot}$ if $[\mathsf{x}\,'\,\mathsf{y} \equiv \mathsf{T}]^{\cdot}$ for at least one map $[\mathsf{y}]^{\cdot}$, and

$[\bar{\exists}\mathsf{x} \equiv \bot]^{\cdot}$ otherwise.

The construct $[\bar{\varepsilon}\mathsf{x}]^{\cdot}$ is a choice construct with the following properties:

If	$[\mathsf{x}\,'\,\mathsf{y} \equiv \bot]^{\cdot}$ for some classical $[\mathsf{y}]^{\cdot}$
then	$[\bar{\varepsilon}\mathsf{x} \equiv \bot]^{\cdot}$
else	
if	$[\mathsf{x}\,'\,\mathsf{y} \equiv \mathsf{T}]^{\cdot}$ for some classical $[\mathsf{y}]^{\cdot}$
then	$[\bar{\varepsilon}\mathsf{x}]^{\cdot}$ is classical and $[\mathsf{x}\,'\,\bar{\varepsilon}\mathsf{x} \equiv \mathsf{T}]^{\cdot}$
else	$[\bar{\varepsilon}\mathsf{x}]^{\cdot}$ is classical.

In Appendix B you can find a complete definition of map theory. The definition takes up about four pages. You don't need to know the definition of map theory since you can use the $[\mathsf{Mac}]^{\cdot}$ system instead, but you may find it nice to know that all of mathematics can be developed from those four pages in Appendix B.

18.12 Answers

Answer 18.1.1 (a) $[\mathsf{T}]^{\cdot}$ and (b) $[\bot]^{\cdot}$.

Answer 18.1.2 (a) $[\mathsf{T}]^{\cdot}$ and (b) $[\bot]^{\cdot}$.

Answer 18.2.1

[**Mac proof of** L18.2.2:

L1 :	SetDi ⊳	$\mathbf{D}^{\cdot}_{\infty} \in \mathbf{Set}$	;
L2 :	Block ⊳	Begin	;
L3 :	Hypothesis ⊳	$\mathsf{x} \in \mathbf{D}^{\cdot}_{\infty}$	;
L4 :	TypeNumeralInDi ⊳	$1 \in \mathbf{D}^{\cdot}_{\infty}$	;
L5 :	Type$\mathbf{D}^{\cdot}_{\infty}$+$\mathbf{D}^{\cdot}_{\infty}$ ⊳	$\mathsf{x}+1 \in \mathbf{D}^{\cdot}_{\infty}$	;
L6 :	Block ⊳	End	;
L7 :	Gen ⊳ L1 ⊳ L5 ⊳	$\forall \mathsf{x}{\in}\mathbf{D}^{\cdot}_{\infty} : \mathsf{x}+1 \in \mathbf{D}^{\cdot}_{\infty}$	]

Answer 18.3.2

[**Mac proof of** L18.3.3:

L01 :	SetZ ▷	$\mathbf{Z} \in \mathbf{Set}$	;
L02 :	Block ▷	Begin	;
L03 :	Hypothesis ▷	$\quad \forall y{\in}\mathbf{Z}: x > y$	;
L04 :	Block ▷	$\quad$ Begin	;
L05 :	Hypothesis ▷	$\qquad y \in \mathbf{Z}$	;
L06 :	TypeNumeralInZ ▷	$\qquad 1 \in \mathbf{Z}$	;
L07 :	Type**Z**+**Z** ▷	$\qquad y + 1 \in \mathbf{Z}$	;
L08 :	ElimAll ▷ L1 ▷ L3 ▷ L7 ▷	$\qquad x > y + 1$	;
L09 :	Block ▷	$\quad$ End	;
L10 :	Gen ▷ L1 ▷ L8 ▷	$\quad \forall y{\in}\mathbf{Z}: x > y + 1$	;
L11 :	Block ▷	End	]

Answer 18.7.1 (a) It can be any, classical $[\ x\ ]^{\cdot}$ which weakly represents "three". (b) $[\ \bullet\ ]^{\cdot}$. (c) It can be any, classical $[\ x\ ]^{\cdot}$ which weakly represents "four".
Answer 18.7.2 $[\ \sqrt{x} \doteq \varepsilon y{\in}\mathbf{R}: y \geq 0 \wedge y^2 = x\ ]^{\cdot}$.