

Mathematics and Computation
7'th edition
Volume 3

Klaus Grue

June 1, 2001

Mathematics and Computation, 7th edition, Volume 3 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 [\perp]	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	$[x \equiv y]$ 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	$[x \wedge y]$	154
4.3	Logical “or”	155
4.4	$[x \vee y]$	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	[Mac] rules	188
6.7	[Mac] lemmas	189
6.8	[Mac] proofs	190
6.9	Use of abbreviations	192
6.10	The [Mac] 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	$[\lambda x.y]$ and $[x'y]$	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 $[x \preceq y]$	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 $[\text{Gen}]$ rule	436
18.3	The $[\text{ElimAll}]$ rule	437
18.4	Existential quantification	437
18.5	The $[\text{IntroExists}]$ rule	438
18.6	Local definitions	439
18.7	Choice	439
18.8	The $[\text{ElimExists}]$ 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 $[\text{case}(x, y, z)]$	506
A.7	choose x in y colon z $[\varepsilon x \in y : z]$	507
A.8	classical $[\ell]$	508
A.9	construct x $[\text{construct } x]$	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 $[\text{if}(x, y, z)]$	519
A.19	infinity $[\infty]$	521
A.20	lambda x dot y $[\lambda x.y]$	522
A.21	minimum x comma y end $[\text{min}(x, y)]$	523

A.22	minus x [$-x$]	524
A.23	nil map [\mathbf{N}]	525
A.24	not x [$\neg x$]	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 [$\langle A \mid x:=B \rangle$]	530
A.29	the class of sets [\mathbf{Set}]	532
A.30	the set of decimal fractions [\mathbf{D}]	533
A.31	the set of decimal fractions of precision x end [\mathbf{D}_x]	535
A.32	the set of exceptions [\mathbf{X}]	537
A.33	the set of false maps [\mathbf{F}]	539
A.34	the set of infinities of precision x end [\mathbf{D}_x^∞]	541
A.35	the set of integers [\mathbf{Z}]	543
A.36	the set of minus infinities of precision x end [$\mathbf{D}_x^{-\infty}$]	545
A.37	the set of natural numbers [\mathbf{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 [\mathbf{E}]	553
A.41	the set of true maps [\mathbf{T}]	554
A.42	the set of truth values [\mathbf{B}]	556
A.43	true [\mathbf{T}]	558
A.44	tuple x end [$\langle x \rangle$]	559
A.45	variable x [$\mathbf{variable} x$]	560
A.46	x and y [$x \wedge y$]	561
A.47	x apply y [$x' y$]	563
A.48	x associates as y [$x \dot{\rightarrow} y$]	564
A.49	x atom [$x \mathbf{atom}$]	565
A.50	x belongs to y [$x \in y$]	566
A.51	x cartesian y [$x \times y$]	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 [$x \neq y$]	573
A.56	x concludes y [$x \triangleright y$]	575
A.57	x deduces y [$x \rightarrow y$]	576
A.58	x defined equal y [$x \doteq y$]	577
A.59	x divide y [x/y]	578
A.60	x equal y [$x \equiv y$]	580
A.61	x exceptional [$x?$]	582
A.62	x faculty [$x!$]	583
A.63	x greater priority y [$x \succ y$]	584
A.64	x head [$x \mathbf{head}$]	585
A.65	x if and only if y [$x \Leftrightarrow y$]	586
A.66	x implied by y [$x \Leftarrow y$]	588
A.67	x implies y [$x \Rightarrow y$]	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 [$x \doteq y$]	596
A.73	x minus y [$x - y$]	597
A.74	x modulo y [$x \% y$]	599
A.75	x modus ponens y [$x \supseteq y$]	601
A.76	x optimised equal y [$x \doteq y$]	602
A.77	x or y [$x \vee y$]	603
A.78	x pair [$x \text{ pair }$]	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 [$x \xrightarrow{\pm} y$]	613
A.85	x round y [$x@y$]	614
A.86	x rule y colon z	616
A.87	x same priority y [$x \cong y$]	617
A.88	x semicolon y [$x; y$]	618
A.89	x simple head [$x \text{ Head}$]	619
A.90	x simple pair y [$x \cdot y$]	620
A.91	x simple tail [$x \text{ Tail}$]	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 [$x \text{ tail}$]	627
A.96	x term reduces to y [$x \xrightarrow{\circ} y$]	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

Appendix A

Reference Manual

A.1 Error reporting

If (despite what I said in Section 1.1) you find an error in this book, please see if it is already reported at

<http://www.diku.dk/~grue/mob0102/index.html>

If it is not, please notify me at

grue@diku.dk

Suggestions for improvement are also welcome.

A.2 Priority

This page is repeated on the last page of this volume.

[Lx	V<
x' y	V<
x@y ≐ x@ ₂ y ≐ x@ ₊ y ≐ x@ ₋ y ≐ x@ ₀ y	V<
#x	V<
x! ≐ x ^y ≐ x ⁺ ≐ x ⁻ ≐ x [*] ≐	V<
x head ≐ x tail ≐ x pair ≐ x atom ≐ x Head ≐ x Tail	V<
x · y ≐ x · ₊ y ≐ x · ₋ y ≐ x · ₀ y ≐ x · ₂ y ≐	V<
x / y ≐ x / ₊ y ≐ x / ₋ y ≐ x / ₀ y ≐ x / ₂ y ≐	V<
x // y ≐ x // ₊ y ≐ x // ₋ y ≐ x // ₀ y ≐ x // ₂ y ≐	V<
x % y ≐ x % ₊ y ≐ x % ₋ y ≐ x % ₀ y ≐ x % ₂ y	V<
+x ≐ -x	V<
x + y ≐ x + ₊ y ≐ x + ₋ y ≐ x + ₀ y ≐ x + ₂ y ≐	V<
x - y ≐ x - ₊ y ≐ x - ₋ y ≐ x - ₀ y ≐ x - ₂ y	V<
x :: y ≐ x ∴ y ≐ x × y	V<
x append y	V<
x, y	V<
x = y ≐ x ≠ y ≐ x < y ≐ x ≤ y ≐ x > y ≐ x ≥ y ≐ x ∈ y ≐ x?	V<
¬x	V<
x ∧ y	V<
x ∨ y	V<
∀x∈y: z ≐ ∃x∈y: z ≐ εx∈y: z	V<
x ⇒ y ≐ x ⇐ y ≐ x ⇔ y	V<
x { y ≐ x { y	V<
x { z ≐ x { z	V<
λx.y	V<
x ≡ y ≐ x ≐ y ≐ x ≐ y	V<
x → y	V<
x ⊢ y	V<
x ▷ y	V<
x : y	V<
x; y	V<
Mac lemma x:y ≐ Mac proof of x:y ≐ Mac rule x:y	V<
]	V<

A.3 Associativity and term reduction

This page is repeated on the penultimate page of this volume.

[$x' y' z$	\mapsto	$(x' y)' z$]
[$x@y@z$	\mapsto	$(x@y)@z$]
[xy^z	\mapsto	$x(y^z)$]
[$x \cdot y \cdot z$	\mapsto	$(x \cdot y) \cdot z$]
[$x + y + z$	\mapsto	$(x + y) + z$]
[$x :: y :: z$	\mapsto	$x :: (y :: z)$]
[$x \text{ append } y$	\mapsto	$x \text{ append } y$]
[$x \text{ append } y$ $\text{ append } z$	\mapsto	$x \text{ append } (y \text{ append } z)$]
[x, y, z	\mapsto	$x, (y, z)$]
[$x \wedge y \wedge z$	\mapsto	$(x \wedge y) \wedge z$]
[$x \vee y \vee z$	\mapsto	$(x \vee y) \vee z$]
[$x \vdash y \vdash z$	\mapsto	$x \vdash (y \vdash z)$]
[$x \triangleright y \triangleright z$	\mapsto	$(x \triangleright y) \triangleright z$]
[$x : y : z$	\mapsto	$x : (y : z)$]
[$x; y; z$	\mapsto	$(x; y); z$]
[$x < y \leq z$	\mapsto	$(x < y) \leq z$]
[$x < y \leq z$	\mapsto°	$(x < y) \wedge (y \leq z) \succ$]
[$x, y < z$	\mapsto°	$x < z \wedge y < z \succ$]
[$x < y, z$	\mapsto°	$x < y \wedge x < z$]
[$\forall x, y \in S: a$	\mapsto°	$\forall x \in S: \forall y \in S: a$]
[$x \Rightarrow y \Leftrightarrow z$	\mapsto	$(x \Rightarrow y) \Leftrightarrow z$]
[$x \Rightarrow y \Leftrightarrow z$	\mapsto°	$(x \Rightarrow y) \wedge (y \Leftrightarrow z)$]
[$x \rightarrow y \rightarrow z$	\mapsto	$x \rightarrow (y \rightarrow z)$]
[$x \rightarrow y \equiv z$	\mapsto°	$\text{case}(x, y, T) \equiv \text{case}(x, z, T)$]
[$\langle x, y \rangle$	\mapsto°	$x :: \langle y \rangle$]

A.4 all x in y colon z $[\forall x \in y : z]$

Parse tree



Sort

$[\forall x \in y : z]$ is an operator (i.e. it may occur in the syntax tree of a term).

Binding

$[\forall * \in * : *]$.

Description

If $[S]$ is a set then the following holds: $[\forall x \in S : \mathcal{A} \equiv \top]$ if $[\mathcal{A} \in \mathbf{T}]$ for all $[x \in S]$. $[\forall x \in S : \mathcal{A} \equiv \mathbf{F}]$ if $[\mathcal{A} \in \mathbf{B}]$ for all $[x \in S]$ and $[\mathcal{A} \in \mathbf{F}]$ for some $[x \in S]$. $[\forall x \in S : \mathcal{A} \equiv \bullet]$ if $[\mathcal{A}]$ is classical for all $[x \in S]$ and $[x \notin \mathbf{B}]$ for some $[x \in S]$. $[\forall x \in S : \mathcal{A} \equiv \perp]$ if $[\mathcal{A}]$ is non-classical for some $[x \in S]$.

Mac rules

[Mac rule Gen : $S \in \mathbf{Set} \vdash x \in S \rightarrow \mathcal{A} \vdash \forall x \in S : \mathcal{A}$]

[Mac rule ElimAll : $S \in \mathbf{Set} \vdash \forall x \in S : \mathcal{A} \vdash \mathcal{B} \in S \vdash \langle \mathcal{A} \mid x := \mathcal{B} \rangle$]

[Mac rule TypeAll : $x \in S \rightarrow \mathcal{A} \in \mathbf{B} \vdash (\forall x \in S : \mathcal{A}) \in \mathbf{B}$]

[Mac rule InverseTypeAll : $(\forall x \in S : \mathcal{A}) \in \mathbf{B} \vdash \mathcal{B} \in S \vdash \langle \mathcal{A} \mid x := \mathcal{B} \rangle \in \mathbf{B}$]

[Mac rule EqualAll : $x \in S \rightarrow \mathcal{A} \equiv \mathcal{B} \vdash \forall x \in S : \mathcal{A} \equiv \forall x \in S : \mathcal{B}$]

Examples

$[\forall x \in \mathbf{N} : 2 \cdot x \geq x \quad \equiv \quad \top]$

$[\forall x \in \mathbf{Z} : 2 \cdot x \geq x \quad \equiv \quad \mathbf{F}]$

$[\forall x \in \mathbf{Z} : 1 // x = 1 // x \quad \equiv \quad \bullet]$

$[\forall x \in \mathbf{Z} : x! = x! \quad \equiv \quad \perp]$

See also

$[\varepsilon x \in y : z]$ Section A.7

$[\exists x \in y : z]$ Section A.12

$[\forall x \in y : z]$	all x in y colon z
[Gen]	rule Gen
[ElimAll]	rule ElimAll
[TypeAll]	rule TypeAll
[InverseTypeAll]	rule InverseTypeAll
[EqualAll]	rule EqualAll

A.5 bottom $[\perp]$

Parse tree

$$\boxed{\perp}$$

Sort

$[\perp]$ is an operator (i.e. it may occur in the syntax tree of a term).

Description

$[\perp]$ denotes total absence of information. If computation of a term proceeds forever and yields no information about the value of the term, then the term equals $[\perp]$. Among all mathematical entities, $[\perp]$ is the least one with respect to the information comparison relation $[x \preceq y]$.

Mac rules

[Mac rule InfoBottom : $\perp \preceq x$]

[Mac rule SubBottom : $\langle \perp \mid x := S \rangle \equiv \perp$]

Examples

$[(-1)! \equiv \perp]$

$[(-2)! \equiv \perp]$

See also

[•] Section A.11

$[x \preceq y]$ Section A.99

$[\perp]$ bottom
[SubBottom] rule SubBottom

A.6 case x comma y comma z end [case(x, y, z)]

Parse tree



Sort

[case(x, y, z)] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[case(x, y, z)] equals [y] if [x] equals [N]. It equals [z] if [x] is a function, i.e. can be written on form [$\lambda u. \mathcal{A}$]. It equals [\perp] if [x] equals [\perp]. [case(x, y, z)] is one of the fundamental constructs of map theory.

Mac rules

[**Mac rule SubCase** : $\langle \text{case}(\mathcal{A}, \mathcal{B}, \mathcal{C}) \mid x := \mathcal{S} \rangle \equiv \text{case}(\langle \mathcal{A} \mid x := \mathcal{S} \rangle, \langle \mathcal{B} \mid x := \mathcal{S} \rangle, \langle \mathcal{C} \mid x := \mathcal{S} \rangle)$]
 [**Mac rule CaseNil** : $\text{case}(N, y, z) \equiv y$]
 [**Mac rule CaseT** : $\text{case}(T, y, z) \equiv y$]
 [**Mac rule CaseLambda** : $\text{case}(\lambda u. v, y, z) \equiv z$]
 [**Mac rule CaseBottom** : $\text{case}(\perp, y, z) \equiv \perp$]
 [**Mac rule CasePair** : $\text{case}(u \cdot v, y, z) \equiv z$]

Examples

[case(N, 2.0F, 3.00F) \equiv 2.0F]

See also

[$\lambda x. y$] Section A.20
 [N] Section A.23
 [x ' y] Section A.47

[case(x, y, z)]	case x comma y comma z end
[SubCase]	rule SubCase
[CaseNil]	rule CaseNil
[CaseT]	rule CaseT
[CaseLambda]	rule CaseLambda
[CaseBottom]	rule CaseBottom
[CasePair]	rule CasePair

A.7 choose x in y colon z $[\varepsilon x \in y : z]$

Parse tree



Sort

$[\varepsilon x \in y : z]$ is an operator (i.e. it may occur in the syntax tree of a term).

Binding

$[\varepsilon * \varepsilon * : *]$.

Description

If $[S]$ is a set then the following holds: If $[\mathcal{A} \in \mathbf{B}]$ for all $[x \in S]$ and if $[\mathcal{A}]$ is true for at least one $[x \in S]$, then $[\varepsilon x \in S : \mathcal{A}]$ equals some $[x \in S]$ for which $[\mathcal{A}]$ is true. If $[\mathcal{A}]$ is true for more than one $[x \in S]$ then it is impossible to tell which one $[\varepsilon]^\circ$ chooses. If $[\mathcal{A}]$ is non-classical for some $[x \in S]$ then $[\varepsilon x \in S : \mathcal{A}]$ equals $[\perp]$. If $[S]$ is a set and none of the two cases above apply, then $[\varepsilon x \in S : \mathcal{A}]$ equals $[\bullet]$.

Mac rules

[Mac rule **EqualChoice** : $x \in S \rightarrow \mathcal{A} \equiv \mathcal{B} \vdash \varepsilon x \in S : \mathcal{A} \equiv \varepsilon x \in S : \mathcal{B}$]

Examples

$[\varepsilon x \in \mathbf{Z} : x \cdot x = 9]$ is a classical map that weakly represents $[3]$ or $[-3]$.

See also

$[\forall x \in y : z]$ Section A.4
 $[\exists x \in y : z]$ Section A.12

$[\varepsilon x \in y : z]$ choose x in y colon z
 $[\text{EqualChoice}]$ rule EqualChoice

A.8 classical [ℓ]

Parse tree

ℓ

Sort

[ℓ] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[$\ell' x$] equals [\top] if [x] is classical and equals [\perp] otherwise.

Mac rules

[**Mac rule ClassicalNil** : $\ell' N \equiv \top$]
 [**Mac rule ClassicalBottom** : $\ell' \perp \equiv \perp$]
 [**Mac rule ClassicalPair** : $\ell' (x \therefore y) \equiv \ell' x \wedge \ell' y$]
 [**Mac rule SubClassical** : $\langle \ell \mid x := \mathcal{S} \rangle \equiv \ell$]
 [**Mac rule ClassicalWellfounded** : $\ell' x \vdash \text{wf}(x)$]

Examples

[$\ell' (N \therefore N)$] \equiv [\top]
 [$\ell' (N \therefore (\perp \therefore N))$] \equiv [\perp]
 [$\ell' \infty\text{-list}(1)$] \equiv [\perp]

See also

[$x \in y$] Section A.50

[ℓ]	classical
[ClassicalNil]	rule ClassicalNil
[ClassicalBottom]	rule ClassicalBottom
[ClassicalPair]	rule ClassicalPair
[SubClassical]	rule SubClassical
[ClassicalWellfounded]	rule ClassicalWellfounded

A.9 `construct x [construct x]`

Parse tree



Sort

`[construct x]` is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

`[construct A]` introduces the construct `[A]` without assigning a meaning to it. `[A]` gets no meaning and, hence, no value, so it becomes a statement construct rather than an operator.

Examples

`[construct x,y]` introduces the construct `[x,y]` as a directive with no meaning. `[⟨x,y⟩]` is a term even though it contains the directive `[x,y]`. This is so because `[x,y]` disappears during term reduction so that `[x,y]` does not occur in the syntax tree of `[⟨x,y⟩]`.

See also

`[x ≐ y]` Section A.58
`[variable x]` Section A.45

A.10 empty list [$\langle \rangle$]

Parse tree

$\langle \rangle$

Sort

[$\langle \rangle$] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[$\langle \rangle$] denotes the empty list, i.e. the unique list that has zero elements. [$\langle \rangle$] is an atom.

Mac rules

[**Mac rule SubEmptyList** : $\langle \rangle \mid x:=S \equiv \langle \rangle$]
 [**Mac rule TypeEmptyListInE** : $\langle \rangle \in \mathbf{E}$]

Examples

[$\langle \rangle$ pair $\equiv \mathbf{F}$]
 [$\langle \rangle$ atom $\equiv \mathbf{T}$]
 [$\langle \rangle$ head $\equiv \langle \rangle$]
 [$\langle \rangle$ tail $\equiv \langle \rangle$]

See also

[\mathbf{E}] Section A.40
 [$x :: y$] Section A.79
 [$\langle x \rangle$] Section A.44

[$\langle \rangle$] empty list
 [SubEmptyList] rule SubEmptyList
 [TypeEmptyListInE] rule TypeEmptyListInE

A.11 exception [•]

Parse tree



Sort

[•] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[•] denotes nonsense. If a computer can determine within finite time that a term is meaningless, then the value of the term is [•]. As an example, one divided by zero equals [•].

Mac rules

[Mac rule SubException : $\langle \bullet \mid x := S \rangle \equiv \bullet$]

[Mac rule TypeXInX : $\bullet \in X$]

Examples

[$1/0 \equiv \bullet$]

[$1 < F \equiv \bullet$]

See also

[\perp] Section A.5

[•]	exception
[SubException]	rule SubException
[TypeXInX]	rule TypeXInX

A.12 exists x in y colon z $[\exists x \in y : z]$

Parse tree



Sort

$[\exists x \in y : z]$ is an operator (i.e. it may occur in the syntax tree of a term).

Binding

$[\exists * \in * : *]$.

Description

If $[S]$ is a set then the following holds: $[\exists x \in S : \mathcal{A} \equiv F]$ if $[\mathcal{A} \in F]$ for all $[x \in S]$. $[\exists x \in S : \mathcal{A} \equiv T]$ if $[\mathcal{A} \in B]$ for all $[x \in S]$ and $[\mathcal{A} \in T]$ for some $[x \in S]$. $[\exists x \in S : \mathcal{A} \equiv \bullet]$ if $[\mathcal{A}]$ is classical for all $[x \in S]$ and $[\mathcal{A} \notin B]$ for some $[x \in S]$. $[\exists x \in S : \mathcal{A} \equiv \perp]$ if $[\mathcal{A}]$ is non-classical for some $[x \in S]$.

Mac rules

[Mac rule IntroExists : $S \in \mathbf{Set} \vdash x \in S \rightarrow \mathcal{A} \in \mathbf{B} \vdash \mathcal{B} \in \mathbf{S} \vdash \langle \mathcal{A} \mid x := \mathcal{B} \rangle \vdash \exists x \in S : \mathcal{A}$]

[Mac rule ElimExistsB : $S \in \mathbf{Set} \vdash \exists x \in S : \mathcal{A} \vdash \mathcal{B} \in \mathbf{S} \vdash \langle \mathcal{A} \mid x := \mathcal{B} \rangle \in \mathbf{B}$]

[Mac rule ElimExistsS : $S \in \mathbf{Set} \vdash \exists x \in S : \mathcal{A} \vdash (\varepsilon x \in S : \mathcal{A}) \in \mathbf{S}$]

[Mac rule ElimExists : $S \in \mathbf{Set} \vdash \exists x \in S : \mathcal{A} \vdash \langle \mathcal{A} \mid x := \varepsilon x \in S : \mathcal{A} \rangle$]

[Mac rule TypeExists : $x \in S \rightarrow \mathcal{A} \in \mathbf{B} \vdash (\exists x \in S : \mathcal{A}) \in \mathbf{B}$]

[Mac rule InverseTypeExists : $(\exists x \in S : \mathcal{A}) \in \mathbf{B} \vdash \mathcal{B} \in \mathbf{S} \vdash \langle \mathcal{A} \mid x := \mathcal{B} \rangle \in \mathbf{B}$]

[Mac rule EqualExists : $x \in S \rightarrow \mathcal{A} \equiv \mathcal{B} \vdash \exists x \in S : \mathcal{A} \equiv \exists x \in S : \mathcal{B}$]

Examples

$[\exists x \in \mathbf{N} : x! = 6 \quad \equiv \quad \mathbf{T}]$
 $[\exists x \in \mathbf{N} : x! = 7 \quad \equiv \quad \mathbf{F}]$
 $[\exists x \in \mathbf{Z} : x! = 6 \quad \equiv \quad \perp]$
 $[\exists x \in \mathbf{N} : 10 // x = 5 \quad \equiv \quad \bullet]$

[$\exists x \in y : z$]	exists x in y colon z
[IntroExists]	rule IntroExists
[ElimExistsB]	rule ElimExistsB
[ElimExistsS]	rule ElimExistsS
[ElimExists]	rule ElimExists
[TypeExists]	rule TypeExists
[InverseTypeExists]	rule InverseTypeExists
[EqualExists]	rule EqualExists

See also

- [$\forall x \in y: z$] Section A.4
- [$\varepsilon x \in y: z$] Section A.7

A.13 false [F]

Parse tree

F

Sort

[F] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[F] denotes falsehood. If a computer can determine within finite time that a term is false, then the value of the term is [F].

Mac rules

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

[**Mac rule TypeFInF** : $F \in \mathbf{F}$]

[**Mac rule SubFalse** : $\langle F \mid x := \mathcal{S} \rangle \equiv F$]

Examples

[$2 < 1 \equiv F$]

[$1 = T \equiv F$]

See also

[T] Section A.43

[if(x, y, z)] Section A.18

[F]	false
[TypeFInB]	rule TypeFInB
[TypeFInF]	rule TypeFInF
[SubFalse]	rule SubFalse

A.14 f four of x end [$f_4(x)$]**Parse tree****Sort** $[f_4(x)]$ is a macro.**Definition** $[f_4(x) \doteq 2 \cdot x + 4]$ **See also**

$[f_2]$ Section A.17
 $[f_3(x)]$ Section A.16
 $[x \doteq y]$ Section A.72

A.15 f one of x end [$f_1(x)$]**Parse tree****Sort**[$f_1(x)$] is an operator (i.e. it may occur in the syntax tree of a term).**Definition**[$f_1(x) \doteq x^2 - 1$]**Mac rules**[**Mac rule DefFOneOfXEnd** : $f_1(x) \equiv x^2 - 1$][**Mac rule SubFOneOfXEnd** : $\langle f_1(\mathcal{A}) \mid x:=S \rangle \equiv f_1(\langle \mathcal{A} \mid x:=S \rangle)$]**See also**[$x \doteq y$] Section A.58

[$f_1(x)$]	f one of x end
[DefFOneOfXEnd]	rule DefFOneOfXEnd
[SubFOneOfXEnd]	rule SubFOneOfXEnd

A.16 f three of x end [$f_3(x)$]**Parse tree****Sort**

[$f_3(x)$] is an operator (i.e. it may occur in the syntax tree of a term).

Definition

[$f_3(x) \doteq 2 \cdot x + 4$]

Mac rules

[**Mac rule DefFThreeOfXEnd** : $f_3(x) \equiv 2 \cdot x + 4$]

[**Mac rule SubFThreeOfXEnd** : $\langle f_3(\mathcal{A}) \mid x := S \rangle \equiv f_3(\langle \mathcal{A} \mid x := S \rangle)$]

See also

[f_2] Section A.17

[$f_4(x)$] Section A.14

[$x \doteq y$] Section A.58

[$f_3(x)$]	f three of x end
[DefFThreeOfXEnd]	rule DefFThreeOfXEnd
[SubFThreeOfXEnd]	rule SubFThreeOfXEnd

A.17 f two [f_2]**Parse tree**

f_2

Sort

[f_2] is an operator (i.e. it may occur in the syntax tree of a term).

Definition

[$f_2 \doteq \lambda x.2 \cdot x + 4$]

Mac rules

[**Mac rule DefFTwo** : $f_2 \equiv \lambda x.2 \cdot x + 4$]

[**Mac rule ApplyFTwo** : $f_2 'x \equiv 2 \cdot x + 4$]

[**Mac rule SubFTwo** : $\langle f_2 \mid x:=S \rangle \equiv f_2$]

See also

[$f_3(x)$] Section A.16
 [$f_4(x)$] Section A.14
 [$\lambda x.y$] Section A.20
 [$x \doteq y$] Section A.58

[f_2]	f two
[DefFTwo]	rule DefFTwo
[ApplyFTwo]	rule ApplyFTwo
[SubFTwo]	rule SubFTwo

A.18 if x then y else z end [if(x, y, z)]

Parse tree



Sort

[if(x, y, z)] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[if(x, y, z)] equals [y] and [z] if [x] is true and false, respectively. If [x] is a decimal fraction, an exception, a pair, or the empty list, then [if(x, y, z) ≡ •]. If [x] equals [⊥] then [if(x, y, z) ≡ ⊥].

Mac rules

[**Mac rule IfTrue** : if(⊤, y, z) ≡ y]
 [**Mac rule IfFalse** : if(⊥, y, z) ≡ z]
 [**Mac rule IfNil** : if(⟨⟩, y, z) ≡ •]
 [**Mac rule IfPair** : if(u :: v, y, z) ≡ •]
 [**Mac rule IfException** : if(•, y, z) ≡ •]
 [**Mac rule IfBottom** : if(⊥, y, z) ≡ ⊥]
 [**Mac rule SubIf** : ⟨if(Ⓐ, Ⓑ, Ⓒ) | x:=S⟩ ≡ if(⟨Ⓐ | x:=S⟩, ⟨Ⓑ | x:=S⟩, ⟨Ⓒ | x:=S⟩)]
 [**Mac rule IfT** : x ∈ **T** ⊢ if(x, y, z) ≡ y]
 [**Mac rule IfF** : x ∈ **F** ⊢ if(x, y, z) ≡ z]
 [**Mac rule IfD** : x ∈ **D** ⊢ if(x, y, z) ≡ •]
 [**Mac rule IfE** : x ∈ **E** ⊢ if(x, y, z) ≡ •]
 [**Mac rule IfX** : x ∈ **X** ⊢ if(x, y, z) ≡ •]

[if(x, y, z)]	if x then y else z end
[IfTrue]	rule IfTrue
[IfFalse]	rule IfFalse
[IfNil]	rule IfNil
[IfPair]	rule IfPair
[IfException]	rule IfException
[IfBottom]	rule IfBottom
[SubIf]	rule SubIf
[IfT]	rule IfT
[IfF]	rule IfF
[IfD]	rule IfD
[IfE]	rule IfE
[IfX]	rule IfX

Examples
$$\begin{aligned} [\text{if}(\top, 2.0F, 3.00F) &\equiv 2.0F] \\ [\text{if}(2.0F, \perp, \perp) &\equiv \bullet] \end{aligned}$$
See also

[\top] Section A.43
[F] Section A.13
[\bullet] Section A.11
[\perp] Section A.5

A.19 infinity [∞]

Parse tree

$\boxed{\infty}$

Sort

[∞] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[∞] denotes plus infinity. [∞] is a decimal fraction that is weakly greater than any decimal fraction (including itself). [∞] has precision [∞], i.e. it is an exact decimal fraction. All infinite decimal fractions can be constructed from [∞] together with unary minus and rounding. As an example, [-∞@3] denotes minus infinity with three significant digits.

Mac rules

[**Mac rule SubInfinity** : $\langle \infty \mid x := \mathcal{S} \rangle \equiv \infty$]
 [**Mac rule TypeIiInIi** : $\infty \in \mathbf{D}_{\infty}^{\infty}$]
 [**Mac rule TypeMiInMi** : $-\infty \in \mathbf{D}_{\infty}^{-\infty}$]
 [**Mac rule TypeImInIm** : $m \in \mathbf{Z}^+ \vdash \infty@m \in \mathbf{D}_m^{\infty}$]
 [**Mac rule TypeMmInMm** : $m \in \mathbf{Z}^+ \vdash -\infty@m \in \mathbf{D}_m^{-\infty}$]

Examples

[$2 < \infty \quad \equiv \quad \mathbf{T}$]
 [$-\infty < 2 \quad \equiv \quad \mathbf{T}$]
 [$2.3\mathbf{F}@\infty \quad \equiv \quad 2.3$]

See also

[-x] Section A.22
 [x@y] Section A.85

[∞]	infinity
[SubInfinity]	rule SubInfinity
[TypeIiInIi]	rule TypeIiInIi
[TypeMiInMi]	rule TypeMiInMi
[TypeImInIm]	rule TypeImInIm
[TypeMmInMm]	rule TypeMmInMm

A.20 lambda x dot y [$\lambda x.y$]


Parse tree



Sort

[$\lambda x.y$] is an operator (i.e. it may occur in the syntax tree of a term).

Binding

[$\lambda x.*$]


Description

[$\lambda x.y$] is the function that maps [x] to [y]. It is one of the fundamental constructs of map theory. [x] must be a variable.

Mac rules

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

[**Mac rule SubLambdaXYFreeFor** :

if $\text{distinct}(x, y) \wedge \text{freefor}(\lambda x.A, y, S)$ then $\langle \lambda y.A \mid x:=S \rangle \equiv \lambda y. \langle A \mid x:=S \rangle$]

[**Mac rule SubLambdaXY** :

if $\text{distinct}(x, y)$ then $\langle \lambda y.A \mid x:=\langle S \mid y:=T \rangle \rangle \equiv \lambda y. \langle A \mid x:=\langle S \mid y:=T \rangle \rangle$]

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

Examples

[$(\lambda x.x + 2) ' 3 \equiv 5$]

[$(\lambda x.x + y) ' 3 \equiv 3 + y$]

See also

[`case(x, y, z)`] Section A.6

[`N`] Section A.23

[`<x | y:=z>`] Section A.28

[`variable x`] Section A.45

[`x ' y`] Section A.47

[<code>$\lambda x.y$</code>]	lambda x dot y
[<code>SubLambdaXX</code>]	rule SubLambdaXX
[<code>SubLambdaXYFreeFor</code>]	rule SubLambdaXYFreeFor
[<code>SubLambdaXY</code>]	rule SubLambdaXY
[<code>LambdaSub</code>]	rule LambdaSub

A.21 minimum x comma y end [min(x, y)]**Parse tree****Sort**

[min(x, y)] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[min(x, y)] denotes the least of the two values [x] and [y].

Definition

$$\left[\min(x, y) \doteq x < y \begin{cases} x \\ y \end{cases} \right].$$

Mac rules

$$\left[\begin{array}{l} \text{Mac rule DefMinimumXCommaYEnd : } \min(x, y) \equiv x < y \begin{cases} x \\ y \end{cases} \\ \text{Mac rule SubMinimumXCommaYEnd : } \langle \min(\mathcal{A}, \mathcal{B}) \mid x := \mathcal{S} \rangle \equiv \min(\langle \mathcal{A} \mid \\ x := \mathcal{S} \rangle, \langle \mathcal{B} \mid x := \mathcal{S} \rangle) \end{array} \right]$$

Examples

$$\begin{array}{lll} [\min(2, 3)] & \equiv & 2] \\ [\min(3, 2)] & \equiv & 2] \\ [\min(2, 2)] & \equiv & 2] \\ [\min(2, \infty)] & \equiv & 2] \\ [\min(2, -\infty)] & \equiv & -\infty] \end{array}$$

See also

[x ∧ y] Section A.46

[min(x, y)]	minimum x comma y end
[DefMinimumXCommaYEnd]	rule DefMinimumXCommaYEnd
[SubMinimumXCommaYEnd]	rule SubMinimumXCommaYEnd

A.22 minus x [-x]**Parse tree****Sort**

[-x] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[-x] changes the sign of [x]. In particular, [-x] may be used to change the sign of an infinity.

Type table

x	B	D	X	E	N	Z	Z⁺	Z⁻
-x	X	D	X	X	Z	Z	Z⁻	Z⁺
x	D_m^∞	D_m	D_m^∞	D_∞	D_∞	D_∞^∞	T	F
-x	D_m^∞	D_m	D_m^∞	D_∞^∞	D_∞	D_∞^∞	X	X

Mac rules

[**Mac rule SubMinusX** : $\langle -\mathcal{A} \mid x:=\mathcal{S} \rangle \equiv -\langle \mathcal{A} \mid x:=\mathcal{S} \rangle$]
 [**Mac rule MinusStrict** : $-\perp \equiv \perp$]

Examples

[--5 \equiv 5]
 [-- ∞ \equiv ∞]

See also

[∞] Section A.19
 [+x] Section A.26
 [x - y] Section A.73

[-x]	minus x
[SubMinusX]	rule SubMinusX
[MinusStrict]	rule MinusStrict

A.23 nil map [N]

Parse tree

N

Sort

[N] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[N] denotes the nil map. [N] is one of the fundamental constructs of map theory.

Mac rules

[Mac rule SubNilMap : $\langle N \mid x := S \rangle \equiv N$]

Examples

[T \equiv N]
[F \equiv N . N]

See also

[case(x, y, z)] Section A.6
[$\lambda x.y$] Section A.20
[x ' y] Section A.47

[N] nil map
[SubNilMap] rule SubNilMap

A.24 not x $[\neg x]$ **Parse tree****Sort**

$[\neg x]$ is an operator (i.e. it may occur in the syntax tree of a term).

Description

$[\neg x]$ is true if $[x]$ is false and vice versa. In other words, $[\neg x]$ is the logical negation of $[x]$.

Type table

x	B	D	X	E	N	Z	Z ⁺	Z ⁻
$\neg x$	B	X	X	X	X	X	X	X
x	D _m [∞]	D _m	D _m ^{-∞}	D _∞ [∞]	D _∞ [∞]	D _∞ ^{-∞}	T	F
$\neg x$	X	X	X	X	X	X	F	T

Mac rules

[**Mac rule SubNotX** : $\langle \neg \mathcal{A} \mid x := \mathcal{S} \rangle \equiv \neg \langle \mathcal{A} \mid x := \mathcal{S} \rangle$]

[**Mac rule NotStrict** : $\neg \perp \equiv \perp$]

Examples

$[\neg T \equiv F]$

$[\neg F \equiv T]$

$[\neg \perp \equiv \perp]$

See also

[T] Section A.43

[F] Section A.13

$[\neg x]$	not x
[SubNotX]	rule SubNotX
[NotStrict]	rule NotStrict

A.25 parenthesis x end [(x)]**Parse tree****Sort**

[(x)] is a macro.

Description

[(x)] means the same as [x]. [(x)] is useful for controlling the order of computation like in [2 · (3 + 4)] where the parenthesis forces addition to occur before multiplication.

Definition

[(x)] ≡ x .

Examples

$$[(2 \cdot 3) + 4] \equiv 10]$$

$$[2 \cdot (3 + 4)] \equiv 14]$$

See also

[x ≡ y] Section A.72

A.26 plus x [+x]

Parse tree



Sort

[+x] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[+x] leaves a decimal fraction unchanged. It may be used to present positive quantities in a format similar to that of negative quantities (e.g. [+2] versus [-2] and [+∞] versus [-∞]).

Type table

x	B	D	X	E	N	Z	Z ⁺	Z ⁻
+x	X	D	X	X	N	Z	Z ⁺	Z ⁻
x	D _m [∞]	D _m	D _m ^{-∞}	D _∞ [∞]	D _∞	D _∞ ^{-∞}	T	F
+x	D _m [∞]	D _m	D _m ^{-∞}	D _∞ [∞]	D _∞	D _∞ ^{-∞}	X	X

Mac rules

[Mac rule SubPlusX : ⟨ +A | x:=S ⟩ ≡ +⟨ A | x:=S ⟩]

[Mac rule PlusStrict : +⊥ ≡ ⊥]

Examples

[+5 ≡ 5]

[+∞ ≡ ∞]

See also

[∞] Section A.19

[-x] Section A.22

[x + y] Section A.80

[+x]	plus x
[SubPlusX]	rule SubPlusX
[PlusStrict]	rule PlusStrict

A.27 precision x [#x]

Parse tree



Sort

[#x] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[#x] denotes the precision (number of significant digits) of decimal fractions. If [x] is a decimal fraction then [#x] is a positive integer or [+∞]. Exact decimal fractions have precision [+∞].

Type table

x	B	D	X	E	N	Z	Z⁺	Z⁻
#x	X	D	X	X	D_∞[∞]	D_∞[∞]	D_∞[∞]	D_∞[∞]
x	D_m[∞]	D_m	D_m^{-∞}	D_∞[∞]	D_∞[∞]	D_∞^{-∞}	T	F
#x	Z⁺	Z⁺	Z⁺	D_∞[∞]	D_∞[∞]	D_∞[∞]	X	X

Mac rules

[**Mac rule SubPrecisionX** : ⟨#A | x:=S⟩ ≡ #⟨A | x:=S⟩]

[**Mac rule PrecisionStrict** : #⊥ ≡ ⊥]

Examples

[#1.234F ≡ 4]

[#1.234 ≡ ∞]

[#∞ ≡ ∞]

[#(∞@3) ≡ 3]

See also

[x@y] Section A.85

[#x]	precision x
[SubPrecisionX]	rule SubPrecisionX
[PrecisionStrict]	rule PrecisionStrict

A.28 substitute x where y is z end [$\langle \mathcal{A} \mid x := \mathcal{B} \rangle$]

Parse tree



Sort

[$\langle \mathcal{A} \mid x := \mathcal{B} \rangle$] is an operator (i.e. it may occur in the syntax tree of a term).

Binding

[$\langle * \mid * := * \rangle$].

Description

[$\langle \mathcal{A} \mid x := \mathcal{B} \rangle$] equals [\mathcal{A}] in which all free occurrences of [x] have value [\mathcal{B}].

Mac rules

[**Mac rule SubXX** : $\langle x \mid x := \mathcal{S} \rangle \equiv \mathcal{S}$]
 [**Mac rule SubXY** : if distinct (x, y) then $\langle y \mid x := \mathcal{S} \rangle \equiv y$]
 [**Mac rule SubXSubX** : $\langle \langle \mathcal{A} \mid x := \mathcal{B} \rangle \mid x := \mathcal{S} \rangle \equiv \langle \mathcal{A} \mid x := \langle \mathcal{B} \mid x := \mathcal{S} \rangle \rangle$]
 [**Mac rule SubXSubY** : if distinct(x, y) then $\langle \langle \mathcal{A} \mid x := \mathcal{B} \rangle \mid y := \langle \mathcal{S} \mid x := \mathcal{T} \rangle \rangle \equiv \langle \langle \mathcal{A} \mid y := \langle \mathcal{S} \mid x := \mathcal{T} \rangle \rangle \mid x := \langle \mathcal{B} \mid y := \langle \mathcal{S} \mid x := \mathcal{T} \rangle \rangle$]
 [**Mac rule SubXSubYNotFree** : if distinct(x, y) \wedge notfree(x, \mathcal{S}) then $\langle \langle \mathcal{A} \mid x := \mathcal{B} \rangle \mid y := \mathcal{S} \rangle \equiv \langle \langle \mathcal{A} \mid y := \mathcal{S} \rangle \mid x := \langle \mathcal{B} \mid y := \mathcal{S} \rangle \rangle$]
 [**Mac rule Substitution** : if Substitution(\mathcal{A}, \mathcal{B}) then $\mathcal{A} \equiv \mathcal{B}$]
 [**Mac rule SubNumeral** : if numeral(y) then $\langle y \mid x := \mathcal{S} \rangle \equiv y$]
 ([Substitution(\mathcal{A}, \mathcal{B})] says that [\mathcal{A}] and [\mathcal{B}] can be made identical by performing substitutions and renaming bound variables).

Examples

[$\langle x + 2 \mid x := 3 \rangle \equiv 5$]
 [$\langle x + y \mid x := 3 \rangle \equiv 3 + y$]

[$\langle x \mid y := z \rangle$]	substitute x where y is z end
[SubXX]	rule SubXX
[SubXY]	rule SubXY
[SubXSubX]	rule SubXSubX
[SubXSubY]	rule SubXSubY
[SubXSubYNotFree]	rule SubXSubYNotFree
[Substitution]	rule Substitution
[SubNumeral]	rule SubNumeral

See also

- [$\lambda x.y$] Section A.20
- [**variable** x] Section A.45

A.29 the class of sets [Set]

Parse tree

Set

Sort

[Set] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[Set] is the class of all sets. $[x \in \text{Set}]$ is true if and only if $[x]$ is a set. For certain, technical reasons, [Set] itself is not a set.

Mac rules

[Mac rule SubSet : $\langle \text{Set} \mid x:=S \rangle \equiv \text{Set}$]

Examples

$[T \in \text{Set}] \equiv T$
 $[B \in \text{Set}] \equiv T$
 $[D \in \text{Set}] \equiv T$
 $[X \in \text{Set}] \equiv T$
 $[117 \in \text{Set}] \equiv F$
 $[\text{Set} \in \text{Set}] \equiv F$

See also

$[\ell]$ Section A.8
 $[x \in y]$ Section A.50

[Set] the class of sets
 [SubSet] rule SubSet

A.30 the set of decimal fractions [D]

Parse tree

D

Sort

[D] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[D] is the set of decimal fractions. $[x \in \mathbf{D}]$ is true if and only if $[x]$ is a classical map that weakly represents a decimal fraction. Decimal fractions include exact fractions like [1.1] and floating fractions like [1.1F]. Decimal fractions include infinities like $[\infty]$ and $[-\infty@4]$.

Mac rules

[Mac rule SubD : $\langle \mathbf{D} \mid x := S \rangle \equiv \mathbf{D}$]

[Mac rule SetD : $\mathbf{D} \in \text{Set}$]

[Mac rule TypeNumeralInD : if x is a numeral that represents a decimal fraction then $x \in \mathbf{D}$]

[Mac rule SubtypeDC : $x \in \mathbf{D} \vdash \ell' x$]

[Mac rule ElimD : $x \in \mathbf{D} \vdash \overline{\mathbf{D}}' x$]

[Mac rule IntroD : $\ell' x \vdash \overline{\mathbf{D}}' x \vdash x \in \mathbf{D}$]

[Mac rule SubtypeNotDC : $x \notin \mathbf{D} \vdash \ell' x$]

[Mac rule ElimNotD : $x \notin \mathbf{D} \vdash \neg \overline{\mathbf{D}}' x$]

[Mac rule IntroNotD : $\ell' x \vdash \neg \overline{\mathbf{D}}' x \vdash x \notin \mathbf{D}$]

[D]	the set of decimal fractions
[SubD]	rule SubD
[SetD]	rule SetD
[TypeNumeralInD]	rule TypeNumeralInD
[SubtypeDC]	rule SubtypeDC
[ElimD]	rule ElimD
[IntroD]	rule IntroD
[SubtypeNotDC]	rule SubtypeNotDC
[ElimNotD]	rule ElimNotD
[IntroNotD]	rule IntroNotD

Examples

$[0 \in \mathbf{D}]$	\equiv	\mathbf{T}
$[1 \in \mathbf{D}]$	\equiv	\mathbf{T}
$[117 \in \mathbf{D}]$	\equiv	\mathbf{T}
$[-1 \in \mathbf{D}]$	\equiv	\mathbf{T}
$[1.1 \in \mathbf{D}]$	\equiv	\mathbf{T}
$[117\mathbf{F} \in \mathbf{D}]$	\equiv	\mathbf{T}
$[\mathbf{T} \in \mathbf{D}]$	\equiv	\mathbf{F}
$[\bullet \in \mathbf{D}]$	\equiv	\mathbf{F}
$[0 :: 1 \in \mathbf{D}]$	\equiv	\mathbf{F}
$[\perp \in \mathbf{D}]$	\equiv	\perp

See also

$[\mathbf{D}_x]$	Section A.31
$[\mathbf{D}_x^\infty]$	Section A.34
$[\mathbf{D}_x^{-\infty}]$	Section A.36
$[\mathbf{Z}]$	Section A.35
$[x \in y]$	Section A.50

A.31 the set of decimal fractions of precision x end $[D_x]$

Parse tree



Sort

$[D_x]$ is an operator (i.e. it may occur in the syntax tree of a term).

Description

$[D_x]$ is the set of decimal fractions of precision $[x]$ whose magnitude differs from $[\infty]$ and $[-\infty]$.

Mac rules

[Mac rule SubDm : $\langle D_A \mid x := S \rangle \equiv D_{\langle A \mid x := S \rangle}$]

[Mac rule SetDi : $D_\infty \in \text{Set}$]

[Mac rule TypeNumeralInDm : if x is a numeral that represents a floating fraction of precision m then $x \in D_m$]

[Mac rule TypeNumeralInDi : if x is a numeral that represents an exact fraction then $x \in D_\infty$]

[Mac rule TypeNumeralNotInDm : if x is a numeral that does not represent a floating fraction of precision m then $x \notin D_m$]

[Mac rule TypeNumeralNotInDi : if x is a numeral that does not represent an exact fraction then $x \in D_\infty$]

[Mac rule SubtypeDiD : $x \in D_\infty \vdash x \in D$]

[Mac rule SubtypeDmD : $m \in \mathbb{Z}^+ \vdash x \in D_m \vdash x \in D$]

[Mac rule SetDm : $m \in \mathbb{Z}^+ \vdash D_m \in \text{Set}$]

$[D_x]$	the set of decimal fractions of precision x end
[SubDm]	rule SubDm
[SetDi]	rule SetDi
[TypeNumeralInDm]	rule TypeNumeralInDm
[TypeNumeralInDi]	rule TypeNumeralInDi
[TypeNumeralNotInDm]	rule TypeNumeralNotInDm
[TypeNumeralNotInDi]	rule TypeNumeralNotInDi
[SubtypeDiD]	rule SubtypeDiD
[SubtypeDmD]	rule SubtypeDmD
[SetDm]	rule SetDm

Examples

$[117 \in \mathbf{D}_\infty]$	\equiv	\mathbf{T}
$[117 \in \mathbf{D}_3]$	\equiv	\mathbf{F}
$[117\mathbf{F} \in \mathbf{D}_3]$	\equiv	\mathbf{T}
$[117\mathbf{F} \in \mathbf{D}_\infty]$	\equiv	\mathbf{F}
$[\infty \in \mathbf{D}_\infty]$	\equiv	\mathbf{F}
$[\mathbf{T} \in \mathbf{D}]$	\equiv	\mathbf{F}
$[\bullet \in \mathbf{D}]$	\equiv	\mathbf{F}
$[0 :: 1 \in \mathbf{D}]$	\equiv	\mathbf{F}
$[\perp \in \mathbf{D}]$	\equiv	\perp

See also

$[\mathbf{D}]$	Section A.30
$[x \in y]$	Section A.50

A.32 the set of exceptions $[X]$

Parse tree

$[X]$

Sort

$[X]$ is an operator (i.e. it may occur in the syntax tree of a term).

Description

$[X]$ is the set of exceptions. $[x \in X]$ is true if and only if $[x]$ is a classical map that weakly represents an exception.

Mac rules

[Mac rule SubX : $\langle X \mid x := \mathcal{S} \rangle \equiv X$]
 [Mac rule TypeNumeralNotInX : if x is a numeral then $x \notin X$]
 [Mac rule SetX : $X \in \text{Set}$]
 [Mac rule SubtypeXC : $x \in X \vdash \ell' x$]
 [Mac rule ElimX : $x \in X \vdash \overline{X}' x$]
 [Mac rule IntroX : $\ell' x \vdash \overline{X}' x \vdash x \in X$]
 [Mac rule SubtypeNotXC : $x \notin X \vdash \ell' x$]
 [Mac rule ElimNotX : $x \notin X \vdash \neg \overline{X}' x$]
 [Mac rule IntroNotX : $\ell' x \vdash \neg \overline{X}' x \vdash x \notin X$]

Examples

[$\bullet \in X$ \equiv \top]
 [$\top \in X$ \equiv F]
 [$\text{F} \in X$ \equiv F]
 [$117 \in X$ \equiv F]
 [$0 :: 1 \in X$ \equiv F]
 [$\perp \in X$ \equiv \perp]

[X]	the set of exceptions
[SubX]	rule SubX
[TypeNumeralNotInX]	rule TypeNumeralNotInX
[SetX]	rule SetX
[SubtypeXC]	rule SubtypeXC
[ElimX]	rule ElimX
[IntroX]	rule IntroX
[SubtypeNotXC]	rule SubtypeNotXC
[ElimNotX]	rule ElimNotX
[IntroNotX]	rule IntroNotX

See also

[$x \in y$] Section A.50

A.33 the set of false maps $[F]$

Parse tree

$[F]$

Sort

$[F]$ is an operator (i.e. it may occur in the syntax tree of a term).

Description

$[F]$ is the set of classical maps that weakly represent falsehood.

Mac rules

[Mac rule **SubF** : $\langle F \mid x := S \rangle \equiv F$]
 [Mac rule **TypeNumeralNotInF** : if x is a numeral then $x \notin F$]
 [Mac rule **SetF** : $F \in \text{Set}$]
 [Mac rule **SubtypeFB** : $x \in F \vdash x \in B$]
 [Mac rule **SubtypeFC** : $x \in F \vdash \ell' x$]
 [Mac rule **ElimF** : $x \in F \vdash \overline{F'} x$]
 [Mac rule **IntroF** : $\ell' x \vdash \overline{F'} x \vdash x \in F$]
 [Mac rule **SubtypeNotFC** : $x \notin F \vdash \ell' x$]
 [Mac rule **ElimNotF** : $x \notin F \vdash \neg \overline{F'} x$]
 [Mac rule **IntroNotF** : $\ell' x \vdash \neg \overline{F'} x \vdash x \notin F$]

Examples

[$T \in F$ \equiv F]
 [$F \in F$ \equiv T]
 [$117 \in F$ \equiv F]
 [$\bullet \in F$ \equiv F]
 [$0 :: 1 \in F$ \equiv F]
 [$\perp \in F$ \equiv \perp]

	[F]	the set of false maps
	[SubF]	rule SubF
[TypeNumeralNotInF]		rule TypeNumeralNotInF
	[SetF]	rule SetF
	[SubtypeFB]	rule SubtypeFB
	[SubtypeFC]	rule SubtypeFC
	[ElimF]	rule ElimF
	[IntroF]	rule IntroF
[SubtypeNotFC]		rule SubtypeNotFC
	[ElimNotF]	rule ElimNotF
	[IntroNotF]	rule IntroNotF

See also

- [**B**] Section A.42
- [$x \in y$] Section A.50

A.34 the set of infinities of precision x end $[D_x^\infty]$

Parse tree



Sort

$[D_x^\infty]$ is an operator (i.e. it may occur in the syntax tree of a term).

Description

$[D_x^\infty]$ is the set of decimal fractions of precision $[x]$ whose magnitude equals $[+\infty]$.

Mac rules

[Mac rule SubIi : $\langle D_{\mathcal{A}}^\infty \mid x := S \rangle \equiv D_{\langle \mathcal{A} \mid x := S \rangle}^\infty$]

[Mac rule TypeNumeralNotInIm : if x is a numeral then $m \in \mathbf{Z}^+ \vdash x \notin D_m^\infty$]

[Mac rule TypeNumeralNotInIi : if x is a numeral then $x \notin D_\infty^\infty$]

[Mac rule SetIi : $D_\infty^\infty \in \mathbf{Set}$]

[Mac rule SetIm : $m \in \mathbf{Z}^+ \vdash D_m^\infty \in \mathbf{Set}$]

[Mac rule SubtypeIiD : $x \in D_\infty^\infty \vdash x \in \mathbf{D}$]

[Mac rule SubtypeImD : $m \in \mathbf{Z}^+ \vdash x \in D_m^\infty \vdash x \in \mathbf{D}$]

Examples

$[\infty \in D_\infty^\infty]$	\equiv	T
$[\infty \in D_3^\infty]$	\equiv	F
$[\infty@3 \in D_\infty^\infty]$	\equiv	F
$[\infty@3 \in D_3^\infty]$	\equiv	T
$[-\infty \in D_\infty^\infty]$	\equiv	F
$[117 \in D_\infty^\infty]$	\equiv	F
$[T \in D_\infty^\infty]$	\equiv	F
$[\bullet \in D_\infty^\infty]$	\equiv	F
$[0 :: 1 \in D_\infty^\infty]$	\equiv	F
$[\perp \in D_\infty^\infty]$	\equiv	\perp

$[D_x^\infty]$	the set of infinities of precision x end
[SubIi]	rule SubIi
[TypeNumeralNotInIm]	rule TypeNumeralNotInIm
[TypeNumeralNotInIi]	rule TypeNumeralNotInIi
[SetIi]	rule SetIi
[SetIm]	rule SetIm
[SubtypeIiD]	rule SubtypeIiD
[SubtypeImD]	rule SubtypeImD

See also

- [**D**] Section A.30
- [$x \in y$] Section A.50

A.35 the set of integers $[Z]$

Parse tree

$[Z]$

Sort

$[Z]$ is an operator (i.e. it may occur in the syntax tree of a term).

Description

$[Z]$ is the set of integers. $[x \in Z]$ is true if and only if $[x]$ is a classical map that weakly represents an integer. A number is an *integer* if it is (1) an exact decimal fractions and (2) a whole number.

Mac rules

[Mac rule SubZ : $\langle Z \mid x := S \rangle \equiv Z$]

[Mac rule TypeNumeralInZ : if x is a numeral that represents an integer then $x \in Z$]

[Mac rule TypeNumeralNotInZ : if x is a numeral that does not represent an integer then $x \notin Z$]

[Mac rule SetZ : $Z \in \text{Set}$]

[Mac rule SubtypeZDi : $x \in Z \vdash x \in D_\infty$]

[Mac rule SubtypeZD : $x \in Z \vdash x \in D$]

Examples

$[0 \in Z]$	\equiv	T
$[1 \in Z]$	\equiv	T
$[117 \in Z]$	\equiv	T
$[-1 \in Z]$	\equiv	T
$[1.1 \in Z]$	\equiv	F
$[117F \in Z]$	\equiv	F
$[T \in Z]$	\equiv	F
$[\bullet \in Z]$	\equiv	F
$[0 :: 1 \in Z]$	\equiv	F
$[\perp \in Z]$	\equiv	\perp

$[Z]$	the set of integers
	integer
$[SubZ]$	rule SubZ
$[TypeNumeralInZ]$	rule TypeNumeralInZ
$[TypeNumeralNotInZ]$	rule TypeNumeralNotInZ
$[SetZ]$	rule SetZ
$[SubtypeZDi]$	rule SubtypeZDi
$[SubtypeZD]$	rule SubtypeZD

See also

- [**D**] Section A.30
- [**N**] Section A.37
- [**Z**⁻] Section A.38
- [**Z**⁺] Section A.39
- [**x** ∈ **y**] Section A.50

A.36 the set of minus infinities of precision x end $[D_x^{-\infty}]$

Parse tree



Sort

$[D_x^{-\infty}]$ is an operator (i.e. it may occur in the syntax tree of a term).

Description

$[D_x^{-\infty}]$ is the set of decimal fractions of precision $[x]$ whose magnitude equals $[-\infty]$.

Mac rules

[Mac rule SubMm : $\langle D_{\mathcal{A}}^{-\infty} \mid x := \mathcal{S} \rangle \equiv D_{\langle \mathcal{A} \mid x := \mathcal{S} \rangle}^{-\infty}$]

[Mac rule TypeNumeralNotInMm : if x is a numeral then $m \in \mathbf{Z}^+ \vdash x \notin D_m^{-\infty}$]

[Mac rule TypeNumeralNotInMi : if x is a numeral then $x \notin D_{\infty}^{-\infty}$]

[Mac rule SetMi : $D_{\infty}^{-\infty} \in \mathbf{Set}$]

[Mac rule SetMm : $m \in \mathbf{Z}^+ \vdash D_m^{-\infty} \in \mathbf{Set}$]

[Mac rule SubtypeMiD : $x \in D_{\infty}^{-\infty} \vdash x \in \mathbf{D}$]

[Mac rule SubtypeMmD : $m \in \mathbf{Z}^+ \vdash x \in D_m^{-\infty} \vdash x \in \mathbf{D}$]

$[D_x^{-\infty}]$	the set of minus infinities of precision x end
[SubMm]	rule SubMm
[TypeNumeralNotInMm]	rule TypeNumeralNotInMm
[TypeNumeralNotInMi]	rule TypeNumeralNotInMi
[SetMi]	rule SetMi
[SetMm]	rule SetMm
[SubtypeMiD]	rule SubtypeMiD
[SubtypeMmD]	rule SubtypeMmD

Examples

$[-\infty \in \mathbf{D}_{\infty}^{-\infty}]$	\equiv	\mathbf{T}
$[-\infty \in \mathbf{D}_3^{-\infty}]$	\equiv	\mathbf{F}
$[-\infty @ 3 \in \mathbf{D}_{\infty}^{-\infty}]$	\equiv	\mathbf{F}
$[-\infty @ 3 \in \mathbf{D}_3^{-\infty}]$	\equiv	\mathbf{T}
$[\infty \in \mathbf{D}_{\infty}^{-\infty}]$	\equiv	\mathbf{F}
$[117 \in \mathbf{D}_{\infty}^{-\infty}]$	\equiv	\mathbf{F}
$[\mathbf{T} \in \mathbf{D}_{\infty}^{-\infty}]$	\equiv	\mathbf{F}
$[\bullet \in \mathbf{D}_{\infty}^{-\infty}]$	\equiv	\mathbf{F}
$[0 :: 1 \in \mathbf{D}_{\infty}^{-\infty}]$	\equiv	\mathbf{F}
$[\perp \in \mathbf{D}_{\infty}^{-\infty}]$	\equiv	\perp

See also

$[\mathbf{D}]$	Section A.30
$[x \in y]$	Section A.50

A.37 the set of natural numbers $[\mathbf{N}]$

Parse tree

$\boxed{\mathbf{N}}$

Sort

$[\mathbf{N}]$ is an operator (i.e. it may occur in the syntax tree of a term).

Description

$[\mathbf{N}]$ is the set of natural numbers. $[x \in \mathbf{N}]$ is true if and only if $[x]$ is a classical map that weakly represents a natural number. A number is *natural* if it is (1) an exact decimal fractions and (2) a whole number and (3) non-negative.

Mac rules

[**Mac rule SubN** : $\langle \mathbf{N} \mid x := S \rangle \equiv \mathbf{N}$]

[**Mac rule TypeNumeralInN** : if x is a numeral that represents a natural number **then** $x \in \mathbf{N}$]

[**Mac rule TypeNumeralNotInN** : if x is a numeral that does not represent a natural number **then** $x \notin \mathbf{N}$]

[**Mac rule SetN** : $\mathbf{N} \in \mathbf{Set}$]

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

[**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 PeanoD'** : $x \in \mathbf{N} \vdash (x = 0) \in \mathbf{F} \vdash x - 1 \in \mathbf{N}$]

[**Mac rule PeanoE** : $x \in \mathbf{N} \vdash (x = 0) \in \mathbf{F} \vdash x^{-+} = x$]

[**Mac rule Induction** :

$n \in \mathbf{N} \rightarrow (n = 0) \in \mathbf{T} \rightarrow \mathcal{A} \vdash$

$n \in \mathbf{N} \rightarrow (n = 0) \in \mathbf{F} \rightarrow \langle \mathcal{A} \mid n := n - 1 \rangle \rightarrow \mathcal{A} \vdash$

$n \in \mathbf{N} \rightarrow \mathcal{A}$]

$[\mathbf{N}]$	the set of natural numbers natural number
[SubN]	rule SubN
[TypeNumeralInN]	rule TypeNumeralInN
[TypeNumeralNotInN]	rule TypeNumeralNotInN
[SetN]	rule SetN
[SubtypeNZ]	rule SubtypeNZ
[PeanoA]	rule PeanoA
[PeanoB]	rule PeanoB
[PeanoC]	rule PeanoC
[PeanoD]	rule PeanoD
[PeanoD']	rule PeanoD'
[PeanoE]	rule PeanoE
[Induction]	rule Induction

Examples

$[0 \in \mathbf{N} \equiv \mathbf{T}]$
 $[1 \in \mathbf{N} \equiv \mathbf{T}]$
 $[117 \in \mathbf{N} \equiv \mathbf{T}]$
 $[-1 \in \mathbf{N} \equiv \mathbf{F}]$
 $[1.1 \in \mathbf{Z} \equiv \mathbf{F}]$
 $[117\mathbf{F} \in \mathbf{N} \equiv \mathbf{F}]$
 $[\mathbf{T} \in \mathbf{N} \equiv \mathbf{F}]$
 $[\bullet \in \mathbf{N} \equiv \mathbf{F}]$
 $[0 :: 1 \in \mathbf{N} \equiv \mathbf{F}]$
 $[\perp \in \mathbf{N} \equiv \perp]$

See also

$[\mathbf{Z}]$ Section A.35
 $[x \in y]$ Section A.50

A.38 the set of negative integers $[Z^-]$

Parse tree

Z^-

Sort

$[Z^-]$ is an operator (i.e. it may occur in the syntax tree of a term).

Description

$[Z^-]$ is the set of negative integers. $[x \in Z^-]$ is true if and only if $[x]$ is a classical map that weakly represents a negative integer. A number is a *negative integer* if it is (1) an exact decimal fractions and (2) a whole number and (3) negative (i.e. strongly less than zero).

Mac rules

[Mac rule SubZm : $\langle Z^- \mid x := S \rangle \equiv Z^-$]

[Mac rule TypeNumeralInZm : if x is a numeral that represents a negative integer then $x \in Z^-$]

[Mac rule TypeNumeralNotInZm : if x is a numeral that does not represent a negative integer then $x \notin Z^-$]

[Mac rule SetZm : $Z^- \in \text{Set}$]

[Mac rule SubtypeZmZ : $x \in Z^- \vdash x \in Z$]

Examples

$[0 \in Z^-]$	\equiv	F
$[1 \in Z^-]$	\equiv	F
$[-117 \in Z^-]$	\equiv	T
$[-1 \in Z^-]$	\equiv	T
$[-1.1 \in Z^-]$	\equiv	F
$[-117F \in Z^-]$	\equiv	F
$[T \in Z^-]$	\equiv	F
$[\bullet \in Z^-]$	\equiv	F
$[0 :: 1 \in Z^-]$	\equiv	F
$[\perp \in Z^-]$	\equiv	\perp

$[Z^-]$	the set of negative integers
	negative integer
[SubZm]	rule SubZm
[TypeNumeralInZm]	rule TypeNumeralInZm
[TypeNumeralNotInZm]	rule TypeNumeralNotInZm
[SetZm]	rule SetZm
[SubtypeZmZ]	rule SubtypeZmZ

See also

- [**Z**] Section A.35
- [$x \in y$] Section A.50

A.39 the set of positive integers $[Z^+]$

Parse tree

Z^+

Sort

$[Z^+]$ is an operator (i.e. it may occur in the syntax tree of a term).

Description

$[Z^+]$ is the set of positive integers. $[x \in Z^+]$ is true if and only if $[x]$ is a classical map that weakly represents a positive integer. A number is a *positive integer* if it is (1) an exact decimal fractions and (2) a whole number and (3) positive (i.e. strongly greater than zero).

Mac rules

[Mac rule SubZp : $\langle Z^+ \mid x := S \rangle \equiv Z^+$]

[Mac rule TypeNumeralInZp : if x is a numeral that represents a positive integer then $x \in Z^+$]

[Mac rule TypeNumeralNotInZp : if x is a numeral that does not represent a positive integer then $x \notin Z^+$]

[Mac rule SetZp : $Z^+ \in \text{Set}$]

[Mac rule SubtypeZpN : $x \in Z^+ \vdash x \in \mathbf{N}$]

[Mac rule SubtypeZpZ : $x \in Z^+ \vdash x \in \mathbf{Z}$]

Examples

$[0 \in Z^+]$	\equiv	F
$[1 \in Z^+]$	\equiv	T
$[117 \in Z^+]$	\equiv	T
$[-1 \in Z^+]$	\equiv	F
$[1.1 \in Z^+]$	\equiv	F
$[117F \in Z^+]$	\equiv	F
$[T \in Z^+]$	\equiv	F
$[\bullet \in Z^+]$	\equiv	F
$[0 :: 1 \in Z^+]$	\equiv	F
$[\perp \in Z^+]$	\equiv	\perp

$[Z^+]$	the set of positive integers
	positive integer
[SubZp]	rule SubZp
[TypeNumeralInZp]	rule TypeNumeralInZp
[TypeNumeralNotInZp]	rule TypeNumeralNotInZp
[SetZp]	rule SetZp
[SubtypeZpN]	rule SubtypeZpN
[SubtypeZpZ]	rule SubtypeZpZ

See also

- [**Z**] Section A.35
- [$x \in y$] Section A.50

A.40 the set of the empty list [E]

Parse tree

E

Sort

[E] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[E] is the set of classical maps that weakly represent the empty list.

Mac rules

[Mac rule SubE : $\langle \mathbf{E} \mid x := S \rangle \equiv \mathbf{E}$]
 [Mac rule TypeNumeralNotInE : if x is a numeral then $x \notin \mathbf{E}$]
 [Mac rule SetE : $\mathbf{E} \in \text{Set}$]
 [Mac rule SubtypeEC : $x \in \mathbf{E} \vdash \ell' x$]
 [Mac rule ElimE : $x \in \mathbf{E} \vdash \overline{\mathbf{E}}' x$]
 [Mac rule IntroE : $\ell' x \vdash \overline{\mathbf{E}}' x \vdash x \in \mathbf{E}$]
 [Mac rule SubtypeNotEC : $x \notin \mathbf{E} \vdash \ell' x$]
 [Mac rule ElimNotE : $x \notin \mathbf{E} \vdash \neg \overline{\mathbf{E}}' x$]
 [Mac rule IntroNotE : $\ell' x \vdash \neg \overline{\mathbf{E}}' x \vdash x \notin \mathbf{E}$]

Examples

[$\langle \rangle \in \mathbf{E}$ \equiv T]
 [$117 \in \mathbf{E}$ \equiv F]
 [$\bullet \in \mathbf{E}$ \equiv F]
 [$0 :: 1 \in \mathbf{E}$ \equiv F]
 [$\perp \in \mathbf{E}$ \equiv \perp]

See also

[$x \in y$] Section A.50
 [$x \times y$] Section A.51

[E]	the set of the empty list
[SubE]	rule SubE
[TypeNumeralNotInE]	rule TypeNumeralNotInE
[SetE]	rule SetE
[SubtypeEC]	rule SubtypeEC
[ElimE]	rule ElimE
[IntroE]	rule IntroE
[SubtypeNotEC]	rule SubtypeNotEC
[ElimNotE]	rule ElimNotE
[IntroNotE]	rule IntroNotE

A.41 the set of true maps [T]

Parse tree

T

Sort

[T] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[T] is the set of classical maps that weakly represent truth. [$x \in T$] is true if and only if [$x \equiv T$].

Mac rules

[**Mac rule SubT** : $\langle T \mid x := S \rangle \equiv T$]
 [**Mac rule TypeNumeralNotInT** : if x is a numeral then $x \notin T$]
 [**Mac rule SetT** : $T \in \text{Set}$]
 [**Mac rule SubtypeTB** : $x \in T \vdash x \in B$]
 [**Mac rule SubtypeTC** : $x \in T \vdash \ell'x$]
 [**Mac rule ElimT** : $x \in T \vdash \overline{T}'x$]
 [**Mac rule IntroT** : $\ell'x \vdash \overline{T}'x \vdash x \in T$]
 [**Mac rule ElimTT** : $x \in T \vdash x$]
 [**Mac rule IntroTT** : $x \vdash x \in T$]
 [**Mac rule SubtypeNotTC** : $x \notin T \vdash \ell'x$]
 [**Mac rule ElimNotT** : $x \notin T \vdash \neg \overline{T}'x$]
 [**Mac rule IntroNotT** : $\ell'x \vdash \neg \overline{T}'x \vdash x \notin T$]

	[T]	the set of true maps
	[SubT]	rule SubT
[TypeNumeralNotInT]		rule TypeNumeralNotInT
	[SetT]	rule SetT
	[SubtypeTB]	rule SubtypeTB
	[SubtypeTC]	rule SubtypeTC
	[ElimT]	rule ElimT
	[IntroT]	rule IntroT
	[ElimTT]	rule ElimTT
	[IntroTT]	rule IntroTT
[SubtypeNotTC]		rule SubtypeNotTC
	[ElimNotT]	rule ElimNotT
	[IntroNotT]	rule IntroNotT

Examples

$[T \in \mathbf{T} \equiv T]$
 $[F \in \mathbf{T} \equiv F]$
 $[117 \in \mathbf{T} \equiv F]$
 $[\bullet \in \mathbf{T} \equiv F]$
 $[0 :: 1 \in \mathbf{T} \equiv F]$
 $[\perp \in \mathbf{T} \equiv \perp]$

See also

$[B]$ Section A.42
 $[x \in y]$ Section A.50

A.42 the set of truth values $[B]$

Parse tree

$[B]$

Sort

$[B]$ is an operator (i.e. it may occur in the syntax tree of a term).

Description

$[B]$ is the set of truth values. $[x \in B]$ is true if and only if $[x]$ is a classical map that weakly represents truth or weakly represents falsehood.

Mac rules

[Mac rule **SubB** : $\langle B \mid x := S \rangle \equiv B$]
 [Mac rule **TypeNumeralNotInB** : if x is a numeral then $x \notin B$]
 [Mac rule **SetB** : $B \in \text{Set}$]
 [Mac rule **Cases** : $x \in T \rightarrow \mathcal{A} \equiv B \vdash x \in F \rightarrow \mathcal{A} \equiv B \vdash x \in B \rightarrow \mathcal{A} \equiv B$]
 [Mac rule **Tautology** : if x is a tautology then x]
 [Mac rule **SubtypeBC** : $x \in B \vdash \ell' x$]
 [Mac rule **ElimB** : $x \in B \vdash \overline{B}' x$]
 [Mac rule **IntroB** : $\ell' x \vdash \overline{B}' x \vdash x \in B$]
 [Mac rule **SubtypeNotBC** : $x \notin B \vdash \ell' x$]
 [Mac rule **ElimNotB** : $x \notin B \vdash \overline{B}' x$]
 [Mac rule **IntroNotB** : $\ell' x \vdash \overline{B}' x \vdash x \notin B$]

Examples

$[T \in B] \equiv T$
 $[F \in B] \equiv T$
 $[117 \in B] \equiv F$
 $[\bullet \in B] \equiv F$
 $[0 :: 1 \in B] \equiv F$
 $[\perp \in B] \equiv \perp$

$[B]$	the set of truth values
$[SubB]$	rule SubB
$[TypeNumeralNotInB]$	rule TypeNumeralNotInB
$[SetB]$	rule SetB
$[Cases]$	rule Cases
$[Tautology]$	rule Tautology
$[SubtypeBC]$	rule SubtypeBC
$[ElimB]$	rule ElimB
$[IntroB]$	rule IntroB
$[SubtypeNotBC]$	rule SubtypeNotBC
$[ElimNotB]$	rule ElimNotB
$[IntroNotB]$	rule IntroNotB

See also

- [F] Section A.33
- [T] Section A.41
- [$x \in y$] Section A.50

A.43 true [T]**Parse tree**

T

Sort

[T] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[T] denotes truth. If a computer can determine within finite time that a term is true, then the value of the term is [T]. If Map sees a term in square brackets (such as [2 + 2 = 4]) then Map computes the value of the term and protests if the value differs from [T]. If a term [A] is stated as a lemma, then it is understood that the lemma says [A ≡ T].

Mac rules

[**Mac rule SubTrue** : $\langle T \mid x := S \rangle \equiv T$]

[**Mac rule TypeTInB** : $T \in B$]

[**Mac rule TypeTInT** : $T \in T$]

Examples

[2 + 2 = 4 ≡ T]

[2 ≠ F ≡ T]

See also

[F] Section A.13

[if(x, y, z)] Section A.18

[T]	true
[SubTrue]	rule SubTrue
[TypeTInB]	rule TypeTInB
[TypeTInT]	rule TypeTInT

A.44 tuple x end [⟨x⟩]

Parse tree



Sort

[⟨x⟩] is a macro.

Description

[⟨x⟩] denotes a one-element tuple that has [x] as its sole element. The term reduction rule [⟨x,y⟩ \rightarrow x :: ⟨y⟩] allows [⟨x⟩] to be combined with commas to form tuples of arbitrary length.

Definition

[⟨x⟩] \equiv x :: ⟨⟩.

Examples

[⟨1⟩	\equiv	1 :: ⟨⟩]
[⟨1,2⟩	\equiv	1 :: 2 :: ⟨⟩]
[⟨1,2,3⟩	\equiv	1 :: 2 :: 3 :: ⟨⟩]
[⟨1,2,3,4⟩	\equiv	1 :: 2 :: 3 :: 4 :: ⟨⟩]
[⟨1,2,3,4⟩ head	\equiv	1]
[⟨1,2,3,4⟩ tail	\equiv	⟨2,3,4⟩]
[⟨1,2,3,4⟩ pair	\equiv	T]
[⟨1,2,3,4⟩ atom	\equiv	F]

See also

[⟨⟩]	Section A.10
[x atom]	Section A.49
[x head]	Section A.64
[x pair]	Section A.78
[x :: y]	Section A.79
[x tail]	Section A.95

A.45 variable x [variable x]

Parse tree



Sort

[variable x] is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

[variable \mathcal{A}] declares the construct [\mathcal{A}] to be a variable. Any construct introduced by [variable \mathcal{A}] becomes a null-ary outfix operator. A statement like [variable [L[n]]] introduces infinitely many variables in one go (c.f. Section 11.6).

Examples

[variable a]	[variable n]	[variable \mathcal{A}]	[variable \mathcal{N}]
[variable b]	[variable o]	[variable \mathcal{B}]	[variable \mathcal{O}]
[variable c]	[variable p]	[variable \mathcal{C}]	[variable \mathcal{P}]
[variable d]	[variable q]	[variable \mathcal{D}]	[variable \mathcal{Q}]
[variable e]	[variable r]	[variable \mathcal{E}]	[variable \mathcal{R}]
[variable f]	[variable s]	[variable \mathcal{F}]	[variable \mathcal{S}]
[variable g]	[variable t]	[variable \mathcal{G}]	[variable \mathcal{T}]
[variable h]	[variable u]	[variable \mathcal{H}]	[variable \mathcal{U}]
[variable i]	[variable v]	[variable \mathcal{I}]	[variable \mathcal{V}]
[variable j]	[variable w]	[variable \mathcal{J}]	[variable \mathcal{W}]
[variable k]	[variable x]	[variable \mathcal{K}]	[variable \mathcal{X}]
[variable l]	[variable y]	[variable \mathcal{L}]	[variable \mathcal{Y}]
[variable m]	[variable z]	[variable \mathcal{M}]	[variable \mathcal{Z}]
[variable *]			

See also

[construct x]	Section A.9
[x \doteq y]	Section A.58

[variable x] variable x

A.46 x and y [x ∧ y]

Parse tree



Sort

[x ∧ y] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[x ∧ y] denotes the minimum of [x] and [y]. In particular, for truth values [x] and [y], [x ∧ y] denotes “[x] and [y]”. For truth values [x] and [y], [x ∧ y] coincides with [x · y]. [x ∧ y ≡ •] if [x] and [y] are decimal fractions of different precision.

Type table

∧	B	D	X	E	∧	N	Z	Z ⁺	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	Z ⁺	N	Z	Z ⁺	Z ⁻
E	X	X	X	X	Z ⁻	Z ⁻	Z ⁻	Z ⁻	Z ⁻
∧	D _m [∞]	D _m	D _m ^{-∞}	D _∞	D _∞	D _∞ ^{-∞}	D _n [∞]	D _n	D _n ^{-∞}
D _m [∞]	D _m [∞]	D _m	D _m ^{-∞}	X	X	X	X	X	X
D _m	D _m	D _m	D _m ^{-∞}	X	X	X	X	X	X
D _m ^{-∞}	D _m [∞]	D _m ^{-∞}	D _m ^{-∞}	X	X	X	X	X	X
D _∞	X	X	X	D _∞	D _∞	D _∞ ^{-∞}	∧	T	F
D _∞ ^{-∞}	X	X	X	D _∞	D _∞	D _∞ ^{-∞}	T	T	F
D _∞	X	X	X	D _∞	D _∞	D _∞ ^{-∞}	F	F	F

Mac rules

[Mac rule StrictAndX : ⊥ ∧ x ≡ ⊥]

[Mac rule XAndStrict : x ∧ ⊥ ≡ ⊥]

[Mac rule SubXAndY : ⟨A ∧ B | x:=S⟩ ≡ ⟨A | x:=S⟩ ∧ ⟨B | x:=S⟩]

[x ∧ y]	x and y
[StrictAndX]	rule StrictAndX
[XAndStrict]	rule XAndStrict
[SubXAndY]	rule SubXAndY

See also

[$\min(x, y)$]	Section A.21
[$x \vee y$]	Section A.77
[$x \cdot y$]	Section A.97

A.47 x apply y $[x \ ' \ y]$ **Parse tree****Sort**

$[x \ ' \ y]$ is an operator (i.e. it may occur in the syntax tree of a term).

Description

$[x \ ' \ y]$ denotes the function $[x]$ applied to the argument $[y]$. $[x \ ' \ y]$ is one of the fundamental constructs of map theory.

Mac rules

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

[**Mac rule ApplyNil** : $N \ ' \ y \equiv N$]

[**Mac rule ApplyT** : $T \ ' \ y \equiv T$]

[**Mac rule ApplyLambda** : $(\lambda u.v) \ ' \ y \equiv \langle v \mid u := y \rangle$]

[**Mac rule ApplyBottom** : $\perp \ ' \ y \equiv \perp$]

Examples

$[(\lambda x.x + 2) \ ' \ 3] \equiv 5$]

$[(\lambda x.x + y) \ ' \ 3] \equiv 3 + y$]

See also

[$\text{case}(x, y, z)$] Section A.6

[N] Section A.23

[$\lambda x.y$] Section A.20

$[x \ ' \ y]$	x apply y
[SubXApplyY]	rule SubXApplyY
[ApplyNil]	rule ApplyNil
[ApplyT]	rule ApplyT
[ApplyLambda]	rule ApplyLambda
[ApplyBottom]	rule ApplyBottom

A.48 x associates as y [$x \overset{\curvearrowright}{\rightarrow} y$]**Parse tree****Sort**

[$x \overset{\curvearrowright}{\rightarrow} y$] is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

[$\mathcal{A} \overset{\curvearrowright}{\rightarrow} \mathcal{B}$] states that the term [\mathcal{A}] should be interpreted like term [\mathcal{B}]. [\mathcal{A}] must consist of two operators of equal priority that are applied to three variables. [\mathcal{B}] must consist of the same two operators applied to the same three variables plus a parenthesis which indicates how the operator associates.

Examples

- [$x + y + z \overset{\curvearrowright}{\rightarrow} (x + y) + z$] makes [$+$]^o left associative.
- [$x + y - z \overset{\curvearrowright}{\rightarrow} (x + y) - z$] is identical to the associativity rule above since [$+$]^o and [$-$]^o have equal priority.
- [$x :: y :: z \overset{\curvearrowright}{\rightarrow} x :: (y :: z)$] makes [$::$]^o right associative.
- [$x^y z \overset{\curvearrowright}{\rightarrow} x^{(y^z)}$] makes [x^y] associate in a way that is neither left nor right associativity.

See also

[$x \overset{\curvearrowleft}{\rightarrow} y$] Section A.87

[$x \overset{\circ}{\rightarrow} y$] Section A.96

A.49 x atom [x atom]

Parse tree



Sort

[x atom] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[x atom] is true if [x] is an atom, i.e. not a pair. Truth values, exceptions, decimal fractions, and the empty list are atoms. [\perp] is neither an atom nor a pair.

Type table

x	B	D	X	E	N	Z	Z⁺	Z⁻
x atom	T	T	T	T	T	T	T	T
x	D_m[∞]	D_m	D_m^{-∞}	D_∞[∞]	D_∞	D_∞^{-∞}	T	F
x atom	T	T	T	T	T	T	T	T

Mac rules

[**Mac rule PairIsNotAtom** : (x :: y) atom \equiv F]
 [**Mac rule SubXAtom** : $\langle \mathcal{A} \text{ atom} \mid x:=S \rangle \equiv \langle \mathcal{A} \mid x:=S \rangle \text{ atom}$]
 [**Mac rule StrictAtom** : $\perp \text{ atom} \equiv \perp$]

Examples

[2.0F atom \equiv T]
 [$\langle x, y, z \rangle \text{ atom} \equiv$ F]

See also

[x pair] Section A.78
 [x :: y] Section A.79

[x atom]	x atom
[PairIsNotAtom]	rule PairIsNotAtom
[SubXAtom]	rule SubXAtom
[StrictAtom]	rule StrictAtom

A.50 x belongs to y [$x \in y$]

Parse tree



Sort

[$x \in y$] is an operator (i.e. it may occur in the syntax tree of a term).

Description

If [y] is a set then the following hold: [$x \in y$] equals [\top] if [x] belongs to [y]. [$x \in y$] equals [F] if [x] is classical and [x] does not belong to [y]. [$x \in y$] equals [\perp] if [x] is non-classical. The definition

$$[x \notin y] \doteq \neg x \in y$$

makes it easy to express non-membership.

Mac rules

- [**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** : $y \in \mathbf{Set} \vdash \ell' x \vdash (x \in y) \in \mathbf{B}$]
- [**Mac rule XBelongsToStrict** : $x \in \perp \equiv \perp$]
- [**Mac rule StrictBelongsToY** : $y \in \mathbf{Set} \vdash \perp \in y \equiv \perp$]
- [**Mac rule SubXBelongsToY** : $\langle \mathcal{A} \in \mathcal{B} \mid x := \mathcal{S} \rangle \equiv \langle \mathcal{A} \mid x := \mathcal{S} \rangle \in \langle \mathcal{B} \mid x := \mathcal{S} \rangle$]

Examples

$$\begin{array}{lcl} [2 \in \mathbf{N}] & \equiv & \top \\ [-2 \in \mathbf{N}] & \equiv & F \\ [\perp \in \mathbf{N}] & \equiv & \perp \\ [2 \in \perp] & \equiv & \perp \end{array}$$

[$x \in y$]	x belongs to y
[$x \notin y$]	x not in y
[SubtypeSetC]	rule SubtypeSetC
[SubtypeNotSetC]	rule SubtypeNotSetC
[TypeCInSet]	rule TypeCInSet
[XBelongsToStrict]	rule XBelongsToStrict
[StrictBelongsToY]	rule StrictBelongsToY
[SubXBelongsToY]	rule SubXBelongsToY

See also

- [ℓ] Section A.8
- [**Set**] Section A.29
- [**D**] Section A.30
- [**X**] Section A.32
- [**E**] Section A.40
- [**B**] Section A.42
- [$x \times y$] Section A.51

A.51 x cartesian y $[x \times y]$

Parse tree



Sort

$[x \times y]$ is an operator (i.e. it may occur in the syntax tree of a term).

Description

$[x \times y]$ is the set of pairs whose first component belongs to $[x]$ and whose second component belongs to $[y]$.

Mac rules

[**Mac rule SubXCartesianY** : $\langle \mathcal{A} \times \mathcal{B} \mid x := S \rangle \equiv \langle \mathcal{A} \mid x := S \rangle \times \langle \mathcal{B} \mid x := S \rangle$]
 [**Mac rule SetXCartesianY** : $A \in \mathbf{Set} \vdash B \in \mathbf{Set} \vdash A \times B \in \mathbf{Set}$]
 [**Mac rule ElimCartesianPair** : $A \in \mathbf{Set} \vdash B \in \mathbf{Set} \vdash x \in A \times B \vdash \overline{\mathbf{P}}'x$]
 [**Mac rule ElimCartesianHead** : $A \in \mathbf{Set} \vdash B \in \mathbf{Set} \vdash x \in A \times B \vdash x$
 $\text{head} \in A$]
 [**Mac rule ElimCartesianTail** : $A \in \mathbf{Set} \vdash B \in \mathbf{Set} \vdash x \in A \times B \vdash x$
 $\text{tail} \in B$]
 [**Mac rule IntroCartesian** : $A \in \mathbf{Set} \vdash B \in \mathbf{Set} \vdash \ell'x \vdash \overline{\mathbf{P}}'x \vdash x \text{ head} \in$
 $A \vdash x \text{ tail} \in B \vdash x \in A \times B$]
 [**Mac rule IntroCartesianPair** : $A \in \mathbf{Set} \vdash B \in \mathbf{Set} \vdash a \in A \vdash b \in B \vdash a ::$
 $b \in A \times B$]

Examples

[$1 :: T \in \mathbf{N} \times \mathbf{B} \equiv T$]
 [$T :: 1 \in \mathbf{N} \times \mathbf{B} \equiv F$]

See also

[**E**] Section A.40
 [$x \in y$] Section A.50
 [$x :: y$] Section A.79

[$x \times y$]	x cartesian y
[SubXCartesianY]	rule SubXCartesianY
[SetXCartesianY]	rule SetXCartesianY
[ElimCartesianPair]	rule ElimCartesianPair
[ElimCartesianHead]	rule ElimCartesianHead
[ElimCartesianTail]	rule ElimCartesianTail
[IntroCartesian]	rule IntroCartesian
[IntroCartesianPair]	rule IntroCartesianPair

A.52 x colon y [x : y]**Parse tree****Sort**

[x : y] is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

[x : y] means the same as [y], and also introduces [x] as shorthand for the meaning of [y]. In particular, if [y] is an instantiation statement, then [x] is introduced as shorthand for the conclusion of [y].

Examples

[x: TailPair▷(1 :: 2) tail ≡ 2] means the same as [TailPair▷(1 :: 2) tail ≡ 2] and introduces [x] as shorthand for [(1 :: 2) tail ≡ 2].

See also

[x **proof of** y:z] Section A.83
 [x▷ y] Section A.56

A.53 x comma y $[x, y]$ **Parse tree****Sort**

$[x, y]$ is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

$[x, y]$ has no meaning of its own. It has a meaning by virtue of term reduction rules (See Appendix A.3). If `Map` is asked to compute an expression whose syntax tree contains $[x, y]$ then `Map` protests because the expression is not a term.

Definition

$[\text{construct } x, y]$.

Examples

$$\begin{aligned}
 [\langle x, y, z \rangle] &\equiv x :: y :: z :: \langle \rangle] \\
 [x, y < z] &\equiv x < z \wedge y < z] \\
 [x < y, z] &\equiv x < y \wedge x < z]
 \end{aligned}$$

See also

$[\text{construct } x]$ Section A.9
 $[\langle x \rangle]$ Section A.44

A.54 x computational equal y [x = y]

Parse tree



Sort

[x = y] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[x = y] compares for equality. If [x] and [y] are decimal fractions, then [x = y] disregards precision. If [x] and [y] are truth values then [x = y] coincides with [x ⇔ y].

Type table

=	B	D	X	E	=	N	Z	Z ⁺	Z ⁻
B	B	F	X	F	N	B	B	B	F
D	F	B	X	F	Z	B	B	B	B
X	X	X	X	X	Z ⁺	B	B	B	F
E	F	F	X	T	Z ⁻	F	B	F	B
=	D _m [∞]	D _m	D _m ^{-∞}	D _∞ [∞]	D _∞	D _∞ ^{-∞}	D _n [∞]	D _n	D _n ^{-∞}
D _m [∞]	T	F	F	T	F	F	T	F	F
D _m	F	B	F	F	B	F	F	B	F
D _m ^{-∞}	F	F	T	F	F	T	F	F	T
D _∞ [∞]	T	F	F	T	F	F	=	T	F
D _∞	F	B	F	F	B	F	T	T	F
D _∞ ^{-∞}	F	F	T	F	F	T	F	F	T

Mac rules

- [Mac rule **StrictComputationalEqualX** : ⊥ = x ≡ ⊥]
- [Mac rule **XComputationalEqualStrict** : x = ⊥ ≡ ⊥]
- [Mac rule **EqualPair** : x :: y = u :: v ≡ x = u ∧ y = v]
- [Mac rule **SubXComputationalEqualY** : ⟨A = B | x:=S⟩ ≡ ⟨A | x:=S⟩ = ⟨B | x:=S⟩]

[x = y]	x computational equal y
[StrictComputationalEqualX]	rule StrictComputationalEqualX
[XComputationalEqualStrict]	rule XComputationalEqualStrict
[EqualPair]	rule EqualPair
[SubXComputationalEqualY]	rule SubXComputationalEqualY

Examples

$$\begin{aligned} [\langle 1, 2 \rangle = \langle 1, 2 \rangle] &\equiv \top] \\ [\langle 1, 2 \rangle = \langle 1, 3 \rangle] &\equiv \text{F}] \\ [\langle 1, 2 \rangle = \langle 1, 2, 3 \rangle] &\equiv \text{F}] \\ [\langle 1, 2 \rangle = \langle 1, \bullet \rangle] &\equiv \bullet] \\ [\langle 1, 2 \rangle = \langle 1, \perp \rangle] &\equiv \perp] \end{aligned}$$

See also

$$\begin{aligned} [x \neq y] &\text{ Section A.55} \\ [x < y] &\text{ Section A.93} \end{aligned}$$

A.55 x computational unequal y [x ≠ y]

Parse tree



Sort

[x ≠ y] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[x ≠ y] compares for inequality. If [x] and [y] are decimal fractions, then [x ≠ y] disregards precision. If [x] and [y] are truth values then [x ≠ y] coincides with exclusive “or”.

Type table

≠	B	D	X	E	≠	N	Z	Z ⁺	Z ⁻
B	B	T	X	T	N	B	B	B	T
D	T	B	X	T	Z	B	B	B	B
X	X	X	X	X	Z ⁺	B	B	B	T
E	T	T	X	F	Z ⁻	T	B	T	B
≠	D _m [∞]	D _m	D _m ^{-∞}	D _∞ [∞]	D _∞ ^{-∞}	D _∞ ^{-∞}	D _n [∞]	D _n	D _n ^{-∞}
D _m [∞]	F	T	T	F	T	T	F	T	T
D _m	T	B	T	T	B	T	T	B	T
D _m ^{-∞}	T	T	F	T	T	F	T	T	F
D _∞ [∞]	F	T	T	F	T	T	≠	T	F
D _∞ ^{-∞}	T	B	T	T	B	T	T	F	T
D _∞ ^{-∞}	T	T	F	T	T	F	F	T	F

Mac rules

- [Mac rule StrictComputationalUnequalX : ⊥ ≠ x ≡ ⊥]
- [Mac rule XComputationalUnequalStrict : x ≠ ⊥ ≡ ⊥]
- [Mac rule UnequalPair : x :: y ≠ u :: v ≡ x ≠ u ∨ y ≠ v]
- [Mac rule SubXComputationalUnequalY : ⟨A ≠ B | x:=S⟩ ≡ ⟨A | x:=S⟩ ≠ ⟨B | x:=S⟩]

[x ≠ y]	x computational unequal y
[StrictComputationalUnequalX]	rule StrictComputationalUnequalX
[XComputationalUnequalStrict]	rule XComputationalUnequalStrict
[UnequalPair]	rule UnequalPair
[SubXComputationalUnequalY]	rule SubXComputationalUnequalY

Examples

$[\langle 1, 2 \rangle \neq \langle 1, 2 \rangle]$	\equiv	F
$[\langle 1, 2 \rangle \neq \langle 1, 3 \rangle]$	\equiv	T
$[\langle 1, 2 \rangle \neq \langle 1, 2, 3 \rangle]$	\equiv	T
$[\langle 1, 2 \rangle \neq \langle 1, \bullet \rangle]$	\equiv	•
$[\langle 1, 2 \rangle \neq \langle 1, \perp \rangle]$	\equiv	\perp

See also

$[x = y]$	Section A.54
$[x < y]$	Section A.93

A.56 x concludes y [$x \triangleright y$] $[x \triangleright y]$ **Parse tree****Sort**

$[x \triangleright y]$ is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

$[x \triangleright y]$ is used to form argumentations from inference rules and equations.

Examples

$[x \equiv y \vdash y \equiv z \vdash x \equiv z \triangleright \langle \rangle \text{ head} \equiv \langle \rangle \triangleright \langle \rangle \equiv \langle \rangle \text{ tail} \triangleright \langle \rangle \text{ head} \equiv \langle \rangle \text{ tail}]$ states that $[\langle \rangle \text{ head} \equiv \langle \rangle \text{ tail}]$ follows from $[\langle \rangle \text{ head} \equiv \langle \rangle]$ and $[\langle \rangle \equiv \langle \rangle \text{ tail}]$ according to the inference rules $[x \equiv y \vdash y \equiv z \vdash x \equiv z]$. Another way to state this is

$$[\text{Transitivity} \triangleright \langle \rangle \text{ head} \equiv \langle \rangle \triangleright \langle \rangle \equiv \langle \rangle \text{ tail} \triangleright \langle \rangle \text{ head} \equiv \langle \rangle \text{ tail}]$$

since Transitivity is shorthand for $[x \equiv y \vdash y \equiv z \vdash x \equiv z]$.

See also

$[x \text{ proof of } y:z]$	Section A.83
$[x \supseteq y]$	Section A.75
$[x : y]$	Section A.52
$[x \vdash y]$	Section A.68

A.57 x deduces y $[x \rightarrow y]$ $[x \rightarrow y]$ **Parse tree****Sort**

$[x \rightarrow y]$ is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

$[x \rightarrow y]$ is used to state that $[x]$ implies $[y]$ when $[x]$ is a term and $[y]$ is an equation. $[x \rightarrow y]$ is shorthand for an equation.

Mac rules

[**Mac rule ModusPonens** : $x \equiv \top \vdash x \rightarrow y \equiv z \vdash y \equiv z$]

Examples

$[x \in \mathbf{Z} \rightarrow y \in \mathbf{Z} \rightarrow x + y \equiv y + x]$ states that if $[x]$ and $[y]$ are integers then $[x + y]$ equals $[y + x]$.

See also

$[x \vdash y]$ Section A.68
 $[x \Rightarrow y]$ Section A.67
 $[x \supseteq y]$ Section A.75

$[x \rightarrow y]$ x deduces y
[**ModusPonens**] rule ModusPonens

A.58 x defined equal y [$x \doteq y$]**Parse tree****Sort**

[$x \doteq y$] is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

Among other, [$\mathcal{A} \doteq \mathcal{B}$] introduces a new operator [\mathcal{A}], the definition axiom [$\mathcal{A} \equiv \mathcal{B}$], and the reduction rule [$\mathcal{A} \dot{\rightarrow} \mathcal{B}$]. Contrary to constructs introduced by [$\mathcal{A} \doteq \mathcal{B}$], constructs introduced by [$\mathcal{A} \equiv \mathcal{B}$] can occur in syntax trees.

Examples

[$\infty\text{-list}(n) \doteq n :: \infty\text{-list}(n + 1)$]

See also

[construct x]	Section A.9
[variable x]	Section A.45
[$x \equiv y$]	Section A.60
[$x \dot{\rightarrow} y$]	Section A.72
[$x \dot{=} y$]	Section A.76

A.59 x divide y [x/y]

Parse tree



Sort

[x/y] is an operator (i.e. it may occur in the syntax tree of a term).

Description

If [x] and [y] are floating fractions of the same precision, then [x/y] denotes [x] divided by [y] rounded to the precision of [x] and [y] using round to even. [x/+y], [x/-y], [x/0y], and [x/2y] are versions of [x/y] that use round to plus infinity, minus infinity, zero, and even, respectively. [x/2y] is identical to [x/y]. If [x] and [y] are exact fractions or if they have different precision then [x/y ≡ •].

Type table

/	B	D	X	E	/	N	Z	Z ⁺	Z ⁻
B	X	X	X	X	N	X	X	X	X
D	X	-	X	X	Z	X	X	X	X
X	X	X	X	X	Z ⁺	X	X	X	X
E	X	X	X	X	Z ⁻	X	X	X	X
/	D _m [∞]	D _m	D _m ^{-∞}	D _∞ [∞]	D _∞	D _∞ ^{-∞}	D _n [∞]	D _n	D _n ^{-∞}
D _m [∞]	X	X	X	X	X	X	X	X	X
D _m	X	-	X	X	X	X	X	X	X
D _m ^{-∞}	X	X	X	X	X	X	X	X	X
D _∞ [∞]	X	X	X	X	X	X	/	T	F
D _∞	X	X	X	X	X	X	T	X	X
D _∞ ^{-∞}	X	X	X	X	X	X	F	X	X

Mac rules

[Mac rule StrictDivideX : ⊥/x ≡ ⊥]

[Mac rule XDivideStrict : x/⊥ ≡ ⊥]

[Mac rule SubXDivideY : ⟨A/B | x:=S⟩ ≡ ⟨A | x:=S⟩/⟨B | x:=S⟩]

-
- [x/y] x divide y
 - [StrictDivideX] rule StrictDivideX
 - [XDivideStrict] rule XDivideStrict
 - [SubXDivideY] rule SubXDivideY

[**Mac rule DivideNonzero** : $m \in \mathbf{Z}^+ \vdash x \in \mathbf{D}_m \vdash y \in \mathbf{D}_m \vdash (y = 0) \in \mathbf{F} \vdash x/y \in \mathbf{D}_m$]

[**Mac rule DivideZero** : $x \in \mathbf{D} \vdash y \in \mathbf{D} \vdash (y = 0) \in \mathbf{T} \vdash x/y \in \mathbf{X}$]

See also

[$x // y$] Section A.69

[$x @ y$] Section A.85

[DivideNonzero] rule DivideNonzero
 [DivideZero] rule DivideZero

A.60 x equal y [$x \equiv y$]

Parse tree



Sort

[$x \equiv y$] is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

[$\mathcal{A} \equiv \mathcal{B}$] states that [\mathcal{A}] equals [\mathcal{B}]. [$\mathcal{A} \equiv \mathcal{B}$] has no value since it is a statement construct. Instead, [$\mathcal{A} \equiv \mathcal{B}$] may “hold” or “fail”. [$\mathcal{A} \equiv \mathcal{B}$] holds if [\mathcal{A}] equals [\mathcal{B}] for all values of free variables in [\mathcal{A}] and [\mathcal{B}]. [$\mathcal{A} \equiv \mathcal{B}$] fails if [\mathcal{A}] is unequal to [\mathcal{B}] for some values of the free variables in [\mathcal{A}] and [\mathcal{B}].

Mac rules

[**Mac antirule Contradiction** : $\perp \equiv \top$]

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

[**Mac rule Reflexivity** : $x \equiv x$]

If [$u \doteq v$] or [$u \dot{=} v$] and if [$x \equiv y$] is a replacement or reverse replacement of [$u \equiv v$], then [$x \equiv y$]:

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

If [x] and [y] are closed and compute to the same value, then [$x \equiv y$]:

[**Mac rule Computation** : if closed(x) \wedge closed(y) \wedge equal(x, y) then $x \equiv y$]

[**Mac rule Commutativity** : $x \equiv y \vdash y \equiv x$]

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

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

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

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

[$x \equiv y$]	x equal y
[Contradiction]	rule Contradiction
[Repetition]	rule Repetition
[Reflexivity]	rule Reflexivity
[Definition]	rule Definition
[Computation]	rule Computation
[Commutativity]	rule Commutativity
[Transitivity]	rule Transitivity
[Rename]	rule Rename
[Replace]	rule Replace
[Reverse]	rule Reverse

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

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

Examples

[$(x :: y)$ head	\equiv	x]	holds
[$(x :: y)$ tail	\equiv	x]	fails
[\perp	\equiv	\perp]	holds
[1	\equiv	\perp]	fails
[$1.0F$	\equiv	$1.00F$]	fails

See also

[$x \doteq y$] Section A.58

[$x = y$] Section A.54

[Replace']	rule Replace'
[Reverse']	rule Reverse'

A.61 x exceptional [x?]

Parse tree



Sort

[x?] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[x?] is true if [x] is exceptional.

Type table

x	B	D	X	E	N	Z	Z⁺	Z⁻
x!	F	F	T	F	F	F	F	F
x	D_m[∞]	D_m	D_m^{-∞}	D_∞[∞]	D_∞	D_∞^{-∞}	T	F
x!	F	F	F	F	F	F	F	F

Mac rules

[Mac rule **SubXExceptional** : $\langle \mathcal{A}^? \mid x:=S \rangle \equiv \langle \mathcal{A} \mid x:=S \rangle^?$]

[Mac rule **StrictExceptional** : $\perp^? \equiv \perp$]

Examples

[$\infty^?$ \equiv **F**]

[$(1/0)^?$ \equiv **T**]

See also

[•] Section A.11

[x?]	x exceptional
[SubXExceptional]	rule SubXExceptional
[StrictExceptional]	rule StrictExceptional

A.62 x faculty [x!]

Parse tree



Sort

[x!] is an operator (i.e. it may occur in the syntax tree of a term).

Description

For positive integers [x], [x!] denotes the product of all numbers from [1] to [x], inclusive. [0! ≡ 1].

Definition

[n! ≡ if(n = 0, 1, n · (n - 1)!)].

Type table

x	B	D	X	E	N	Z	Z⁺	Z⁻
x!	X	-	X	X	Z⁺	-	Z⁺	-
x	D_m[∞]	D_m	D_m^{-∞}	D_∞[∞]	D_∞	D_∞^{-∞}	T	F
x!	X	-	X	X	-	X	X	X

Mac rules

[**Mac rule DefXFaculty** : n! ≡ if(n = 0, 1, n · (n - 1)!)]

[**Mac rule SubXFaculty** : ⟨A! | x:=S⟩ ≡ ⟨A | x:=S⟩!]

[**Mac rule StrictFaculty** : ⊥! ≡ ⊥]

See also

[x ≐ y] Section A.62

[x!]	x faculty
[DefXFaculty]	rule DefXFaculty
[SubXFaculty]	rule SubXFaculty
[StrictFaculty]	rule StrictFaculty

A.63 x greater priority y [$x \succ y$]**Parse tree****Sort**

[$x \succ y$] is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

[$\mathcal{A} \succ \mathcal{B}$] states that the construct [\mathcal{A}] has greater priority than the construct [\mathcal{B}] when computing expressions. Statements of form [$\mathcal{A} \succ \mathcal{B}$] affect the process of forming parse trees. They affect syntax trees indirectly in that they affect the parse trees from which the syntax trees are constructed. In [$\mathcal{A} \succ \mathcal{B}$], [\mathcal{A}] and [\mathcal{B}] must have the form of an operator applied to variables. [$\mathcal{A} \succ \mathcal{B}$] has no effect on outfix operators.

Examples

[$x \cdot y \succ x + y$] makes Map interpret [$2 \cdot 3 + 4$] as [$(2 \cdot 3) + 4$].

See also

[$x \rightarrow y$] Section A.48

[$x \doteq y$] Section A.87

A.64 `x head` [`x head`]

Parse tree



Sort

[`x head`] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[`x head`] denotes the first element of a pair [`x`] or (which is the same) the first element of a list. If [`x`] is a truth value, a decimal fraction, an exception or the empty list, then [`x head`] equals [`x`].

Type table

<code>x</code>	B	D	X	E	N	Z	Z⁺	Z⁻
<code>x head</code>	B	D	X	E	N	Z	Z⁺	Z⁻
<code>x</code>	D_m[∞]	D_m	D_m^{-∞}	D_∞[∞]	D_∞	D_∞^{-∞}	T	F
<code>x head</code>	D_m[∞]	D_m	D_m^{-∞}	D_∞[∞]	D_∞	D_∞^{-∞}	T	F

Mac rules

[**Mac rule HeadEmpty** : $\langle \rangle \text{ head} \equiv \langle \rangle$]
 [**Mac rule HeadPair** : $\langle x :: y \rangle \text{ head} \equiv x$]
 [**Mac rule HeadBottom** : $\perp \text{ head} \equiv \perp$]
 [**Mac rule SubXHead** : $\langle \mathcal{A} \text{ head} \mid x := \mathcal{S} \rangle \equiv \langle \mathcal{A} \mid x := \mathcal{S} \rangle \text{ head}$]

Examples

[$\langle 1, 2, 3, 4, 5 \rangle \text{ head} \equiv 1$]

See also

[`x :: y`] Section A.79
 [`x tail`] Section A.95

[<code>x head</code>]	<code>x head</code>
[<code>HeadEmpty</code>]	rule <code>HeadEmpty</code>
[<code>HeadPair</code>]	rule <code>HeadPair</code>
[<code>HeadBottom</code>]	rule <code>HeadBottom</code>
[<code>SubXHead</code>]	rule <code>SubXHead</code>

A.65 x if and only if y [$x \Leftrightarrow y$]

Parse tree



Sort

[$x \Leftrightarrow y$] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[$x \Leftrightarrow y$] expresses “[x] if and only if [y]”. [$x \Leftrightarrow y$] coincides with [$x = y$] for truth values [x] and [y].

Type table

\Leftrightarrow	B	D	X	E	\Leftrightarrow	N	Z	Z⁺	Z⁻
B	B	X	X	X	N	X	X	X	X
D	X	X	X	X	Z	X	X	X	X
X	X	X	X	X	Z⁺	X	X	X	X
E	X	X	X	X	Z⁻	X	X	X	X
\Leftrightarrow	D_m^∞	D_m	$D_m^{-\infty}$	D_∞^∞	$D_\infty^{-\infty}$	D_∞^∞	D_n^∞	D_n	$D_n^{-\infty}$
D_m^∞	X	X	X	X	X	X	X	X	X
D_m	X	X	X	X	X	X	X	X	X
$D_m^{-\infty}$	X	X	X	X	X	X	X	X	X
D_∞^∞	X	X	X	X	X	X	\Leftrightarrow	T	F
$D_\infty^{-\infty}$	X	X	X	X	X	X	T	T	F
D_∞^∞	X	X	X	X	X	X	F	F	T

Mac rules

[**Mac rule StrictIfAndOnlyIfX** : $\perp \Leftrightarrow x \equiv \perp$]

[**Mac rule XIfAndOnlyIfStrict** : $x \Leftrightarrow \perp \equiv \perp$]

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

[$x \Leftrightarrow y$]	x if and only if y
[StrictIfAndOnlyIfX]	rule StrictIfAndOnlyIfX
[XIfAndOnlyIfStrict]	rule XIfAndOnlyIfStrict
[SubXIfAndOnlyIfY]	rule SubXIfAndOnlyIfY

See also

- [$x = y$] Section A.54
- [$x \Rightarrow y$] Section A.67
- [$x \Leftarrow y$] Section A.66

A.66 x implied by y [$x \Leftarrow y$]

Parse tree



Sort

[$x \Leftarrow y$] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[$x \Leftarrow y$] expresses “[x] follows from [y]”. [$x \Leftarrow y$] coincides with [$x \geq y$] for truth values [x] and [y].

Type table

\Leftarrow	B	D	X	E	\Leftarrow	N	Z	Z⁺	Z⁻
B	B	X	X	X	N	X	X	X	X
D	X	X	X	X	Z	X	X	X	X
X	X	X	X	X	Z⁺	X	X	X	X
E	X	X	X	X	Z⁻	X	X	X	X
\Leftarrow	D_m^∞	D_m	$D_m^{-\infty}$	D_∞^∞	$D_\infty^{-\infty}$	D_∞^∞	D_n^∞	D_n	$D_n^{-\infty}$
D_m^∞	X	X	X	X	X	X	X	X	X
D_m	X	X	X	X	X	X	X	X	X
$D_m^{-\infty}$	X	X	X	X	X	X	X	X	X
D_∞^∞	X	X	X	X	X	X	\Leftarrow	T	F
$D_\infty^{-\infty}$	X	X	X	X	X	X	T	T	T
D_∞^∞	X	X	X	X	X	X	F	F	T

Mac rules

[Mac rule **StrictImpliedByX** : $\perp \Leftarrow x \equiv \perp$]

[Mac rule **XImpliedByStrict** : $x \Leftarrow \perp \equiv \perp$]

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

[$x \Leftarrow y$]	x implied by y
[StrictImpliedByX]	rule StrictImpliedByX
[XImpliedByStrict]	rule XImpliedByStrict
[SubXImpliedByY]	rule SubXImpliedByY

See also

- [$x \Leftrightarrow y$] Section A.65
- [$x \Rightarrow y$] Section A.67
- [$x \geq y$] Section A.98

A.67 x implies y [$x \Rightarrow y$]

Parse tree



Sort

[$x \Rightarrow y$] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[$x \Rightarrow y$] expresses “[x] implies [y]”. [$x \Rightarrow y$] coincides with [$x \leq y$] for truth values [x] and [y].

Type table

\Rightarrow	B	D	X	E	\Rightarrow	N	Z	Z⁺	Z⁻
B	B	X	X	X	N	X	X	X	X
D	X	X	X	X	Z	X	X	X	X
X	X	X	X	X	Z⁺	X	X	X	X
E	X	X	X	X	Z⁻	X	X	X	X
\Rightarrow	D_m^∞	D_m	$D_m^{-\infty}$	D_∞^∞	$D_\infty^{-\infty}$	D_∞^∞	D_n^∞	D_n	$D_n^{-\infty}$
D_m^∞	X	X	X	X	X	X	X	X	X
D_m	X	X	X	X	X	X	X	X	X
$D_m^{-\infty}$	X	X	X	X	X	X	X	X	X
D_∞^∞	X	X	X	X	X	X	\Rightarrow	T	F
$D_\infty^{-\infty}$	X	X	X	X	X	X	T	T	F
$D_\infty^{-\infty}$	X	X	X	X	X	X	F	T	T

Mac rules

[**Mac rule StrictImpliesX** : $\perp \Rightarrow x \equiv \perp$]

[**Mac rule XImpliesStrict** : $x \Rightarrow \perp \equiv \perp$]

[**Mac rule SubXImpliesY** : $\langle A \Rightarrow B \mid x := S \rangle \equiv \langle A \mid x := S \rangle \Rightarrow \langle B \mid x := S \rangle$]

See also

[$x \rightarrow y$] Section A.57

[$x \Leftrightarrow y$] Section A.65

[$x \Leftarrow y$] Section A.66

[$x \leq y$] Section A.98

[$x \Rightarrow y$]	x implies y
[StrictImpliesX]	rule StrictImpliesX
[XImpliesStrict]	rule XImpliesStrict
[SubXImpliesY]	rule SubXImpliesY

A.68 x infers y [$x \vdash y$]**Parse tree****Sort**

[$x \vdash y$] is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

[$x \vdash y$] is used to form inference rules from equations.

Examples

[$x \equiv y \vdash y \equiv z \vdash x \equiv z$] states that if [x] equals [y] and [y] equals [z] then [x] equals [z].

See also

[$x \triangleright y$] Section A.56

[$x \rightarrow y$] Section A.57

A.69 x integer divide y [x // y]

Parse tree



Sort

[x // y] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[x // y] denotes [x] divided by [y], rounded to the nearest integer in the direction of $[-\infty]$. [x //₊ y], [x //₋ y], [x //₀ y], and [x //₂ y] are versions of [x // y] that use round to plus infinity, minus infinity, zero, and even, respectively. [x //₋ y] is identical to [x // y]. [x // y] disregards the precision of [x] and [y]. When [x] is a finite decimal fraction and [y] is a positive finite decimal fraction, [x // y] produces an integer result, i.e. an exact fraction that is a whole number. [x // y] equals [•] if [y] is zero or negative.

Type table

//	B	D	X	E	//	N	Z	Z ⁺	Z ⁻
B	X	X	X	X	N	-	-	N	X
D	X	-	X	X	Z	-	-	Z	X
X	X	X	X	X	Z ⁺	-	-	N	X
E	X	X	X	X	Z ⁻	-	-	Z ⁻	X
//	D _m [∞]	D _m	D _m ^{-∞}	D _∞ [∞]	D _∞ ^{-∞}	D _∞ [∞]	D _n [∞]	D _n ^{-∞}	D _n ^{-∞}
D _m [∞]	X	X	X	X	X	X	X	X	X
D _m	X	-	X	X	-	X	X	-	X
D _m ^{-∞}	X	X	X	X	X	X	X	X	X
D _∞ [∞]	X	X	X	X	X	X	//	T	F
D _∞ ^{-∞}	X	-	X	X	-	X	T	X	X
D _∞ [∞]	X	X	X	X	X	X	F	X	X

Mac rules

[Mac rule StrictIntegerDivideX : $\perp // y \equiv \perp$]

[Mac rule XIntegerDivideStrict : $x // \perp \equiv \perp$]

[Mac rule SubXIntegerDivideY : $\langle A // B \mid x := S \rangle \equiv \langle A \mid x := S \rangle // \langle B \mid x := S \rangle$]

[x // y]	x integer divide y
[StrictIntegerDivideX]	rule StrictIntegerDivideX
[XIntegerDivideStrict]	rule XIntegerDivideStrict
[SubXIntegerDivideY]	rule SubXIntegerDivideY

[**Mac rule IntegerDivideZero** : $x \in \mathbf{D} \vdash y \in \mathbf{D} \vdash y \leq 0 \vdash x // y \in \mathbf{X}$]
 [**Mac rule TypeDmIntegerDivideDm** : $m \in \mathbf{Z}^+ \vdash x \in \mathbf{D}_m \vdash y \in \mathbf{D}_m \vdash y > 0 \vdash x // y \in \mathbf{D}_\infty$]
 [**Mac rule TypeDmIntegerDivideDi** : $m \in \mathbf{Z}^+ \vdash x \in \mathbf{D}_m \vdash y \in \mathbf{D}_\infty \vdash y > 0 \vdash x // y \in \mathbf{D}_\infty$]
 [**Mac rule TypeDiIntegerDivideDm** : $m \in \mathbf{Z}^+ \vdash x \in \mathbf{D}_\infty \vdash y \in \mathbf{D}_m \vdash y > 0 \vdash x // y \in \mathbf{D}_\infty$]
 [**Mac rule TypeDiIntegerDivideDi** : $x \in \mathbf{D}_\infty \vdash y \in \mathbf{D}_\infty \vdash y > 0 \vdash x // y \in \mathbf{D}_\infty$]
 [**Mac rule TypeDmIntegerDivideDn** : $m \in \mathbf{Z}^+ \vdash n \in \mathbf{Z}^+ \vdash x \in \mathbf{D}_m \vdash y \in \mathbf{D}_n \vdash y > 0 \vdash x // y \in \mathbf{D}_\infty$]

See also

[$x \% y$] Section A.74
 [x/y] Section A.59

[IntegerDivideZero]	rule IntegerDivideZero
[TypeDmIntegerDivideDm]	rule TypeDmIntegerDivideDm
[TypeDmIntegerDivideDi]	rule TypeDmIntegerDivideDi
[TypeDiIntegerDivideDm]	rule TypeDiIntegerDivideDm
[TypeDiIntegerDivideDi]	rule TypeDiIntegerDivideDi
[TypeDmIntegerDivideDn]	rule TypeDmIntegerDivideDn

A.70 x lemma y colon z

Syntax

`[x lemma y:z]`.

Parse tree



Sort

`[x lemma y:z]` is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

Lemmas have the form `[x lemma y:z]` where `[y]` is the name of the lemma to be proved and `[z]` is the statement that the lemma claims to hold. `[x]` is the system in which the proof has to be stated. `[x lemma y:z]` defines `[y]` as shorthand for `[z]`. For each lemma there must be a proof of the lemma.

Examples

The following statement introduces L11.4.1 as shorthand for `[(1 :: 2) head ≡ (2 :: 1) tail]`.

`[Mac lemma L11.4.1 : (1 :: 2) head ≡ (2 :: 1) tail]`

See also

`[x proof of y:z]` Section A.83
`[x rule y:z]` Section A.86

A.71 x listset [x*]

Parse tree



Sort

[x*] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[x*] is the set of lists whose components belongs to [x].

Mac rules

[**Mac rule SubXListSet** : $\langle \mathcal{A}^* \mid x:=S \rangle \equiv \langle \mathcal{A} \mid x:=S \rangle^*$]
 [**Mac rule SetXListSet** : $A \in \mathbf{Set} \vdash A^* \in \mathbf{Set}$]
 [**Mac rule ElimListSet** : $A \in \mathbf{Set} \vdash x \in A^* \vdash x \in \mathbf{E} \vee x \in A \times A^*$]
 [**Mac rule IntroListSet** : $A \in \mathbf{Set} \vdash x \in \mathbf{E} \vee x \in A \times A^* \vdash x \in A^*$]
 [**Mac rule IntroListSetEmpty** : $A \in \mathbf{Set} \vdash \langle \rangle \in A^*$]
 [**Mac rule IntroListSetPair** : $A \in \mathbf{Set} \vdash x \in A \vdash y \in A^* \vdash x :: y \in A^*$]

Examples

[[$\langle 1, 2, 3 \rangle \in \mathbf{N}^*$ $\equiv \top$]]
 [[$\langle 0, 1, 2 \rangle \in (\mathbf{Z}^+)^*$ $\equiv \mathbf{F}$]]
 [[$\infty\text{-list}(1) \in \mathbf{N}^*$ $\equiv \perp$]]

See also

[$\langle \rangle$] Section A.10
 [**E**] Section A.40
 [$\langle x \rangle$] Section A.44
 [$x \in y$] Section A.50
 [$x \times y$] Section A.51
 [$x :: y$] Section A.79

[x*]	x listset
[SubXListSet]	rule SubXListSet
[SetXListSet]	rule SetXListSet
[ElimListSet]	rule ElimListSet
[IntroListSet]	rule IntroListSet
[IntroListSetEmpty]	rule IntroListSetEmpty
[IntroListSetPair]	rule IntroListSetPair

A.72 x macro equal y $[x \doteq y]$ **Parse tree****Sort**

$[x \doteq y]$ is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

$[\mathcal{A} \doteq \mathcal{B}]$ introduces $[\mathcal{A}]$ as a macro construct which is shorthand for $[\mathcal{B}]$. Contrary to constructs introduced by $[\mathcal{A} \doteq \mathcal{B}]$, constructs introduced by $[\mathcal{A} \doteq \mathcal{B}]$ cannot occur in syntax trees. Constructs introduced by $[\mathcal{A} \doteq \mathcal{B}]$ can occur in parse trees but disappear when constructing the syntax tree. Constructs introduced by $[\mathcal{A} \doteq \mathcal{B}]$ may expand to terms or statement constructs.

Examples

$$[\langle x \rangle \doteq x :: \langle \rangle]$$

$$[(x) \doteq x]$$
See also

$[x \doteq y]$ Section A.58

A.73 x minus y [x - y]

Parse tree



Sort

[x - y] is an operator (i.e. it may occur in the syntax tree of a term).

Description

If [x] and [y] are exact fractions then [x - y] denotes the exact difference of [x] and [y]. If [x] and [y] are floating fractions of equal precision, then [x - y] denotes the exact difference of [x] and [y] rounded to the common precision of [x] and [y] using round to even. If [x] and [y] are floating fractions of different precisions, then [x - y] equals [•]. If [x] is an exact fraction and [y] is a floating fraction or vice versa, then [x - y] denotes the exact difference of [x] and [y] rounded to the precision of the floating fraction using round to even. [x -₊ y], [x -₋ y], [x -₀ y], and [x -₂ y] are versions of [x - y] that use round to plus infinity, minus infinity, zero, and even, respectively. [x -₂ y] is identical to [x - y].

Type table

-	B	D	X	E	-	N	Z	Z ⁺	Z ⁻
B	X	X	X	X	N	Z	Z	Z	Z ⁺
D	X	-	X	X	Z	Z	Z	Z	Z
X	X	X	X	X	Z ⁺	Z	Z	Z	Z ⁺
E	X	X	X	X	Z ⁻	Z ⁻	Z	Z ⁻	Z
-	D _m [∞]	D _m	D _m ^{-∞}	D _∞ [∞]	D _∞ ^{-∞}	D _∞ ^{-∞}	D _n [∞]	D _n	D _n ^{-∞}
D _m [∞]	X	X	X	X	X	X	X	X	X
D _m	X	D _m	X	X	D _m	X	X	X	X
D _m ^{-∞}	X	X	X	X	X	X	X	X	X
D _∞ [∞]	X	X	X	X	X	X	-	T	F
D _∞ ^{-∞}	X	D _m	X	X	D _∞ ^{-∞}	X	T	X	X
D _∞ ^{-∞}	X	X	X	X	X	X	F	X	X

Mac rules

[Mac rule StrictMinusX : ⊥ - x ≡ ⊥]

[x - y] x minus y
 [StrictMinusX] rule StrictMinusX

[**Mac rule XMinusStrict** : $x - \perp \equiv \perp$]

[**Mac rule SubXMinusY** : $\langle A - B \mid x := S \rangle \equiv \langle A \mid x := S \rangle - \langle B \mid x := S \rangle$]

See also

[$x + y$] Section A.80

[$x @ y$] Section A.85

[$x \cdot y$] Section A.97

[XMinusStrict] rule XMinusStrict
[SubXMinusY] rule SubXMinusY

A.74 x modulo y [x % y]

Parse tree



Sort

[x % y] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[x % y] denotes the remainder when integer dividing [x] by [y]. [x % y] disregards the precision of [x] and [y]. When [x] is a finite decimal fraction and [y] is a positive finite decimal fraction, [x % y] produces an exact fraction. [x % y] equals [•] if [y] is zero or negative.

[x %₊ y], [x %₋ y], [x %₀ y], and [x %₂ y] are versions of [x % y] that express the remainders of [x //₊ y], [x //₋ y], [x //₀ y], and [x //₂ y], respectively. [x %₋ y] is identical to [x % y].

For exact decimal fractions [x] and positive exact decimal fractions [y] you have:

$$\begin{aligned}
 [x \% y] &\equiv x - (x // y) \cdot y \\
 [x \%_+ y] &\equiv x - (x //_+ y) \cdot y \\
 [x \%_- y] &\equiv x - (x //_- y) \cdot y \\
 [x \%_0 y] &\equiv x - (x //_0 y) \cdot y \\
 [x \%_2 y] &\equiv x - (x //_2 y) \cdot y
 \end{aligned}$$

Type table

%	B	D	X	E	%	N	Z	Z ⁺	Z ⁻			
B	X	X	X	X	N	-	-	N	X			
D	X	-	X	X	Z	-	-	N	X			
X	X	X	X	X	Z ⁺	-	-	N	X			
E	X	X	X	X	Z ⁻	-	-	N	X			
%	D _m [∞]	D _m	D _m ^{-∞}	D _∞ [∞]	D _∞ ^{-∞}	D _∞ [∞]	D _n [∞]	D _n ^{-∞}	D _n ^{-∞}			
D _m [∞]	X	X	X	X	X	X	X	X	X			
D _m	X	-	X	X	-	X	X	-	X			
D _m ^{-∞}	X	X	X	X	X	X	X	X	X			
D _∞ [∞]	X	X	X	X	X	X	<table border="1"><tr><td>%</td><td>T</td><td>F</td></tr></table>	%	T	F		
%	T	F										
D _∞ ^{-∞}	X	-	X	X	-	X	<table border="1"><tr><td>T</td><td>X</td><td>X</td></tr></table>	T	X	X		
T	X	X										
D _∞ ^{-∞}	X	X	X	X	X	X	<table border="1"><tr><td>F</td><td>X</td><td>X</td></tr></table>	F	X	X		
F	X	X										

[x % y] x modulo y

Mac rules

[**Mac rule StrictModuloX** : $\perp \% x \equiv \perp$]
 [**Mac rule XModuloStrict** : $x \% \perp \equiv \perp$]
 [**Mac rule SubXModuloY** : $\langle \mathcal{A} \% \mathcal{B} \mid x := \mathcal{S} \rangle \equiv \langle \mathcal{A} \mid x := \mathcal{S} \rangle \% \langle \mathcal{B} \mid x := \mathcal{S} \rangle$]
 [**Mac rule ModuloZero** : $x \in \mathbf{D} \vdash y \in \mathbf{D} \vdash y \leq 0 \vdash x \% y \in \mathbf{X}$]
 [**Mac rule TypeDmModuloDm** : $m \in \mathbf{Z}^+ \vdash x \in \mathbf{D}_m \vdash y \in \mathbf{D}_m \vdash y > 0 \vdash x \% y \in \mathbf{D}_\infty$]
 [**Mac rule TypeDmModuloDi** : $m \in \mathbf{Z}^+ \vdash x \in \mathbf{D}_m \vdash y \in \mathbf{D}_\infty \vdash y > 0 \vdash x \% y \in \mathbf{D}_\infty$]
 [**Mac rule TypeDiModuloDm** : $m \in \mathbf{Z}^+ \vdash x \in \mathbf{D}_\infty \vdash y \in \mathbf{D}_m \vdash y > 0 \vdash x \% y \in \mathbf{D}_\infty$]
 [**Mac rule TypeDiModuloDi** : $x \in \mathbf{D}_\infty \vdash y \in \mathbf{D}_\infty \vdash y > 0 \vdash x \% y \in \mathbf{D}_\infty$]
 [**Mac rule TypeDmModuloDn** : $m \in \mathbf{Z}^+ \vdash n \in \mathbf{Z}^+ \vdash x \in \mathbf{D}_m \vdash y \in \mathbf{D}_n \vdash y > 0 \vdash x \% y \in \mathbf{D}_\infty$]

See also

[$x // y$] Section A.69

[StrictModuloX]	rule StrictModuloX
[XModuloStrict]	rule XModuloStrict
[SubXModuloY]	rule SubXModuloY
[ModuloZero]	rule ModuloZero
[TypeDmModuloDm]	rule TypeDmModuloDm
[TypeDmModuloDi]	rule TypeDmModuloDi
[TypeDiModuloDm]	rule TypeDiModuloDm
[TypeDiModuloDi]	rule TypeDiModuloDi
[TypeDmModuloDn]	rule TypeDmModuloDn

A.75 x modus ponens y [x \triangleright y]**Parse tree****Sort**

[x \triangleright y] is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

[x \triangleright y] is used to form argumentations from equations using the inference rules of Modus Ponens.

Examples

$$[x \in \mathbf{Z} \rightarrow x + 2 \in \mathbf{Z} \triangleright x \in \mathbf{Z} \triangleright x + 2 \in \mathbf{Z}]$$

states that [x + 2 \in \mathbf{Z}] follows from [x \in \mathbf{Z} \rightarrow x + 2 \in \mathbf{Z}] and [x \in \mathbf{Z}] according to the inference rule of ModusPonens. The statement is shorthand for

$$[\text{ModusPonens } \triangleright x \in \mathbf{Z} \triangleright x \in \mathbf{Z} \rightarrow x + 2 \in \mathbf{Z} \triangleright x + 2 \in \mathbf{Z}].$$

See also

[x \triangleright y] Section A.56

[x \rightarrow y] Section A.57

A.76 x optimised equal y $[x \doteq y]$ **Parse tree****Sort**

$[x \doteq y]$ is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

$[A \doteq B]$ means the same as $[A \doteq B]$ but may affect the time Map spends on computing $[A]$.

Examples

$[n! \doteq \text{if}(n = 0, 1, n \cdot (n - 1)!)]$.

See also

$[x \doteq y]$ Section A.58

A.77 x or y [x ∨ y]

Parse tree



Sort

[x ∨ y] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[x ∨ y] denotes the maximum of [x] and [y]. In particular, for truth values [x] and [y], [x ∨ y] denotes “[x] or [y]”. For truth values [x] and [y], [x ∨ y] coincides with [x + y]. [x ∨ y ≡ •] if [x] and [y] are decimal fractions of different precision.

Type table

∨	B	D	X	E	∨	N	Z	Z ⁺	Z ⁻
B	B	X	X	X	N	N	N	Z ⁺	N
D	X	-	X	X	Z	N	Z	Z ⁺	Z
X	X	X	X	X	Z ⁺	Z ⁺	Z ⁺	Z ⁺	Z ⁺
E	X	X	X	X	Z ⁻	N	Z	Z ⁺	Z ⁻
∨	D _m [∞]	D _m	D _m ^{-∞}	D _∞ [∞]	D _∞	D _∞ ^{-∞}	D _n [∞]	D _n	D _n ^{-∞}
D _m [∞]	D _m [∞]	D _m [∞]	D _m [∞]	X	X	X	X	X	X
D _m	D _m [∞]	D _m	D _m	X	X	X	X	X	X
D _m ^{-∞}	D _m [∞]	D _m	D _m ^{-∞}	X	X	X	X	X	X
D _∞ [∞]	X	X	X	D _∞ [∞]	D _∞ [∞]	D _∞ [∞]	∨	T	F
D _∞	X	X	X	D _∞ [∞]	D _∞ [∞]	D _∞ [∞]	T	T	T
D _∞ ^{-∞}	X	X	X	D _∞ [∞]	D _∞ [∞]	D _∞ [∞]	F	T	F

Mac rules

[Mac rule StrictOrX : ⊥ ∨ x ≡ ⊥]

[Mac rule XOrStrict : x ∨ ⊥ ≡ ⊥]

[Mac rule SubXOrY : ⟨A ∨ B | x:=S⟩ ≡ ⟨A | x:=S⟩ ∨ ⟨B | x:=S⟩]

[x ∨ y]	x or y
[StrictOrX]	rule StrictOrX
[XOrStrict]	rule XOrStrict
[SubXOrY]	rule SubXOrY

See also

[$x \wedge y$] Section A.46
[$x + y$] Section A.80

A.78 `x pair` [`x pair`]**Parse tree****Sort**

[`x pair`] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[`x pair`] is true if [`x`] is a pair. Truth values, exceptions, decimal fractions, and the empty list are atoms rather than pairs. [`⊥`] is neither an atom nor a pair.

Type table

<code>x</code>	B	D	X	E	N	Z	Z⁺	Z⁻
<code>x pair</code>	F	F	F	F	F	F	F	F
<code>x</code>	D_m[∞]	D_m	D_m^{-∞}	D_∞[∞]	D_∞	D_∞^{-∞}	T	F
<code>x pair</code>	F	F	F	F	F	F	F	F

Mac rules

[**Mac rule PairIsPair** : (`x :: y`) pair \equiv T]
 [**Mac rule SubXPair** : (`A pair | x:=S`) \equiv (`A | x:=S`) pair]
 [**Mac rule StrictPair** : `⊥ pair` \equiv `⊥`]

Examples

[`2.0F pair` \equiv `F`]
 [`<x,y> pair` \equiv `T`]

See also

[`x atom`] Section A.49
 [`x :: y`] Section A.79

[<code>x pair</code>]	<code>x pair</code>
[<code>PairIsPair</code>]	rule <code>PairIsPair</code>
[<code>SubXPair</code>]	rule <code>SubXPair</code>
[<code>StrictPair</code>]	rule <code>StrictPair</code>

A.79 x pair y $[x :: y]$ **Parse tree****Sort**

$[x :: y]$ is an operator (i.e. it may occur in the syntax tree of a term).

Description

$[x :: y]$ denotes the pair whose first element is $[x]$ and whose second element is $[y]$. $[x :: y]$ aggregates two mathematical values $[x]$ and $[y]$ into a single mathematical value such that they can be taken apart using $[z \text{ head}]$ and $[z \text{ tail}]$ and in such a way that $[z \text{ pair}]$ can recognise the aggregate as a pair. $[x :: y]$ contains the information contained in $[x]$ plus the information contained in $[y]$ plus the information that it is a pair. $[x :: y]$ is non-strict in both $[x]$ and $[y]$. $[x :: y]$ is known as “lazy cons” in the literature.

$[x :: y \equiv u :: v]$ holds if $[x \equiv u]$ and $[y \equiv v]$ both hold. $[x :: y \equiv u :: v]$ fails if $[x \equiv u]$ fails or $[y \equiv v]$ fails or both fail. $[x :: y \equiv z]$ fails for all $[x]$ and $[y]$ if $[z]$ is a truth value, a decimal fraction, an exception, the empty list, or $[\perp]$. In particular, $[\perp :: \perp \equiv \perp]$ fails.

Mac rules

[**Mac rule SubXPairY** : $\langle A :: B \mid x := S \rangle \equiv \langle A \mid x := S \rangle :: \langle B \mid x := S \rangle$]

Examples

$[\langle 1, 2, 3 \rangle \equiv 1 :: 2 :: 3 :: \langle \rangle]$

See also

$[x \times y]$ Section A.51
 $[x \text{ head}]$ Section A.64
 $[x \text{ tail}]$ Section A.95
 $[x \text{ pair}]$ Section A.78
 $[x \text{ atom}]$ Section A.49

$[x :: y]$ x pair y
 [SubXPairY] rule SubXPairY

A.80 x plus y [x + y]

Parse tree



Sort

[x + y] is an operator (i.e. it may occur in the syntax tree of a term).

Description

If [x] and [y] are truth values then [x + y] coincides with [x ∨ y]. If [x] and [y] are exact fractions then [x + y] denotes the exact sum of [x] and [y]. If [x] and [y] are floating fractions of equal precision, then [x + y] denotes the exact sum of [x] and [y] rounded to the common precision of [x] and [y] using round to even. If [x] and [y] are floating fractions of different precisions, then [x + y] equals [•]. If [x] is an exact fraction and [y] is a floating fraction or vice versa, then [x + y] denotes the exact sum of [x] and [y] rounded to the precision of the floating fraction using round to even. [x++y], [x+-y], [x+0y], and [x+2y] are versions of [x + y] that use round to plus infinity, minus infinity, zero, and even, respectively. [x+2y] is identical to [x + y].

Type table

+	B	D	X	E	+	N	Z	Z ⁺	Z ⁻
B	B	X	X	X	N	N	Z	Z ⁺	Z
D	X	-	X	X	Z	Z	Z	Z	Z
X	X	X	X	X	Z ⁺	Z ⁺	Z	Z ⁺	Z
E	X	X	X	X	Z ⁻	Z	Z	Z	Z ⁻
+	D _m [∞]	D _m	D _m ^{-∞}	D _∞ [∞]	D _∞	D _∞ ^{-∞}	D _n [∞]	D _n	D _n ^{-∞}
D _m [∞]	X	X	X	X	X	X	X	X	X
D _m	X	D _m	X	X	D _m	X	X	X	X
D _m ^{-∞}	X	X	X	X	X	X	X	X	X
D _∞ [∞]	X	X	X	X	X	X	+	T	F
D _∞	X	D _m	X	X	D _∞	X	T	T	T
D _∞ ^{-∞}	X	X	X	X	X	X	F	T	F

Mac rules

[Mac rule StrictPlusX : ⊥ + x ≡ ⊥]

[x + y] x plus y
 [StrictPlusX] rule StrictPlusX

[**Mac rule XPlusStrict** : $x + \perp \equiv \perp$]

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

See also

[$x - y$] Section A.73

[$x \vee y$] Section A.77

[$x @ y$] Section A.85

[$x \cdot y$] Section A.97

[XPlusStrict] rule XPlusStrict
[SubXPlusY] rule SubXPlusY

A.81 x power y end [x^y]

Parse tree



Sort

[x^y] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[x^y] denotes [x] raised to the power of [y]. For decimal fractions [x] and [y], [x^y] differs from [•] in the following cases:

- (a) [x] and [y] are floating fractions with equal precision and [x] is positive. In this case [x^y] has the same precision as [x] and [y].
- (b) [x] is a floating fraction and [y] is an exact fraction or vice versa, and [x] is positive. In this case [x^y] has the same precision as the floating fraction.
- (c) [x] is a decimal fraction and [y] is a natural number. In this case [x^y] has the same precision as [x]. In particular, [x⁰ ≡ 1@#x].
- (d) [x = 10] and [y] is an integer. In this case [x^y] has precision [∞].

Type table

u^v	B	D	X	E	u^v	N	Z	Z ⁺	Z ⁻
B	X	X	X	X	N	N	-	N	-
D	X	-	X	X	Z	Z	-	Z	-
X	X	X	X	X	Z ⁺	Z ⁺	-	Z ⁺	-
E	X	X	X	X	Z ⁻	Z	-	Z	-
u^v	D _m [∞]	D _m	D _m ^{-∞}	D _∞ [∞]	D _∞	D _∞ ^{-∞}	D _n [∞]	D _n	D _n ^{-∞}
D _m [∞]	X	X	X	X	X	X	X	X	X
D _m	X	-	X	X	-	X	X	-	X
D _m ^{-∞}	X	X	X	X	X	X	X	X	X
D _∞ [∞]	X	X	X	X	X	X	u^v	T	F
D _∞	X	-	X	X	-	X	T	X	X
D _∞ ^{-∞}	X	X	X	X	X	X	F	X	X

[x^y] x power y end

Mac rules

[**Mac rule StrictPowerXEnd** : $\perp^X \equiv \perp$]
[**Mac rule XPowerStrictEnd** : $x^\perp \equiv \perp$]
[**Mac rule SubXPowerYEnd** : $\langle \mathcal{A}^B \mid x:=S \rangle \equiv \langle \mathcal{A} \mid x:=S \rangle^{\langle B \mid x:=S \rangle}$]

[StrictPowerXEnd] rule StrictPowerXEnd
[XPowerStrictEnd] rule XPowerStrictEnd
[SubXPowerYEnd] rule SubXPowerYEnd

A.82 x predecessor [x⁻]

Parse tree



Sort

[x⁻] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[x⁻] denotes [x - 1]. All integers can be expressed using [0], [x⁺] and [x⁻].

Definition

[x⁻ ≡ x - 1].

Type table

x	B	D	X	E	N	Z	Z ⁺	Z ⁻
x tail	X	-	X	X	Z	Z	N	Z ⁻
x	D _m [∞]	D _m	D _m ^{-∞}	D _∞ [∞]	D _∞	D _∞ ^{-∞}	T	F
x tail	X	D _m	X	X	D _∞	X	X	X

Mac rules

[Mac rule SubXPredecessor : ⟨A⁻ | x:=S⟩ ≡ ⟨A | x:=S⟩⁻]

Examples

[5⁻ ≡ 2]
 [0⁻ ≡ -5]

See also

[x⁺] Section A.94

[x⁻] x predecessor
 [SubXPredecessor] rule SubXPredecessor

A.83 x proof of y colon z

Syntax

$\boxed{[x \text{ proof of } y:z]}$.

Parse tree



Sort

$[x \text{ proof of } y:z]$ is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

Proofs have the form $[x \text{ proof of } y:z]$ where $[y]$ is the lemma to be proved, $[z]$ is a list of instantiation statements separated by semicolons, and $[x]$ is the axiomatic system in use.

Examples

$[Mac \text{ proof of } L11.4.1:$

L1 : HeadPair ▷	(1 :: 2) head ≡ 1	;
L2 : TailPair ▷	(2 :: 1) tail ≡ 1	;
L3 : Commutativity ▷ L2 ▷	1 ≡ (2 :: 1) tail	;
L4 : Transitivity ▷ L1 ▷ L3 ▷	(1 :: 2) head ≡ (2 :: 1) tail]

See also

$[x \text{ lemma } y:z]$	Section A.70
$[x \text{ rule } y:z]$	Section A.86
$[x:y]$	Section A.52
$[x ▷ y]$	Section A.56
$[x;y]$	Section A.88

$[x \text{ proof of } y:z]$ x proof of y colon z

A.84 x reduces to y [$x \overset{\pm}{\rightarrow} y$]**Parse tree****Sort**

[$x \overset{\pm}{\rightarrow} y$] is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

[$\mathcal{A} \overset{\pm}{\rightarrow} \mathcal{B}$] states that [x] reduces to [y] in finite, non-zero time. If [$\mathcal{A} \overset{\pm}{\rightarrow} \mathcal{B}$] then [$\mathcal{A} \equiv \mathcal{B}$].

Examples

[$2 \cdot 3 + 4 \overset{\pm}{\rightarrow} 6 + 4 \overset{\pm}{\rightarrow} 10$]

See also

[$x \equiv y$] Section A.60

[$x \overset{\circ}{\rightarrow} y$] Section A.96

[$x \overset{\pm}{\rightarrow} y$] x reduces to y

A.85 x round y [$x@y$]

Parse tree



Sort

[$x@y$] is an operator (i.e. it may occur in the syntax tree of a term).

Description

If [x] is a decimal fraction and [y] is a positive integer or [$+\infty$], then [$x@y$] rounds [x] to precision [y]. If [y] is weakly greater than the precision of [x], then [$x@y$] extends the precision of [x] without affecting the value of [x]. If [y] is strongly less than the precision of [x], then [$x@y$] rounds [x] to the nearest value of precision [y]. If two values of precision [y] are equally distant from [x], then [$x@y$] chooses the one of the two values whose least significant digit is even. [$x@_+y$], [$x@_-y$], [$x@_0y$], and [$x@_2y$] are versions of [$x@y$] that round towards plus infinity, minus infinity, zero, and even, respectively. [$x@_2y$] is identical to [$x@y$].

Type table

@	B	D	X	E	@	N	Z	Z ⁺	Z ⁻
B	X	X	X	X	N	-	-	D	-
D	X	-	X	X	Z	-	-	D	-
X	X	X	X	X	Z ⁺	-	-	D	-
E	X	X	X	X	Z ⁻	-	-	D	-
@	D _m [∞]	D _m	D _m ^{-∞}	D _∞ [∞]	D _∞	D _∞ ^{-∞}	D _n [∞]	D _n	D _n ^{-∞}
D _m [∞]	X	X	X	D _∞ [∞]	-	X	X	X	X
D _m	X	X	X	D _∞ [∞]	-	X	X	X	X
D _m ^{-∞}	X	X	X	D _∞ ^{-∞}	-	X	X	X	X
D _∞ [∞]	X	X	X	D _∞ [∞]	-	X	@	T	F
D _∞	X	X	X	D _∞ [∞]	-	X	T	X	X
D _∞ ^{-∞}	X	X	X	D _∞ ^{-∞}	-	X	F	X	X

Mac rules

[Mac rule **StrictRoundX** : $\perp @ x \equiv \perp$]

[Mac rule **XRoundStrict** : $x @ \perp \equiv \perp$]

[$x@y$]	x round y
[StrictRoundX]	rule StrictRoundX
[XRoundStrict]	rule XRoundStrict

[**Mac rule SubRound** : $\langle \mathcal{A} @ \mathcal{B} \mid x := S \rangle \equiv \langle \mathcal{A} \mid x := S \rangle @ \langle \mathcal{B} \mid x := S \rangle$]
 [**Mac rule RoundException** : $x \in \mathbf{D} \vdash y \in \mathbf{D} \vdash y \notin \mathbf{Z}^+ \vdash y \notin \mathbf{D}_\infty^\infty \vdash x @ y \in \mathbf{X}$]
 [**Mac rule RoundIiIn** : $x \in \mathbf{D}_\infty^\infty \vdash n \in \mathbf{Z}^+ \vdash x @ y \in \mathbf{D}_n^\infty$]
 [**Mac rule RoundDiDn** : $x \in \mathbf{D}_\infty^\infty \vdash n \in \mathbf{Z}^+ \vdash x @ y \in \mathbf{D}_n$]
 [**Mac rule RoundMiMn** : $x \in \mathbf{D}_\infty^\infty \vdash n \in \mathbf{Z}^+ \vdash x @ y \in \mathbf{D}_n^{-\infty}$]
 [**Mac rule RoundImIn** : $m \in \mathbf{Z}^+ \vdash x \in \mathbf{D}_m^\infty \vdash n \in \mathbf{Z}^+ \vdash x @ y \in \mathbf{D}_n^\infty$]
 [**Mac rule RoundDmDn** : $m \in \mathbf{Z}^+ \vdash x \in \mathbf{D}_m \vdash n \in \mathbf{Z}^+ \vdash x @ y \in \mathbf{D}_n$]
 [**Mac rule RoundMmMn** : $m \in \mathbf{Z}^+ \vdash x \in \mathbf{D}_m^{-\infty} \vdash n \in \mathbf{Z}^+ \vdash x @ y \in \mathbf{D}_n^{-\infty}$]

Examples

x	x@2	x@+2	x@-2	x@02
0F	0.0F	0.0F	0.0F	0.0F
0.0F	0.0F	0.0F	0.0F	0.0F
0.00F	0.0F	0.0F	0.0F	0.0F
0	0.0F	0.0F	0.0F	0.0F
1.14F	1.1F	1.2F	1.1F	1.1F
1.16F	1.2F	1.2F	1.1F	1.1F
1.15F	1.2F	1.2F	1.1F	1.1F
1.25F	1.2F	1.3F	1.2F	1.2F
1.35F	1.4F	1.4F	1.3F	1.3F
-1.14F	-1.1F	-1.1F	-1.2F	-1.1F
-1.16F	-1.2F	-1.1F	-1.2F	-1.1F
-1.15F	-1.2F	-1.1F	-1.2F	-1.1F
-1.25F	-1.2F	-1.2F	-1.3F	-1.2F
-1.35F	-1.4F	-1.3F	-1.4F	-1.3F

See also

[#x] Section A.27
 [x + y] Section A.80

[SubRound]	rule SubRound
[RoundException]	rule RoundException
[RoundIiIn]	rule RoundIiIn
[RoundDiDn]	rule RoundDiDn
[RoundMiMn]	rule RoundMiMn
[RoundImIn]	rule RoundImIn
[RoundDmDn]	rule RoundDmDn
[RoundMmMn]	rule RoundMmMn

A.86 x rule y colon z

Syntax

$[\boxed{x \text{ rule } y:z}]$.

Parse tree



Sort

$[\text{Mac rule } x:y]$ is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

Rules have the form $[x \text{ rule } y:z]$ where $[y]$ is the name of the rule, $[z]$ is the rule itself, i.e. a statement that claimed to hold without proof, and $[x]$ is the axiomatic system in which the rule holds without proof. $[x \text{ rule } y:z]$ defines $[y]$ as shorthand for $[z]$. Rules may occur in argumentations in proofs.

Examples

The following statement introduces Transitivity as shorthand for $[x \equiv y \vdash y \equiv z \vdash x \equiv z]$ and states that Transitivity holds without proof in the $[\text{Mac}]$ system.

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

See also

$[x \text{ lemma } y:z]$ Section A.70
 $[x \text{ proof of } y:z]$ Section A.83

A.87 x same priority y $[x \overset{\cup}{=} y]$

Parse tree

**Sort**

$[x \overset{\cup}{=} y]$ is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

$[A \overset{\cup}{=} B]$ states that the constructs $[A]$ and $[B]$ have the same priority and the same associativity when forming parse trees. Furthermore, $[A]$ and $[B]$ are subject to the same term reduction rules when reducing parse trees to syntax trees.

Examples

$[x - y \overset{\cup}{=} x + y]$ and $[x + y + z \overset{\cup}{\rightarrow} (x + y) + z]$ together make `Map` interpret $[2 + 3 - 4]$ as $[(2 + 3) - 4]$ and $[2 - 3 + 4]$ as $[(2 - 3) + 4]$.

See also

- $[x \overset{\rightarrow}{=} y]$ Section A.48
- $[x \overset{\succ}{=} y]$ Section A.63
- $[x \overset{\circ}{=} y]$ Section A.96

A.88 x semicolon y [x; y]**Parse tree****Sort**

[x; y] is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

[x; y] is used to form lists of statements. It is used between lines of derivation proofs, between equations in counterexamples, and between rules and lemmas in statements that proclaim more than one rule or lemma.

See also

[x proof of y:z] Section A.83

A.89 x simple head [x Head]

Parse tree



Sort

[x Head] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[x Head] denotes the first element of a simple pair [x].

Mac rules

[Mac rule SimpleHeadNil : N Head \equiv N]
 [Mac rule SimpleHeadPair : (x . y) Head \equiv x]
 [Mac rule SimpleHeadBottom : \perp Head \equiv \perp]
 [Mac rule SubXSimpleHead : $\langle \mathcal{A} \text{ Head} \mid x := \mathcal{S} \rangle \equiv \langle \mathcal{A} \mid x := \mathcal{S} \rangle \text{ Head}$]

Examples

[F Head \equiv T]

See also

[x . y] Section A.90
 [x Tail] Section A.91

[x Head]	x simple head
[SimpleHeadNil]	rule SimpleHeadNil
[SimpleHeadPair]	rule SimpleHeadPair
[SimpleHeadBottom]	rule SimpleHeadBottom
[SubXSimpleHead]	rule SubXSimpleHead

A.90 x simple pair y $[x \dot{.} y]$ **Parse tree****Sort**

$[x \dot{.} y]$ is an operator (i.e. it may occur in the syntax tree of a term).

Description

$[x \dot{.} y]$ denotes the simple pair whose first element is $[x]$ and whose second element is $[y]$.

Mac rules

[Mac rule SubXSimplePairY : $\langle A \dot{.} B \mid x := S \rangle \equiv \langle A \mid x := S \rangle \dot{.} \langle B \mid x := S \rangle$]

Examples

[F \equiv N $\dot{.}$ N]

See also

[x Head] Section A.89

[x Tail] Section A.91

[$x \dot{.} y$] x simple pair y
 [SubXSimplePairY] rule SubXSimplePairY

A.91 x simple tail [x Tail]

Parse tree



Sort

[x Tail] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[x Tail] denotes the first element of a simple pair [x].

Mac rules

[Mac rule SimpleTailNil : N Tail \equiv N]
 [Mac rule SimpleTailPair : (x .: y) Tail \equiv y]
 [Mac rule SimpleTailBottom : \perp Tail \equiv \perp]
 [Mac rule SubXSimpleTail : $\langle \mathcal{A} \text{ Tail} \mid x := \mathcal{S} \rangle \equiv \langle \mathcal{A} \mid x := \mathcal{S} \rangle \text{ Tail}$]

Examples

[F Tail \equiv T]

See also

[x Head] Section A.89
 [x .: y] Section A.90

[x Tail]	x simple tail
[SimpleTailNil]	rule SimpleTailNil
[SimpleTailPair]	rule SimpleTailPair
[SimpleTailBottom]	rule SimpleTailBottom
[SubXSimpleTail]	rule SubXSimpleTail

A.92 x strongly greater than y [$x > y$]

Parse tree



Sort

[$x > y$] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[$x > y$] is true if [x] is strongly greater than [y] (i.e. greater than and, in particular, not equal to). [$x > y$] uses the convention that truth is greater than falsehood when comparing truth values. When comparing decimal fractions, [$x > y$] disregards precision.

Type table

$>$	B	D	X	E	$>$	N	Z	Z⁺	Z⁻
B	B	X	X	X	N	B	B	B	T
D	X	B	X	X	Z	B	B	B	B
X	X	X	X	X	Z⁺	B	B	B	T
E	X	X	X	X	Z⁻	F	B	F	B
$>$	D_m^∞	D_m	$D_m^{-\infty}$	D_∞	D_∞	$D_\infty^{-\infty}$	D_n^∞	D_n	$D_n^{-\infty}$
D_m^∞	F	T	T	F	T	T	F	T	T
D_m	F	B	T	F	B	T	F	B	T
$D_m^{-\infty}$	F	F	F	F	F	F	F	F	F
D_∞	F	T	T	F	T	T	$>$	T	F
$D_\infty^{-\infty}$	F	B	T	F	B	T	T	F	T
D_n^∞	F	F	F	F	F	F	F	F	F

Mac rules

[**Mac rule StrictStronglyGreaterThanX** : $\perp > x \equiv \perp$]

[**Mac rule XStronglyGreaterThanStrict** : $x > \perp \equiv \perp$]

[**Mac rule SubXStronglyGreaterThanY** : $\langle A > B \mid x := S \rangle \equiv$

$\langle A \mid x := S \rangle > \langle B \mid x := S \rangle$]

[$x > y$]	x strongly greater than y
[StrictStronglyGreaterThanX]	rule StrictStronglyGreaterThanX
[XStronglyGreaterThanStrict]	rule XStronglyGreaterThanStrict
[SubXStronglyGreaterThanY]	rule SubXStronglyGreaterThanY

See also

- [$x = y$] Section A.54
- [$x < y$] Section A.93
- [$x \geq y$] Section A.98
- [$x \leq y$] Section A.100

A.93 x strongly less than y [$x < y$]

Parse tree



Sort

[$x < y$] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[$x < y$] is true if [x] is strongly less than [y] (i.e. less than and, in particular, not equal to). [$x < y$] uses the convention that truth is greater than falsehood when comparing truth values. When comparing decimal fractions, [$x < y$] disregards precision.

Type table

<	B	D	X	E	<	N	Z	Z ⁺	Z ⁻
B	B	X	X	X	N	B	B	B	F
D	X	B	X	X	Z	B	B	B	B
X	X	X	X	X	Z ⁺	B	B	B	F
E	X	X	X	X	Z ⁻	T	B	T	B
<	D _m [∞]	D _m	D _m ^{-∞}	D _∞ [∞]	D _∞	D _∞ ^{-∞}	D _n [∞]	D _n	D _n ^{-∞}
D _m [∞]	F	F	F	F	F	F	F	F	F
D _m	T	B	F	T	B	F	T	B	F
D _m ^{-∞}	T	T	F	T	T	F	T	T	F
D _∞ [∞]	F	F	F	F	F	F	<	T	F
D _∞	T	B	F	T	B	F	T	F	F
D _∞ ^{-∞}	T	T	F	T	T	F	F	T	F

Mac rules

[Mac rule **StrictStronglyLessThanX** : $\perp < x \equiv \perp$]

[Mac rule **XStronglyLessThanStrict** : $x < \perp \equiv \perp$]

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

[$x < y$]	x strongly less than y
[StrictStronglyLessThanX]	rule StrictStronglyLessThanX
[XStronglyLessThanStrict]	rule XStronglyLessThanStrict
[SubXStronglyLessThanY]	rule SubXStronglyLessThanY

See also

- [$x = y$] Section A.54
- [$x > y$] Section A.92
- [$x \geq y$] Section A.98
- [$x \leq y$] Section A.100

A.94 x successor $[x^+]$ **Parse tree****Sort**

$[x^+]$ is an operator (i.e. it may occur in the syntax tree of a term).

Description

$[x^+]$ denotes $[x + 1]$. All natural numbers can be expressed using $[0]$ and $[x^+]$.

Definition

$[x^+ \doteq x + 1]$.

Type table

x	B	D	X	E	N	Z	Z⁺	Z⁻
x tail	X	-	X	X	N	Z	Z⁺	Z
x	D_m[∞]	D_m	D_m[∞]	D_∞[∞]	D_∞	D_∞[∞]	T	F
x tail	X	D_m	X	X	D_∞	X	X	X

Mac rules

[**Mac rule SubXSuccessor** : $\langle A^+ \mid x := S \rangle \equiv \langle A \mid x := S \rangle^+$]

Examples

$[0^{+++++} \equiv 5]$

See also

$[x^-]$ Section A.82

$[x^+]$ x successor
 [SubXSuccessor] rule SubXSuccessor

A.95 `x tail` [`x tail`]

Parse tree



Sort

[`x tail`] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[`x tail`] denotes the second element of a pair [`x`] or (which is the same) all of a list except the first element. If [`x`] is a truth value, a decimal fraction, an exception or the empty list, then [`x tail`] equals [`x`].

Type table

<code>x</code>	B	D	X	E	N	Z	Z⁺	Z⁻
<code>x tail</code>	B	D	X	E	N	Z	Z⁺	Z⁻
<code>x</code>	D_m[∞]	D_m	D_m^{-∞}	D_∞[∞]	D_∞	D_∞^{-∞}	T	F
<code>x tail</code>	D_m[∞]	D_m	D_m^{-∞}	D_∞[∞]	D_∞	D_∞^{-∞}	T	F

Mac rules

- [**Mac rule TailEmpty** : $\langle \rangle \text{ tail} \equiv \langle \rangle$]
- [**Mac rule TailPair** : $\langle x :: y \rangle \text{ tail} \equiv y$]
- [**Mac rule TailBottom** : $\perp \text{ tail} \equiv \perp$]
- [**Mac rule SubXTail** : $\langle \mathcal{A} \text{ tail} \mid x := \mathcal{S} \rangle \equiv \langle \mathcal{A} \mid x := \mathcal{S} \rangle \text{ tail}$]

Examples

[$\langle 1, 2, 3, 4, 5 \rangle \text{ tail} \equiv \langle 2, 3, 4, 5 \rangle$]

See also

- [`x head`] Section A.64
- [`x :: y`] Section A.79

[<code>x tail</code>]	<code>x tail</code>
[TailEmpty]	rule TailEmpty
[TailPair]	rule TailPair
[TailBottom]	rule TailBottom
[SubXTail]	rule SubXTail

A.96 x term reduces to y [$x \overset{\circ}{\rightarrow} y$]

Parse tree



Sort

[$x \overset{\circ}{\rightarrow} y$] is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

[$\mathcal{A} \overset{\circ}{\rightarrow} \mathcal{B}$] states that [\mathcal{A}] reduces to [\mathcal{B}]. [$\mathcal{A} \doteq \mathcal{B}$] also states that [\mathcal{A}] reduces to [\mathcal{B}]. The former rule is a *term reduction rule* whereas the latter is a *graph reduction rule*. The term *reduction rule* stands for “graph reduction rule”.

The difference between term and graph reduction rules is that term reduction rules are applied before graph reduction rules when computing the value of an expression.

To compute the value of an expression, do as follows: Convert the expression into a parse tree using priority and associativity rules. Then convert the parse tree into a syntax tree using term reduction rules and *macro reduction rules*. Finally convert the syntax tree into a value using graph reduction rules.

To check the correctness of a proof, do as follows: Convert all expressions in the proof into parse trees using priority and associativity rules. Then convert all the parse trees into syntax trees using term reduction rules and macro reduction rules. Finally check each step of the proof on basis of the syntax trees.

Hence, another way to state the difference between term reduction rules and graph reduction rules is: Term reduction rules have to be applied both before computing the value of a term and before checking the correctness of a proof. Graph reduction rules are not applied when checking the correctness of a proof.

For each graph reduction rule there is a corresponding axiom (but not vice versa: there are axioms that do not correspond to graph reduction rules). Graph reduction rules may enter proofs through the axioms they correspond to, but when they do, they do so explicitly. Term reduction rules do not correspond to axioms.

For an introduction to term reduction, see Section 4.8. For an overview of the process of forming parse and syntax trees, see Section 4.10. For details on the order of performing term reduction, see Section 4.12. For a list of term

[$x \overset{\circ}{\rightarrow} y$]	x term reduces to y
	term reduction rule
	graph reduction rule
	reduction rule
	macro reduction rule

reduction rules, see Appendix A.3. For ways of making proofs more readable when notation is heavy due to term reduction, see Section 6.13. For the construct $[x, y]$ that has meaning only because of term reduction, see Section 4.11 and Section A.53.

Examples

$[\langle x, y \rangle \dot{\rightarrow} x :: \langle y \rangle]$ introduces a term reduction rule and $[\langle x \rangle \dot{=} x :: \langle \rangle]$ introduces a macro reduction rule. Due to these term and macro reduction rules, $[\langle x, y \rangle]$ and $[x :: y :: \langle \rangle]$ have the same syntax trees.

In contrast, $[\infty\text{-list}(n) \dot{=} n :: \infty\text{-list}(n+1)]$ introduces the graph reduction rule $[\infty\text{-list}(n) \dot{\rightarrow} n :: \infty\text{-list}(n+1)]$ and the corresponding axiom $[\infty\text{-list}(n) \equiv n :: \infty\text{-list}(n+1)]$.

$[\perp \text{ head} \equiv \perp]$ is an example of an axiom that does not correspond to any kind of reduction rule.

See also

$[x \dot{=} y]$ Section A.58

$[x \dot{=} y]$ Section A.72

A.97 x times y $[x \cdot y]$ **Parse tree****Sort**

$[x \cdot y]$ is an operator (i.e. it may occur in the syntax tree of a term).

Description

If $[x]$ and $[y]$ are truth values then $[x \cdot y]$ coincides with $[x \wedge y]$. If $[x]$ and $[y]$ are exact fractions then $[x \cdot y]$ denotes the exact product of $[x]$ and $[y]$. If $[x]$ and $[y]$ are floating fractions of equal precision, then $[x \cdot y]$ denotes the exact product of $[x]$ and $[y]$ rounded to the common precision of $[x]$ and $[y]$ using round to even. If $[x]$ and $[y]$ are floating fractions of different precisions, then $[x \cdot y]$ equals $[\bullet]$. If $[x]$ is an exact fraction and $[y]$ is a floating fraction or vice versa, then $[x \cdot y]$ denotes the exact product of $[x]$ and $[y]$ rounded to the precision of the floating fraction using round to even. $[x \cdot_+ y]$, $[x \cdot_- y]$, $[x \cdot_0 y]$, and $[x \cdot_2 y]$ are versions of $[x \cdot y]$ that use round to plus infinity, minus infinity, zero, and even, respectively. $[x \cdot_2 y]$ is identical to $[x \cdot y]$.

Type table

\cdot	B	D	X	E	\cdot	N	Z	Z⁺	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	Z⁺	N	Z	Z⁺	Z⁻
E	X	X	X	X	Z⁻	Z	Z	Z⁻	Z⁺
\cdot	D_m^∞	D_m	$D_m^{-\infty}$	D_∞^∞	D_∞	$D_\infty^{-\infty}$	D_n^∞	D_n	$D_n^{-\infty}$
D_m^∞	X	X	X	X	X	X	X	X	X
D_m	X	D_m	X	X	D_m	X	X	X	X
$D_m^{-\infty}$	X	X	X	X	X	X	X	X	X
D_∞^∞	X	X	X	X	X	X	\cdot	T	F
D_∞	X	D_m	X	X	D_∞	X	T	T	F
$D_\infty^{-\infty}$	X	X	X	X	X	X	F	F	F

Mac rules

[**Mac rule StrictTimesX** : $\perp \cdot y \equiv \perp$]

$[x \cdot y]$ x times y
 [**StrictTimesX**] rule **StrictTimesX**

[**Mac rule XTimesStrict** : $x \cdot \perp \equiv \perp$]

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

See also

- [$x \wedge y$] Section A.46
- [$x - y$] Section A.73
- [$x + y$] Section A.80
- [$x@y$] Section A.85

[XTimesStrict] rule XTimesStrict
[SubXTimesY] rule SubXTimesY

A.98 x weakly greater than y [$x \geq y$]

Parse tree



Sort

[$x \geq y$] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[$x \geq y$] is true if [x] is weakly greater than [y] (i.e. greater than or equal to). [$x \geq y$] uses the convention that truth is greater than falsehood when comparing truth values. For truth values [x] and [y], [$x \geq y$] coincides with [$x \Leftarrow y$]. When comparing decimal fractions, [$x \geq y$] disregards precision.

Type table

\geq	B	D	X	E	\geq	N	Z	Z⁺	Z⁻
B	B	X	X	X	N	B	B	B	T
D	X	B	X	X	Z	B	B	B	B
X	X	X	X	X	Z⁺	B	B	B	T
E	X	X	X	X	Z⁻	F	B	F	B
\geq	D_m^∞	D_m	$D_m^{-\infty}$	D_∞^∞	D_∞	$D_\infty^{-\infty}$	D_n^∞	D_n	$D_n^{-\infty}$
D_m^∞	T	T	T	T	T	T	T	T	T
D_m	F	B	T	F	B	T	F	B	T
$D_m^{-\infty}$	F	F	T	F	F	T	F	F	T
D_∞^∞	T	T	T	T	T	T	\geq	T	F
D_∞	F	B	T	F	B	T	T	T	T
$D_\infty^{-\infty}$	F	F	T	F	F	T	F	F	T

Mac rules

[**Mac rule StrictWeaklyGreaterThanX** : $\perp \geq y \equiv \perp$]

[**Mac rule XWeaklyGreaterThanStrict** : $x \geq \perp \equiv \perp$]

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

	[$x \geq y$]	x weakly greater than y
[StrictWeaklyGreaterThanX	rule StrictWeaklyGreaterThanX
[XWeaklyGreaterThanStrict	rule XWeaklyGreaterThanStrict
[SubXWeaklyGreaterThanY	rule SubXWeaklyGreaterThanY

See also

- [$x = y$] Section A.54
- [$x > y$] Section A.92
- [$x < y$] Section A.93
- [$x \leq y$] Section A.100
- [$x \Leftarrow y$] Section A.66

A.99 x weakly less information y [$x \preceq y$]

Parse tree



Sort

$[x \preceq y]$ is a directive (i.e. it cannot occur in the syntax tree of a term).

Description

$[x \preceq y]$ holds if any property of $[x]$ is also a property of $[y]$. The following fact explains why $[x \preceq y]$ is relevant to computer science:

Fact A.99.1 If $[f, a_1, a_2, a_3, \dots]^\circ$ are computable by machine and if

$$[a_1, a_2, a_3, \dots]^\circ$$

is an increasing sequence with supremum $[a]$, then

$$[f' a_1, f' a_2, f' a_3, \dots]^\circ$$

is an increasing sequence with supremum $[f' a]$.

Mac rules

[**Mac lemma InfoReflexive** : $x \preceq x$]
 [**Mac rule InfoBottom** : $\perp \preceq x$]
 [**Mac rule InfoImply** : $x \preceq y \vdash f'x \rightarrow f'y$]
 [**Mac rule ImplyInfo** : if $\text{notfree}(f, x) \wedge \text{notfree}(f, y)$ then $f'x \rightarrow f'y \vdash x \preceq y$]
 [**Mac rule InfoAntiSymmetry** : $x \preceq y \vdash y \preceq x \vdash x \equiv y$]
 [**Mac lemma InfoTransitivity** : $x \preceq y \vdash y \preceq z \vdash x \preceq z$]
 [**Mac lemma Monotonicity** : $x \preceq y \vdash g'x \preceq g'y$]
 [**Mac lemma Monotonicity'** : $x \preceq y \vdash \langle \mathcal{A} \mid u:=x \rangle \preceq \langle \mathcal{A} \mid u:=y \rangle$]
 [**Mac rule InfoLambda** : $\mathcal{A} \preceq \mathcal{B} \vdash \lambda x. \mathcal{A} \preceq \lambda x. \mathcal{B}$]
 [**Mac lemma MonotonicityHead** : $x \preceq y \vdash x \text{ head } \preceq y \text{ head}$]
 [**Mac lemma MonotonicityTail** : $x \preceq y \vdash x \text{ tail } \preceq y \text{ tail}$]
 [**Mac lemma MonotonicityHead'** : $x :: y \preceq u :: v \vdash x \preceq u$]
 [**Mac lemma MonotonicityTail'** : $x :: y \preceq u :: v \vdash y \preceq v$]

[$x \preceq y$]	x weakly less information y
[InfoBottom]	rule InfoBottom
[InfoImply]	rule InfoImply
[ImplyInfo]	rule ImplyInfo
[InfoAntiSymmetry]	rule InfoAntiSymmetry
[InfoLambda]	rule InfoLambda

[**Mac lemma MonotonicityHead**⁹ : $x \preceq y \vdash x :: z \preceq y :: z$]
 [**Mac lemma MonotonicityTail**⁹ : $y \preceq z \vdash x :: y \preceq x :: z$]
 [**Mac lemma MonotonicityPair** : $x \preceq u \vdash y \preceq v \vdash x :: y \preceq u :: v$]
 [**Mac rule MinimalY** : $f'x \equiv x \vdash Y'f \preceq x$]
 [**Mac rule Minimal** : $f'x \preceq x \vdash Y'f \preceq x$]
 [**Mac lemma InfoLessBottom** : $x \preceq \perp \vdash x \equiv \perp$]

Examples

[$\perp :: \perp$] \wedge [$\perp :: \perp$] holds
 [$\perp :: \perp$] \wedge [$\perp :: 2$] holds
 [$\perp :: \perp$] \wedge [$\top :: \perp$] holds
 [$\perp :: 2$] \wedge [$\top :: 2$] holds
 [$\top :: \perp$] \wedge [$\top :: 2$] holds
 [$\perp :: \perp$] \wedge [$\top :: 2$] holds
 [$\top :: \perp$] \wedge [$F :: 2$] fails

See also

[\perp] Section A.5
 [$x \equiv y$] Section A.60

[MinimalY] rule MinimalY
 [Minimal] rule Minimal

A.100 x weakly less than y [$x \leq y$]

Parse tree



Sort

[$x \leq y$] is an operator (i.e. it may occur in the syntax tree of a term).

Description

[$x \leq y$] is true if [x] is weakly less than [y] (i.e. less than or equal to). [$x \leq y$] uses the convention that truth is greater than falsehood when comparing truth values. For truth values [x] and [y], [$x \leq y$] coincides with [$x \Rightarrow y$]. When comparing decimal fractions, [$x \leq y$] disregards precision.

Type table

$<$	B	D	X	E	$<$	N	Z	Z⁺	Z⁻
B	B	X	X	X	N	B	B	B	F
D	X	B	X	X	Z	B	B	B	B
X	X	X	X	X	Z⁺	B	B	B	F
E	X	X	X	X	Z⁻	T	B	T	B
\leq	D_m^∞	D_m	$D_m^{-\infty}$	D_∞^∞	D_∞	$D_\infty^{-\infty}$	D_n^∞	D_n	$D_n^{-\infty}$
D_m^∞	T	F	F	T	F	F	T	F	F
D_m	T	B	F	T	B	F	T	B	F
$D_m^{-\infty}$	T	T	T	T	T	T	T	T	T
D_∞^∞	T	F	F	T	F	F	\leq	T	F
D_∞	T	B	F	T	B	F	T	T	F
$D_\infty^{-\infty}$	T	T	T	T	T	T	F	T	T

Mac rules

[Mac rule **StrictWeaklyLessThanX** : $\perp \leq x \equiv \perp$]

[Mac rule **XWeaklyLessThanStrict** : $x \leq \perp \equiv \perp$]

[Mac rule **XWeaklyLessThanY** : $\langle A \leq B \mid x := S \rangle \equiv \langle A \mid x := S \rangle \leq \langle B \mid x := S \rangle$]

[$x \leq y$]	x weakly less than y
[StrictWeaklyLessThanX]	rule StrictWeaklyLessThanX
[XWeaklyLessThanStrict]	rule XWeaklyLessThanStrict
[XWeaklyLessThanY]	rule XWeaklyLessThanY

See also

- [$x = y$] Section A.54
- [$x > y$] Section A.92
- [$x < y$] Section A.93
- [$x \geq y$] Section A.98
- [$x \Rightarrow y$] Section A.67

B.4 Fundamental constructs

$[N]$	The uhr-element.
$[\lambda x.A]$	The function that maps $[x]$ to $[A]$
$[x'y]$	$[x]$ applied to $[y]$
$[\text{case}(x, y, z)]$	$\begin{cases} [y] & \text{if } [x] \text{ is the uhr-element} \\ [z] & \text{if } [x] \text{ is a function} \\ [\perp] & \text{if } [x] \text{ equals } [\perp] \end{cases}$
$[\bar{\epsilon}x]$	The choice operator
$[\bar{\exists}x]$	Fundamental existence

B.5 Definitions

$[\perp]$	$\doteq (\lambda x.x'x)'(\lambda x.x'x)$
$[\hat{F}]$	$\doteq \lambda x.\lambda y.x'y$
$[Y]$	$\doteq \lambda f.(\lambda x.f'(x'x))'(\lambda x.f'(x'x))$
$[T]$	$\doteq N$
$[\hat{F}]$	$\doteq \lambda x.N$
$[\hat{\approx}x]$	$\doteq \text{case}(x, T, \hat{F})$
$\left[x \left\langle \begin{array}{l} y \\ z \end{array} \right. \right]$	$\doteq \text{case}(x, y, z)$
$\left[x \bar{\wedge} y \right]$	$\doteq x \left\langle \begin{array}{l} \text{case}(y, T, \hat{F}) \\ \text{case}(y, \hat{F}, \hat{F}) \end{array} \right.$
$[x \hat{\wedge} y]$	$\doteq \text{case}(x, y, \hat{F})$
$[x \hat{\Rightarrow} y]$	$\doteq \text{case}(x, y, T)$
$[\hat{\wedge}x]$	$\doteq \text{case}(x, \hat{F}, T)$
$[\hat{\exists}x]$	$\doteq \text{case}(x, T, T)$
$\left[x \downarrow y \right]$	$\doteq x \left\langle \begin{array}{l} \text{case}(y, N, \perp) \\ \text{case}(y, \perp, \lambda z.x'z \downarrow y'z) \end{array} \right.$
$[\hat{?}x]$	$\doteq \text{case}(x, T, \perp)$
$[\epsilon x: y]$	$\doteq \bar{\epsilon} \lambda x.y$
$[\bar{\exists}x: y]$	$\doteq \bar{\exists} \lambda x.y$
$[\exists x: y]$	$\doteq \hat{\approx}(\lambda x.y)' \epsilon x: y$
$[\forall x: y]$	$\doteq \hat{\wedge} \bar{\exists}x: \hat{\wedge}y$
$\left[x \hat{=} y \right]$	$\doteq x \left\langle \begin{array}{l} \text{case}(y, T, \hat{F}) \\ \text{case}(y, \hat{F}, \forall z: x'z \hat{=} y'z) \end{array} \right.$
$\left[x \hat{=}_{\mathbf{u}} y \right]$	$\doteq x \left\langle \begin{array}{l} \text{case}(y, T, \hat{F}) \\ \text{case}(y, \hat{F}, \forall z: x'(u'z) \hat{=}_{\mathbf{u}} y'(u'z)) \end{array} \right.$
$[\ell]$	$\doteq Y' \lambda f. \lambda x. x \left\langle \begin{array}{l} T \\ (\forall y: f'(x'y)) \bar{\wedge} \bar{\exists}u: f'u \bar{\wedge} \forall v: \forall w: (v \hat{=}_{\mathbf{u}} w \Rightarrow x'v \hat{=} x'w) \end{array} \right.$

Whenever a term $[\mathcal{A}]$ occurs in a position where a formula is expected, $[\mathcal{A}]$ is shorthand for $[\mathcal{A} \equiv \top]$.

$[\mathcal{A} \rightarrow (\mathcal{B} \equiv \mathcal{C})]$ is shorthand for $[\mathcal{A} \hat{\wedge} \mathcal{B} \equiv \mathcal{A} \hat{\wedge} \mathcal{C}]$.

$[\mathbf{x} \preceq \mathbf{y}]$ is shorthand for $[\mathbf{x} \equiv \mathbf{x} \downarrow \mathbf{y}]$.

B.6 Axioms and inferences

[**Map rule MapTrans** : $\mathcal{A} \equiv \mathcal{B} \vdash \mathcal{A} \equiv \mathcal{C} \vdash \mathcal{B} \equiv \mathcal{C}$]
 [**Map rule MapSubLambda** : $\mathcal{A} \equiv \mathcal{B} \vdash \lambda \mathbf{x} . \mathcal{A} \equiv \lambda \mathbf{x} . \mathcal{B}$]
 [**Map rule MapSubApply** : $\mathcal{A} \equiv \mathcal{B} \vdash \mathcal{C}' \mathcal{A} \equiv \mathcal{C}' \mathcal{B}$]
 [**Map rule MapApplyNil** : $\mathbf{N}' \mathcal{B} \equiv \mathbf{N}$]
 [**Map rule MapApplyLambda** : $(\lambda \mathbf{x} . \mathcal{A})' \mathcal{B} \equiv \langle \mathcal{A} \mid \mathbf{x} := \mathcal{B} \rangle$]
 [**Map rule MapApplyBottom** : $\perp' \mathcal{B} \equiv \perp$]
 [**Map rule MapIfNil** : $\text{case}(\mathbf{N}, \mathcal{B}, \mathcal{C}) \equiv \mathcal{B}$]
 [**Map rule MapIfLambda** : $\text{case}(\lambda \mathbf{x} . \mathcal{A}, \mathcal{B}, \mathcal{C}) \equiv \mathcal{C}$]
 [**Map rule MapIfBottom** : $\text{case}(\perp, \mathcal{B}, \mathcal{C}) \equiv \perp$]
 [**Map rule MapMonotonicity** : $\mathcal{A} \preceq \mathcal{B} \vdash \mathcal{C}' \mathcal{A} \preceq \mathcal{C}' \mathcal{B}$]
 [**Map rule MapMinimality** : $\mathcal{A}' \mathcal{B} \preceq \mathcal{B} \vdash \mathbf{Y}' \mathcal{A} \preceq \mathcal{B}$]
 [**Map rule MapQND** : $\mathcal{A}' \mathbf{N} \equiv \mathcal{B}' \mathbf{N} \vdash \mathcal{A}'(\tilde{\mathcal{F}}' \mathcal{C}) \equiv \mathcal{A}'(\tilde{\mathcal{F}}' \mathcal{C}) \vdash \mathcal{A}' \perp \equiv \mathcal{B}' \perp \vdash \mathcal{A}' \mathcal{C} \equiv \mathcal{B}' \mathcal{C}$]
 [**Map rule MapExtensionality** : if $\text{distinct}(\mathbf{u}, \mathbf{v}) \wedge \text{notfree}(\mathbf{u}, \mathcal{A}) \wedge \text{notfree}(\mathbf{v}, \mathcal{A}) \wedge \text{notfree}(\mathbf{u}, \mathcal{B}) \wedge \text{notfree}(\mathbf{v}, \mathcal{B})$ then $\approx(\mathcal{A}' \mathbf{u}) \equiv \approx(\mathcal{B}' \mathbf{u}) \vdash \mathcal{A}' \mathbf{u}' \mathbf{v} \equiv \mathcal{A}' \mathcal{C} \vdash \mathcal{B}' \mathbf{u}' \mathbf{v} \equiv \mathcal{B}' \mathcal{C} \vdash \mathcal{A}' \mathbf{u} \equiv \mathcal{B}' \mathbf{u}$]
 [**Map rule MapQuantifyA** : $\ell' \mathbf{a} \rightarrow \forall \mathbf{x} : \mathcal{A} \rightarrow \langle \mathcal{A} \mid \mathbf{x} := \mathbf{a} \rangle$]
 [**Map rule MapQuantifyB** : $\forall \mathbf{x} : \hat{\mathcal{A}} \equiv \hat{\forall \mathbf{x} : \mathcal{A}}$]
 [**Map rule MapQuantifyC** : $\forall \mathbf{x} : \hat{\mathcal{A}} \equiv \ell' \varepsilon \mathbf{x} : \mathcal{A}$]
 [**Map rule MapQuantifyD** : $\varepsilon \mathbf{x} : \mathcal{A} \equiv \varepsilon \mathbf{x} : \ell' \mathbf{x} \wedge \mathcal{A}$]
 [**Map rule MapExistsA** : $\mathbf{x}' \mathbf{y} \rightarrow \exists \bar{\mathbf{x}}$]
 [**Map rule MapExistsB** : if $\text{notfree}(\mathbf{y}, \mathbf{u})$ then $\mathbf{x}' \mathbf{y} \rightarrow \mathbf{u}' \mathbf{v} \vdash \exists \bar{\mathbf{x}} \rightarrow \exists \bar{\mathbf{u}}$]

[MapTrans]	rule MapTrans
[MapSubLambda]	rule MapSubLambda
[MapSubApply]	rule MapSubApply
[MapApplyNil]	rule MapApplyNil
[MapApplyLambda]	rule MapApplyLambda
[MapApplyBottom]	rule MapApplyBottom
[MapIfNil]	rule MapIfNil
[MapIfLambda]	rule MapIfLambda
[MapIfBottom]	rule MapIfBottom
[MapMonotonicity]	rule MapMonotonicity
[MapMinimality]	rule MapMinimality
[MapQND]	rule MapQND
[MapExtensionality]	rule MapExtensionality
[MapQuantifyA]	rule MapQuantifyA
[MapQuantifyB]	rule MapQuantifyB
[MapQuantifyC]	rule MapQuantifyC
[MapQuantifyD]	rule MapQuantifyD
[MapExistsA]	rule MapExistsA
[MapExistsB]	rule MapExistsB

[**Map rule MapExistsC** : $\bar{\exists}x \equiv \hat{\exists}x$]

[**MapExistsC**] rule MapExistsC

Appendix C

Index

[$\forall x \in y: z$]	all x in y colon z, 435, 504
[∞ -list(x)]	ascension list of x end, 174
[\perp]	bottom, 127, 505
[$\epsilon x \in y: z$]	choose x in y colon z, 439, 507
[8]	eight, 103
[$\langle \rangle$]	empty list, 170, 510
[\bullet]	exception, 131, 328, 511
[$\exists x \in y: z$]	exists x in y colon z, 437, 512
[5]	five, 103
[4]	four, 103
[∞]	infinity, 138, 521
[$\lambda x.y$]	lambda x dot y, 304, 522
[$-\infty$]	minus infinity, 138
[$-x$]	minus x, 112, 524
[9]	nine, 103
[$\neg x$]	not x, 157, 526
[0]	null, 103
[1]	one, 103
[(x)]	parenthesis x end, 110, 527
[π]	pi, 137
[$+\infty$]	plus infinity, 138
[$+x$]	plus x, 113, 528
[$\#x$]	precision x, 144, 529
[7]	seven, 103
[$\sim x$]	simple not x, 335
[$\langle x \rangle$]	simple tuple x end, 328
[6]	six, 103
[$\langle x \mid y:=z \rangle$]	substitute x where y is z end, 203, 530
[3]	three, 103
[$\langle x \rangle$]	tuple x end, 171, 559
[2]	two, 103

$[x \wedge y]$	x and y, 154, 561
$[x' y]$	x apply y, 304, 563
$[x \rightarrow y]$	x associates as y, 111, 564
$[x \in y]$	x belongs to y, 339, 566
$[x \times y]$	x cartesian y, 343, 568
$\left[x \begin{array}{l} \langle \\ y \\ z \end{array} \right]$	x case y else z end, 327
$[x : y]$	x colon y, 352, 569
$[x, y]$	x comma y, 162, 570
$[x = y]$	x computational equal y, 104, 123, 141, 571
$[x \neq y]$	x computational unequal y, 124, 141, 573
$[x \triangleright y]$	x concludes y, 191, 350, 575
$[x \rightarrow y]$	x deduces y, 369, 576
$[x \doteq y]$	x defined equal y, 104, 577
$[x/y]$	x divide y, 146, 578
$[x \equiv y]$	x equal y, 104, 580
$[x?]$	x exceptional, 133, 582
$[x!]$	x faculty, 124, 583
$[x \succ y]$	x greater priority y, 105, 584
$[x \Leftrightarrow y]$	x if and only if y, 158, 586
$[x \Leftarrow y]$	x implied by y, 158, 588
$[x \Rightarrow y]$	x implies y, 158, 590
$[x \vdash y]$	x infers y, 349, 591
$[x // y]$	x integer divide y, 131, 147, 592
$[x^*]$	x listset, 344, 595
$[x \doteq y]$	x macro equal y, 109, 596
$[x - y]$	x minus y, 103, 144, 597
$[x \% y]$	x modulo y, 131, 147, 599
$[x \supseteq y]$	x modus ponens y, 375, 601
$[x \notin y]$	x not in y, 566
$[x \doteq y]$	x optimised equal y, 113, 602
$[x \vee y]$	x or y, 156, 603
$[x :: y]$	x pair y, 165, 606
$[x + y]$	x plus y, 102, 144, 607
$[x^y]$	x power y end, 103, 148, 609
$[x^-]$	x predecessor, 113, 398, 611
$\left[x \begin{array}{l} \dagger \\ \rightarrow \\ y \end{array} \right]$	x reduces to y, 120, 613
$[x@_ - y]$	x round down y, 143
$[x@_2 y]$	x round even y, 143
$[x@_+ y]$	x round up y, 143
$[x@ y]$	x round y, 142, 614
$[x@_0 y]$	x round zero y, 143
$[x \doteq y]$	x same priority y, 112, 617
$\left[x \begin{array}{l} \{ \\ y \\ z \end{array} \right]$	x select y else z end, 122

- [$x; y$] x semicolon y, 191, 618
- [$x \# y$] x separable y, 419
- [$x \wedge y$] x simple and y, 335
- [$x \doteq y$] x simple equal y, 335
- [$x \dot{\cdot} y$] x simple pair y, 327, 620
- [$x \neq y$] x simple unequal y, 335
- [$x > y$] x strongly greater than y, 124, 141, 622
- [$x < y$] x strongly less than y, 123, 141, 624
- [x^+] x successor, 113, 398, 626
- [$x \overset{\circ}{\rightarrow} y$] x term reduces to y, 159, 628
- [$x \cdot y$] x times y, 102, 144, 630
- [$x \geq y$] x weakly greater than y, 124, 141, 632
- [$x \leq y$] x weakly less information y, 417, 634
- [$x \leq y$] x weakly less than y, 124, 141, 636

- [a] a, 103
- [A] capital a, 103
- [\mathcal{A}] script a, 103
- a [a], 103
- Algebra [Algebra], 191
- [Algebra] Algebra, 191
- algebraic block, 387
- algebraic proof, 185
- algebraic rule, 181
- algebraic system, 186
- all sets, the class of, 362
- all x in y colon z [$\forall x \in y: z$], 435, 504
- alpha-reduction, 316
- alpha-renaming, 316
- alpha-rule, 310
- ambiguous, 164
- and, 153
- and/or, 156
- anonymous binding diagram, 210
- antilemma, 407
- antilemma [Antilemma], 408
- [Antilemma] antilemma, 408
- application of deduction, backward, 384
- application of rule to formula, backward, 384
- [ApplyBottom] rule ApplyBottom, 563
- [ApplyFTwo] rule ApplyFTwo, 518
- [ApplyLambda] rule ApplyLambda, 563
- [ApplyNil] rule ApplyNil, 563
- [ApplyT] rule ApplyT, 563
- approximation, finite, 425
- argument, 105

argumentation, 191, 350
 arithmetic operation, exact, 144
 arity, 106
 ary, [117]-, 105
 ascension list of x end [∞ -list(x)], 174
 associative, left, 111
 associative, right, 111
 associativity rule, 111
 atom, 169
 axiom, 186, 350
 axiom of reflexivity, 195
 axiom, computation, 199
 axiom, definition, 196
 axiom, lemma, 198
 axiomatic system, 186

 [$\overline{\mathbf{B}}$] is truth value, 336
 [\mathbf{B}] the set of truth values, 340, 556
 backchaining, 380
 backchaining, suited for, 383
 backward application of deduction, 384
 backward application of rule to formula, 384
 beta-reduction, 315
 beta-rule, 310
 big number, 323
 bignum, 323
 binary, 105
 binary term, 321
 bind, 207
 binding, 209
 binding construct, 218
 binding diagram, 208
 bit, 321
 block, 372
 block, algebraic, 387
 block, inner, 373
 block, inside, 373
 block, outer, 373
 bottom, 421
 bottom [\perp], 127, 505
 bound, 209
 brute force, 379
 bus, 322

 calculus, lambda, 310
 capital a [\mathbf{A}], 103

- [$\text{case}(x, y, z)$] case x comma y comma z end, 326, 506
 - case x comma y comma z end [$\text{case}(x, y, z)$], 326, 506
- [CaseBottom] rule CaseBottom, 506
- [CaseLambda] rule CaseLambda, 506
- [CaseNil] rule CaseNil, 506
- [CasePair] rule CasePair, 506
- [Cases] rule Cases, 556
- [CaseT] rule CaseT, 506
 - choice operator, 440
 - choose x in y colon z [$\varepsilon x \in y: z$], 439, 507
 - circular, 124
 - circular proof, 360
 - class of all sets, the, 362
 - classical, 340
 - classical [ℓ], 340, 508
 - classical, non-, 340
- [ClassicalBottom] rule ClassicalBottom, 508
- [ClassicalNil] rule ClassicalNil, 508
- [ClassicalPair] rule ClassicalPair, 508
- [ClassicalWellfounded] rule ClassicalWellfounded, 508
 - closed, 365
- [Commutativity] rule Commutativity, 580
 - computable, 425, 436
 - computable, -non, 436
 - computation axiom, 199
- [Computation] rule Computation, 580
 - conclusion, 349, 350, 370
 - conclusion of proof, 352
 - condition, side, 206
 - conjecture, 186
 - conquer, divide and, 381
 - consequence, 370
 - consistent, 187
- [**construct x**] construct x, 162, 509
 - construct x [**construct x**], 162, 509
 - construct, lambda-, 304
 - contents, information, 417
 - continuity, 425
 - contradiction, 186
- [Contradiction] rule Contradiction, 580
 - coordinates, 165
 - correct proof, 352
 - counterexample, 130, 410
 - counterproof, 408
- [CounterTF] rule CounterTF, 409

- [$\overline{\mathbf{D}}$] is decimal fraction, 336
- [$\overline{\mathbf{D}}_\infty$] is exact fraction, 336
- [$\overline{\mathbf{D}}.$] is floating fraction, 336
- [\mathbf{D}] the set of decimal fractions, 340, 533
- [\mathbf{D}_x] the set of decimal fractions of precision x end, 342, 535
- [\mathbf{D}_x^∞] the set of infinities of precision x end, 342, 541
- [$\mathbf{D}_x^{-\infty}$] the set of minus infinities of precision x end, 342, 545
- de Morgan, 395
- decimal fraction, 137
- deduction, 372
- deduction, backward application of, 384
- [DefFOneOfXEnd] rule DefFOneOfXEnd, 516
- [DefFThreeOfXEnd] rule DefFThreeOfXEnd, 517
- [DefFTwo] rule DefFTwo, 518
- definition [Definition], 197
- definition axiom, 196
- [Definition] definition, 197
- [Definition] rule Definition, 580
- definition, local, 439
- definition, macro, 110
- [DefMinimumXCommaYEnd] rule DefMinimumXCommaYEnd, 523
- [DefXFaculty] rule DefXFaculty, 583
- derivation, 349
- derivation proof, 352
- derivation rule, 349
- derivation rule, instance of, 350
- derivation system, 366
- deterministic, 193
- diagram, 422
- directive, 115
- disproves, 408
- distribute, 218
- divide and conquer, 381
- divide, integer [$x // y$], 147
- [DivideNonzero] rule DivideNonzero, 579
- [DivideZero] rule DivideZero, 579
- division, integer, 147
- down, round [$x@_y$], 143

- [$\overline{\mathbf{E}}$] is empty list, 336
- [\mathbf{E}] the set of the empty list, 344, 553
- edge, 107, 423
- eight [8], 103
- either-or, 156
- [ElimAll] rule ElimAll, 504
- [ElimB] rule ElimB, 556

- [ElimCartesianHead] rule ElimCartesianHead, 568
- [ElimCartesianPair] rule ElimCartesianPair, 568
- [ElimCartesianTail] rule ElimCartesianTail, 568
- [ElimD] rule ElimD, 533
- [ElimE] rule ElimE, 553
- [ElimExists] rule ElimExists, 512
- [ElimExistsB] rule ElimExistsB, 512
- [ElimExistsS] rule ElimExistsS, 512
- [ElimF] rule ElimF, 539
- [ElimListSet] rule ElimListSet, 595
- [ElimNotB] rule ElimNotB, 556
- [ElimNotD] rule ElimNotD, 533
- [ElimNotE] rule ElimNotE, 553
- [ElimNotF] rule ElimNotF, 539
- [ElimNotT] rule ElimNotT, 554
- [ElimNotX] rule ElimNotX, 537
- [ElimT] rule ElimT, 554
- [ElimTT] rule ElimTT, 554
- [ElimX] rule ElimX, 537
 - empty list, 170
 - empty list [$\langle \rangle$], 170, 510
 - epsilon operator, Hilberts, 439
 - equal, computational [$x = y$], 123, 141, 571
- [EqualAll] rule EqualAll, 504
- [EqualChoice] rule EqualChoice, 507
- [EqualExists] rule EqualExists, 512
- [EqualPair] rule EqualPair, 571
 - equals, substitution of, 104, 219
 - equation, 130
 - even, round [$x@_2y$], 143
 - even, round to, 142
 - exact arithmetic operation, 144
 - exact fraction, 139
 - example, counter-, 410
 - exception [\bullet], 131, 328, 511
 - exception, more than one, 334
 - exceptional, 133
 - exclusive or, 156
 - existential quantification, 437
 - exists x in y colon z [$\exists x \in y : z$], 437, 512
 - exponent, 149
- [$f_5(x, y)$] f five of x comma y end, 122
- [$f_4(x)$] f four of x end, 109, 515
- [$f_9(x)$] f nine of x end, 365
- [$f_1(x)$] f one of x end, 104, 516

$[f_7(x, y)]$ f seven of x comma y end, 133
 $[f_3(x)]$ f three of x end, 109, 517
 $[f_2]$ f two, 305, 518
 $[F]$ false, 122, 328, 514
 $[\overline{F}]$ is false, 336
 f eight $[f_8]$, 428
 $[f_8]$ f eight, 428
 $[f_7(x)]$ f seven of x end, 428
 f five of x comma y end $[f_5(x, y)]$, 122
 f four of x end $[f_4(x)]$, 109, 515
 f nine of x end $[f_9(x)]$, 365
 f one of x end $[f_1(x)]$, 104, 516
 f seven of x comma y end $[f_7(x, y)]$, 133
 f seven of x end $[f_7(x)]$, 428
 $[F]$ the set of false maps, 342, 539
 f three of x end $[f_3(x)]$, 109, 517
 f two $[f_2]$, 305, 518
 $[\text{fac}]$ faculty, 311
 faculty $[\text{fac}]$, 311
 faculty function, 126
 fail, 130
 false $[F]$, 122, 328, 514
 feasible, 379
 finite approximations, 425
 five $[5]$, 103
 fixation, variable, 377
 fixed point, 318
 fixed point, minimal, 427
 fixed variable, 377
 fixity, 106
 fixpoint $[Y]$, 317
 floating fraction, 139
 floating point standard, IEEE, 143
 force, 167
 force, brute, 379
 form, normal, 121
 four $[4]$, 103
 fraction, decimal, 137
 fraction, exact, 139
 fraction, floating, 139
 free, 208
 free for, 217
 fresh variable, 315
 function, 301, 307
 function, type, 334

- [Gen] rule Gen, 504
 - generalisation, 436
 - graph reduction, 119
 - graph reduction rule, 628
 - greater than, strongly $[x > y]$, 124, 141, 622
 - greater than, weakly $[x \geq y]$, 124, 141, 632
- head of list, 173
- [HeadBottom] rule HeadBottom, 585
- [HeadEmpty] rule HeadEmpty, 585
- [HeadPair] rule HeadPair, 585
 - Hilbert style proof, 349
 - Hilberts epsilon operator, 439
- hold, 130
- hypothesis, 370
- identical, 109
- identical except for naming of bound variables, 210
- IEEE floating point standard, 143
- [if(x, y, z)] if x then y else z end, 328
- [if(x, y, z)] if x then y else z end, 122, 519
 - if x then y else z end [if(x, y, z)], 328
 - if x then y else z end [if(x, y, z)], 122, 519
- [IfBottom] rule IfBottom, 519
- [IfD] rule IfD, 519
- [IfE] rule IfE, 519
- [IfException] rule IfException, 519
- [IfF] rule IfF, 519
- [IfFalse] rule IfFalse, 519
- [IfNil] rule IfNil, 519
- [IfPair] rule IfPair, 519
- [IfT] rule IfT, 519
- [IfTrue] rule IfTrue, 519
- [IfX] rule IfX, 519
 - implication, 369
 - imply, 370
- [ImplyInfo] rule ImplyInfo, 634
 - inclusive or, 156
 - inconsistent, 187, 366
 - increasing sequence, 425
 - index, 102
 - indexed capital l x $[Lx]$, 353
 - induction, 398
 - induction principle, 398
- [Induction] rule Induction, 547
 - induction, Peano, 398

- infer, 370
- inference rule, 350
- infinite list, potentially, 175
- infinite, potentially, 323
- infinity $[\infty]$, 138, 521
- infix, 106
- [InfoAntiSymmetry] rule InfoAntiSymmetry, 634
- [InfoBottom] rule InfoBottom, 634
- [InfoImply] rule InfoImply, 634
- [InfoLambda] rule InfoLambda, 634
- [InfoLessBottom] rule InfoLessBottom, 427
- [InfoReflexive] rule InfoReflexive, 420
- information contents, 417
- information, weakly less $[x \preceq y]$, 417, 634
- informative, 126
- [InfoTransitivity] rule InfoTransitivity, 420
- inner block, 373
- inseparable, 419
- inside, 373
- instance of derivation rule, 350
- instantiation statement, 350
- integer, 123, 140, 543
- integer division, 147
- integer, negative, 549
- integer, positive, 551
- [IntegerDivideZero] rule IntegerDivideZero, 593
- [IntroB] rule IntroB, 556
- [IntroCartesian] rule IntroCartesian, 568
- [IntroCartesianPair] rule IntroCartesianPair, 568
- [IntroD] rule IntroD, 533
- [IntroE] rule IntroE, 553
- [IntroExists] rule IntroExists, 512
- [IntroF] rule IntroF, 539
- [IntroListSet] rule IntroListSet, 595
- [IntroListSetEmpty] rule IntroListSetEmpty, 595
- [IntroListSetPair] rule IntroListSetPair, 595
- [IntroNotB] rule IntroNotB, 556
- [IntroNotD] rule IntroNotD, 533
- [IntroNotE] rule IntroNotE, 553
- [IntroNotF] rule IntroNotF, 539
- [IntroNotT] rule IntroNotT, 554
- [IntroNotX] rule IntroNotX, 537
- [IntroT] rule IntroT, 554
- [IntroTT] rule IntroTT, 554
- [IntroX] rule IntroX, 537
- [InverseTypeAll] rule InverseTypeAll, 504

- [InverseTypeExists] rule InverseTypeExists, 512
 - is decimal fraction [$\overline{\mathbf{D}}$], 336
 - is empty list [$\overline{\mathbf{E}}$], 336
 - is exact fraction [$\overline{\mathbf{D}}_{\infty}$], 336
 - is exception [$\overline{\mathbf{X}}$], 336
 - is false [$\overline{\mathbf{F}}$], 336
 - is floating fraction [$\overline{\mathbf{D}}.$], 336
 - is pair [$\overline{\mathbf{P}}$], 336
 - is true [$\overline{\mathbf{T}}$], 336
 - is truth value [$\overline{\mathbf{B}}$], 336
 - italics, 102

- [ℓ] classical, 340, 508
- [\mathbf{Lx}] indexed capital l x, 353
 - label, 107
 - lambda calculus, 310
 - lambda x dot y [$\lambda x.y$], 304, 522
 - lambda-construct, 304
- [LambdaSub] rule LambdaSub, 522
 - lazy, 129, 167
 - left associative, 111
 - lemma, 186
 - lemma axiom, 198
- [**x lemma y:z**] x lemma y colon z, 190, 594
 - lemma, anti-, 407
 - less information, weakly [$x \preceq y$], 417, 634
 - less than, strongly [$x < y$], 123, 141, 624
 - less than, weakly [$x \leq y$], 124, 141, 636
 - line, proof, 191
 - lines, proof, 352
 - Lisp, 171
 - list, 165, 169
 - list, empty, 170
 - list, head of, 173
 - list, minimum of [$\text{minlist}(x)$], 174
 - list, n [$\text{nlist}(x)$], 172
 - list, n n [$\text{nnlist}(x)$], 173
 - list, potentially infinite, 175
 - list, tail of, 174
 - local definition, 439

- mac [**Mac**], 188
- [**Mac**] mac, 188
 - machine, Turing, 323
 - macro, 110
 - macro definition, 110

- macro reduction, 110, 161
- macro reduction rule, 628
- magnitude, 139
- main theorem, 186
- mantissa, 149
- map, 324
- map theory, 324, 443, 639
- map, nil, 325
- [MapApplyBottom] rule MapApplyBottom, 641
- [MapApplyLambda] rule MapApplyLambda, 641
- [MapApplyNil] rule MapApplyNil, 641
- [MapExistsA] rule MapExistsA, 641
- [MapExistsB] rule MapExistsB, 641
- [MapExistsC] rule MapExistsC, 642
- [MapExtensionality] rule MapExtensionality, 641
- [MapIfBottom] rule MapIfBottom, 641
- [MapIfLambda] rule MapIfLambda, 641
- [MapIfNil] rule MapIfNil, 641
- [MapMinimality] rule MapMinimality, 641
- [MapMonotonicity] rule MapMonotonicity, 641
- [MapQND] rule MapQND, 641
- [MapQuantifyA] rule MapQuantifyA, 641
- [MapQuantifyB] rule MapQuantifyB, 641
- [MapQuantifyC] rule MapQuantifyC, 641
- [MapQuantifyD] rule MapQuantifyD, 641
- [MapSubApply] rule MapSubApply, 641
- [MapSubLambda] rule MapSubLambda, 641
- [MapTrans] rule MapTrans, 641
- membership relation, 339
- [min(x,y)] minimum x comma y end, 138, 523
- minimal fixed point, 427
- [Minimal] rule Minimal, 635
- minimal solution, 427
- [MinimalY] rule MinimalY, 635
- minimum of list x end [minlist(x)], 174
- minimum x comma y end [min(x,y)], 138, 523
- [minlist(x)] minimum of list x end, 174
- minus infinity [$-\infty$], 138
- minus x [$-x$], 112, 524
- minus, unary, 112
- [MinusStrict] rule MinusStrict, 524
- [ModuloZero] rule ModuloZero, 600
- [ModusPonens] rule ModusPonens, 576
- [Monotonicity] rule Monotonicity, 421
- [Monotonicity'] rule Monotonicity', 422
- [MonotonicityHead] rule MonotonicityHead, 422

- [MonotonicityHead'] rule MonotonicityHead', 422
- [MonotonicityHead"] rule MonotonicityHead", 422
- [MonotonicityPair] rule MonotonicityPair, 422
- [MonotonicityTail] rule MonotonicityTail, 422
- [MonotonicityTail'] rule MonotonicityTail', 422
- [MonotonicityTail"] rule MonotonicityTail", 422
 - more than one exception, 334

- [N] nil map, 325
 - n list of x end [nlist(x)], 172
 - n n list of x end [nnlist(x)], 173
- [N] nil map, 525
- [N] the set of natural numbers, 339, 342, 547
 - natural number, 123, 140, 547
 - natural numbers, set of, 339
 - negative integer, 549
 - nil map, 325
 - nil map [N], 325, 525
 - nine [9], 103
- [nlist(x)] n list of x end, 172
- [nnlist(x)] n n list of x end, 173
- node, 107
 - node, root, 107
- non-classical, 340
- non-computable, 436
- non-deterministic, 193
- non-informative, 126
- non-strict, 128
- non-strict in [x], 128
- normal form, 121
- not x [$\neg x$], 157, 526
- [NotStrict] rule NotStrict, 526
- null [0], 103
- null-ary, 105
- number, big, 323
- number, natural, 140, 547
- numbers, set of natural, 339
- numeral, 206
- numeral type rule, 357

- one [1], 103
- operation, exact arithmetic, 144
- operator, 105, 110, 115
 - operator, choice, 440
 - operator, Hilberts epsilon, 439
 - operator, principal, 108

outer block, 373
 outfix, 106

 $[\overline{\mathbf{P}}]$ is pair, 336
 pair, 165
 $[\text{PairIsNotAtom}]$ rule PairIsNotAtom, 565
 $[\text{PairIsPair}]$ rule PairIsPair, 605
 parenthesis x end $[(x)]$, 110, 527
 parse reduction, 110, 161
 parse tree, 110, 161
 Peano induction, 398
 $[\text{PeanoA}]$ rule PeanoA, 547
 $[\text{PeanoB}]$ rule PeanoB, 547
 $[\text{PeanoC}]$ rule PeanoC, 547
 $[\text{PeanoD}]$ rule PeanoD, 547
 $[\text{PeanoD}']$ rule PeanoD', 547
 $[\text{PeanoE}]$ rule PeanoE, 547
 pi $[\pi]$, 137
 plus infinity $[+\infty]$, 138
 plus x $[+x]$, 113, 528
 plus, unary, 113
 $[\text{PlusStrict}]$ rule PlusStrict, 528
 point, fixed, 318
 point, minimal fixed, 427
 positive integer, 551
 potentially infinite, 323
 potentially infinite list, 175
 precision, 139
 precision x $[\#x]$, 144, 529
 $[\text{PrecisionStrict}]$ rule PrecisionStrict, 529
 predecessor, 398
 prefix, 106
 premise, 360, 370
 premisses, 349
 principal operator, 108
 principle, induction, 398
 priority, 105
 proof line, 191, 352
 proof step, 185
 proof, algebraic, 185
 proof, circular, 360
 proof, conclusion of, 352
 proof, correct, 352
 proof, counter-, 408
 proof, derivation, 352
 proof, Hilbert style, 349

- property, 415
- property, universal, 416
- [Pük] pyk, 102
- pyk [Pük], 102

- quantification, existential, 437
- quantification, universal, 435

- recursive, 124
- redex, 119
- reduct, 120
- reduction, 119
- reduction rule, 120, 628
- reduction rule, graph, 628
- reduction rule, macro, 628
- reduction rule, term, 159, 628
- reduction step, 119
- reduction, alpha-, 316
- reduction, beta-, 315
- reduction, macro, 110, 161
- reduction, parse, 110, 161
- reduction, term, 159, 161
- referential transparency, 109, 302
- reflexivity [Reflexivity], 195
- [Reflexivity] reflexivity, 195
- [Reflexivity] rule Reflexivity, 580
- reflexivity, axiom of, 195
- relation, membership, 339
- [Rename] rule Rename, 580
- renaming, alpha-, 316
- [Repetition] rule Repetition, 580
- replace [Replace], 191
- [Replace] replace, 191
- [Replace] rule Replace, 580
- [Replace'] rule Replace', 581
- replacement, 182, 363
- replacement, reverse, 185, 363
- represent, strongly, 329
- represent, weakly, 329
- retract, 332
- reverse [Reverse], 191
- reverse replacement, 185, 363
- [Reverse] reverse, 191
- [Reverse] rule Reverse, 580
- [Reverse'] rule Reverse', 581
- right associative, 111

root, 107
 root node, 107
 round to even, 142
 [RoundDiDn] rule RoundDiDn, 615
 [RoundDmDn] rule RoundDmDn, 615
 [RoundException] rule RoundException, 615
 [RoundIiIn] rule RoundIiIn, 615
 [RoundImIn] rule RoundImIn, 615
 [RoundMiMn] rule RoundMiMn, 615
 [RoundMmMn] rule RoundMmMn, 615
 rule, 189, 350
 rule ApplyBottom [ApplyBottom], 563
 rule ApplyFTwo [ApplyFTwo], 518
 rule ApplyLambda [ApplyLambda], 563
 rule ApplyNil [ApplyNil], 563
 rule ApplyT [ApplyT], 563
 rule CaseBottom [CaseBottom], 506
 rule CaseLambda [CaseLambda], 506
 rule CaseNil [CaseNil], 506
 rule CasePair [CasePair], 506
 rule Cases [Cases], 556
 rule CaseT [CaseT], 506
 rule ClassicalBottom [ClassicalBottom], 508
 rule ClassicalNil [ClassicalNil], 508
 rule ClassicalPair [ClassicalPair], 508
 rule ClassicalWellfounded [ClassicalWellfounded], 508
 rule Commutativity [Commutativity], 580
 rule Computation [Computation], 580
 rule Contradiction [Contradiction], 580
 rule CounterTF [CounterTF], 409
 rule DefFOneOfXEnd [DefFOneOfXEnd], 516
 rule DefFThreeOfXEnd [DefFThreeOfXEnd], 517
 rule DefFTwo [DefFTwo], 518
 rule Definition [Definition], 580
 rule DefMinimumXCommaYEnd [DefMinimumXCommaYEnd], 523
 rule DefXFaculty [DefXFaculty], 583
 rule DivideNonzero [DivideNonzero], 579
 rule DivideZero [DivideZero], 579
 rule ElimAll [ElimAll], 504
 rule ElimB [ElimB], 556
 rule ElimCartesianHead [ElimCartesianHead], 568
 rule ElimCartesianPair [ElimCartesianPair], 568
 rule ElimCartesianTail [ElimCartesianTail], 568
 rule ElimD [ElimD], 533
 rule ElimE [ElimE], 553
 rule ElimExists [ElimExists], 512

rule ElimExistsB [ElimExistsB], 512
rule ElimExistsS [ElimExistsS], 512
rule ElimF [ElimF], 539
rule ElimListSet [ElimListSet], 595
rule ElimNotB [ElimNotB], 556
rule ElimNotD [ElimNotD], 533
rule ElimNotE [ElimNotE], 553
rule ElimNotF [ElimNotF], 539
rule ElimNotT [ElimNotT], 554
rule ElimNotX [ElimNotX], 537
rule ElimT [ElimT], 554
rule ElimTT [ElimTT], 554
rule ElimX [ElimX], 537
rule EqualAll [EqualAll], 504
rule EqualChoice [EqualChoice], 507
rule EqualExists [EqualExists], 512
rule EqualPair [EqualPair], 571
rule Gen [Gen], 504
rule HeadBottom [HeadBottom], 585
rule HeadEmpty [HeadEmpty], 585
rule HeadPair [HeadPair], 585
rule IfBottom [IfBottom], 519
rule IfD [IfD], 519
rule IfE [IfE], 519
rule IfException [IfException], 519
rule IfF [IfF], 519
rule IfFalse [IfFalse], 519
rule IfNil [IfNil], 519
rule IfPair [IfPair], 519
rule IfT [IfT], 519
rule IfTrue [IfTrue], 519
rule IfX [IfX], 519
rule ImplyInfo [ImplyInfo], 634
rule Induction [Induction], 547
rule InfoAntiSymmetry [InfoAntiSymmetry], 634
rule InfoBottom [InfoBottom], 634
rule InfoImply [InfoImply], 634
rule InfoLambda [InfoLambda], 634
rule InfoLessBottom [InfoLessBottom], 427
rule InfoReflexive [InfoReflexive], 420
rule InfoTransitivity [InfoTransitivity], 420
rule IntegerDivideZero [IntegerDivideZero], 593
rule IntroB [IntroB], 556
rule IntroCartesian [IntroCartesian], 568
rule IntroCartesianPair [IntroCartesianPair], 568
rule IntroD [IntroD], 533

rule IntroE [IntroE], 553
rule IntroExists [IntroExists], 512
rule IntroF [IntroF], 539
rule IntroListSet [IntroListSet], 595
rule IntroListSetEmpty [IntroListSetEmpty], 595
rule IntroListSetPair [IntroListSetPair], 595
rule IntroNotB [IntroNotB], 556
rule IntroNotD [IntroNotD], 533
rule IntroNotE [IntroNotE], 553
rule IntroNotF [IntroNotF], 539
rule IntroNotT [IntroNotT], 554
rule IntroNotX [IntroNotX], 537
rule IntroT [IntroT], 554
rule IntroTT [IntroTT], 554
rule IntroX [IntroX], 537
rule InverseTypeAll [InverseTypeAll], 504
rule InverseTypeExists [InverseTypeExists], 512
rule LambdaSub [LambdaSub], 522
rule MapApplyBottom [MapApplyBottom], 641
rule MapApplyLambda [MapApplyLambda], 641
rule MapApplyNil [MapApplyNil], 641
rule MapExistsA [MapExistsA], 641
rule MapExistsB [MapExistsB], 641
rule MapExistsC [MapExistsC], 642
rule MapExtensionality [MapExtensionality], 641
rule MapIfBottom [MapIfBottom], 641
rule MapIfLambda [MapIfLambda], 641
rule MapIfNil [MapIfNil], 641
rule MapMinimality [MapMinimality], 641
rule MapMonotonicity [MapMonotonicity], 641
rule MapQND [MapQND], 641
rule MapQuantifyA [MapQuantifyA], 641
rule MapQuantifyB [MapQuantifyB], 641
rule MapQuantifyC [MapQuantifyC], 641
rule MapQuantifyD [MapQuantifyD], 641
rule MapSubApply [MapSubApply], 641
rule MapSubLambda [MapSubLambda], 641
rule MapTrans [MapTrans], 641
rule Minimal [Minimal], 635
rule MinimalY [MinimalY], 635
rule MinusStrict [MinusStrict], 524
rule ModuloZero [ModuloZero], 600
rule ModusPonens [ModusPonens], 576
rule Monotonicity [Monotonicity], 421
rule Monotonicity' [Monotonicity'], 422
rule MonotonicityHead [MonotonicityHead], 422

rule MonotonicityHead' [MonotonicityHead'], 422
rule MonotonicityHead'' [MonotonicityHead''], 422
rule MonotonicityPair [MonotonicityPair], 422
rule MonotonicityTail [MonotonicityTail], 422
rule MonotonicityTail' [MonotonicityTail'], 422
rule MonotonicityTail'' [MonotonicityTail''], 422
rule NotStrict [NotStrict], 526
rule PairIsNotAtom [PairIsNotAtom], 565
rule PairIsPair [PairIsPair], 605
rule PeanoA [PeanoA], 547
rule PeanoB [PeanoB], 547
rule PeanoC [PeanoC], 547
rule PeanoD [PeanoD], 547
rule PeanoD' [PeanoD'], 547
rule PeanoE [PeanoE], 547
rule PlusStrict [PlusStrict], 528
rule PrecisionStrict [PrecisionStrict], 529
rule Reflexivity [Reflexivity], 580
rule Rename [Rename], 580
rule Repetition [Repetition], 580
rule Replace [Replace], 580
rule Replace' [Replace'], 581
rule Reverse [Reverse], 580
rule Reverse' [Reverse'], 581
rule RoundDiDn [RoundDiDn], 615
rule RoundDmDn [RoundDmDn], 615
rule RoundException [RoundException], 615
rule RoundIiIn [RoundIiIn], 615
rule RoundImIn [RoundImIn], 615
rule RoundMiMn [RoundMiMn], 615
rule RoundMmMn [RoundMmMn], 615
rule SetB [SetB], 556
rule SetD [SetD], 533
rule SetDi [SetDi], 535
rule SetDm [SetDm], 535
rule SetE [SetE], 553
rule SetF [SetF], 539
rule SetIi [SetIi], 541
rule SetIm [SetIm], 541
rule SetMi [SetMi], 545
rule SetMm [SetMm], 545
rule SetN [SetN], 547
rule SetT [SetT], 554
rule SetX [SetX], 537
rule SetXCartesianY [SetXCartesianY], 568
rule SetXListSet [SetXListSet], 595

rule SetZ [SetZ], 543
rule SetZm [SetZm], 549
rule SetZp [SetZp], 551
rule SimpleHeadBottom [SimpleHeadBottom], 619
rule SimpleHeadNil [SimpleHeadNil], 619
rule SimpleHeadPair [SimpleHeadPair], 619
rule SimpleTailBottom [SimpleTailBottom], 621
rule SimpleTailNil [SimpleTailNil], 621
rule SimpleTailPair [SimpleTailPair], 621
rule StrictAndX [StrictAndX], 561
rule StrictAtom [StrictAtom], 565
rule StrictBelongsToY [StrictBelongsToY], 566
rule StrictComputationalEqualX [StrictComputationalEqualX], 571
rule StrictComputationalUnequalX [StrictComputationalUnequalX], 573
rule StrictDivideX [StrictDivideX], 578
rule StrictExceptional [StrictExceptional], 582
rule StrictFaculty [StrictFaculty], 583
rule StrictIfAndOnlyIfX [StrictIfAndOnlyIfX], 586
rule StrictImpliedByX [StrictImpliedByX], 588
rule StrictImpliesX [StrictImpliesX], 590
rule StrictIntegerDivideX [StrictIntegerDivideX], 592
rule StrictMinusX [StrictMinusX], 597
rule StrictModuloX [StrictModuloX], 600
rule StrictOrX [StrictOrX], 603
rule StrictPair [StrictPair], 605
rule StrictPlusX [StrictPlusX], 607
rule StrictPowerXEnd [StrictPowerXEnd], 610
rule StrictRoundX [StrictRoundX], 614
rule StrictStronglyGreaterThanX [StrictStronglyGreaterThanX], 622
rule StrictStronglyLessThanX [StrictStronglyLessThanX], 624
rule StrictTimesX [StrictTimesX], 630
rule StrictWeaklyGreaterThanX [StrictWeaklyGreaterThanX], 632
rule StrictWeaklyLessThanX [StrictWeaklyLessThanX], 636
rule SubB [SubB], 556
rule SubBottom [SubBottom], 505
rule SubCase [SubCase], 506
rule SubClassical [SubClassical], 508
rule SubD [SubD], 533
rule SubDm [SubDm], 535
rule SubE [SubE], 553
rule SubEmptyList [SubEmptyList], 510
rule SubException [SubException], 511
rule SubF [SubF], 539
rule SubFalse [SubFalse], 514
rule SubFOneOfXEnd [SubFOneOfXEnd], 516
rule SubFThreeOfXEnd [SubFThreeOfXEnd], 517

rule SubFTwo [SubFTwo], 518
rule SubIf [SubIf], 519
rule SubIi [SubIi], 541
rule SubInfinity [SubInfinity], 521
rule SubLambdaXX [SubLambdaXX], 522
rule SubLambdaXY [SubLambdaXY], 522
rule SubLambdaXYFreeFor [SubLambdaXYFreeFor], 522
rule SubMinimumXCommaYEnd [SubMinimumXCommaYEnd], 523
rule SubMinusX [SubMinusX], 524
rule SubMm [SubMm], 545
rule SubN [SubN], 547
rule SubNilMap [SubNilMap], 525
rule SubNotX [SubNotX], 526
rule SubNumeral [SubNumeral], 530
rule SubPlusX [SubPlusX], 528
rule SubPrecisionX [SubPrecisionX], 529
rule SubRound [SubRound], 615
rule SubSet [SubSet], 532
rule Substitution [Substitution], 530
rule SubT [SubT], 554
rule SubTrue [SubTrue], 558
rule SubtypeBC [SubtypeBC], 556
rule SubtypeDC [SubtypeDC], 533
rule SubtypeDiD [SubtypeDiD], 535
rule SubtypeDmD [SubtypeDmD], 535
rule SubtypeEC [SubtypeEC], 553
rule SubtypeFB [SubtypeFB], 539
rule SubtypeFC [SubtypeFC], 539
rule SubtypeIiD [SubtypeIiD], 541
rule SubtypeImD [SubtypeImD], 541
rule SubtypeMiD [SubtypeMiD], 545
rule SubtypeMmD [SubtypeMmD], 545
rule SubtypeNotBC [SubtypeNotBC], 556
rule SubtypeNotDC [SubtypeNotDC], 533
rule SubtypeNotEC [SubtypeNotEC], 553
rule SubtypeNotFC [SubtypeNotFC], 539
rule SubtypeNotSetC [SubtypeNotSetC], 566
rule SubtypeNotTC [SubtypeNotTC], 554
rule SubtypeNotXC [SubtypeNotXC], 537
rule SubtypeNZ [SubtypeNZ], 547
rule SubtypeSetC [SubtypeSetC], 566
rule SubtypeTB [SubtypeTB], 554
rule SubtypeTC [SubtypeTC], 554
rule SubtypeXC [SubtypeXC], 537
rule SubtypeZD [SubtypeZD], 543
rule SubtypeZDi [SubtypeZDi], 543

rule SubtypeZmZ [SubtypeZmZ], 549
 rule SubtypeZpN [SubtypeZpN], 551
 rule SubtypeZpZ [SubtypeZpZ], 551
 rule SubX [SubX], 537
 rule SubXAndY [SubXAndY], 561
 rule SubXApplyY [SubXApplyY], 563
 rule SubXAtom [SubXAtom], 565
 rule SubXBelongsToY [SubXBelongsToY], 566
 rule SubXCartesianY [SubXCartesianY], 568
 rule SubXComputationalEqualY [SubXComputationalEqualY], 571
 rule SubXComputationalUnequalY [SubXComputationalUnequalY], 573
 rule SubXDivideY [SubXDivideY], 578
 rule SubXExceptional [SubXExceptional], 582
 rule SubXFaculty [SubXFaculty], 583
 rule SubXHead [SubXHead], 585
 rule SubXIfAndOnlyIfY [SubXIfAndOnlyIfY], 586
 rule SubXImpliedByY [SubXImpliedByY], 588
 rule SubXImpliesY [SubXImpliesY], 590
 rule SubXIntegerDivideY [SubXIntegerDivideY], 592
 rule SubXListSet [SubXListSet], 595
 rule SubXMinusY [SubXMinusY], 598
 rule SubXModuloY [SubXModuloY], 600
 rule SubXOrY [SubXOrY], 603
 rule SubXPair [SubXPair], 605
 rule SubXPairY [SubXPairY], 606
 rule SubXPlusY [SubXPlusY], 608
 rule SubXPowerYEnd [SubXPowerYEnd], 610
 rule SubXPredecessor [SubXPredecessor], 611
 rule SubXSimpleHead [SubXSimpleHead], 619
 rule SubXSimplePairY [SubXSimplePairY], 620
 rule SubXSimpleTail [SubXSimpleTail], 621
 rule SubXStronglyGreaterThanY [SubXStronglyGreaterThanY], 622
 rule SubXStronglyLessThanY [SubXStronglyLessThanY], 624
 rule SubXSubX [SubXSubX], 530
 rule SubXSubY [SubXSubY], 530
 rule SubXSubYNotFree [SubXSubYNotFree], 530
 rule SubXSuccessor [SubXSuccessor], 626
 rule SubXTail [SubXTail], 627
 rule SubXTimesY [SubXTimesY], 631
 rule SubXWeaklyGreaterThanY [SubXWeaklyGreaterThanY], 632
 rule SubXX [SubXX], 530
 rule SubXY [SubXY], 530
 rule SubZ [SubZ], 543
 rule SubZm [SubZm], 549
 rule SubZp [SubZp], 551
 rule TailBottom [TailBottom], 627

rule TailEmpty [TailEmpty], 627
 rule TailPair [TailPair], 627
 rule Tautology [Tautology], 556
 rule Transitivity [Transitivity], 580
 rule TUnequalF [TUnequalF], 407
 rule TypeAll [TypeAll], 504
 rule TypeCInSet [TypeCInSet], 566
 rule TypeDiIntegerDivideDi [TypeDiIntegerDivideDi], 593
 rule TypeDiIntegerDivideDm [TypeDiIntegerDivideDm], 593
 rule TypeDiModuloDi [TypeDiModuloDi], 600
 rule TypeDiModuloDm [TypeDiModuloDm], 600
 rule TypeDmIntegerDivideDi [TypeDmIntegerDivideDi], 593
 rule TypeDmIntegerDivideDm [TypeDmIntegerDivideDm], 593
 rule TypeDmIntegerDivideDn [TypeDmIntegerDivideDn], 593
 rule TypeDmModuloDi [TypeDmModuloDi], 600
 rule TypeDmModuloDm [TypeDmModuloDm], 600
 rule TypeDmModuloDn [TypeDmModuloDn], 600
 rule TypeEmptyListInE [TypeEmptyListInE], 510
 rule TypeExists [TypeExists], 512
 rule TypeFInB [TypeFInB], 514
 rule TypeFInF [TypeFInF], 514
 rule TypeIInIi [TypeIInIi], 521
 rule TypeImInIm [TypeImInIm], 521
 rule TypeMiInMi [TypeMiInMi], 521
 rule TypeMmInMm [TypeMmInMm], 521
 rule TypeNumeralInD [TypeNumeralInD], 533
 rule TypeNumeralInDi [TypeNumeralInDi], 535
 rule TypeNumeralInDm [TypeNumeralInDm], 535
 rule TypeNumeralInN [TypeNumeralInN], 547
 rule TypeNumeralInZ [TypeNumeralInZ], 543
 rule TypeNumeralInZm [TypeNumeralInZm], 549
 rule TypeNumeralInZp [TypeNumeralInZp], 551
 rule TypeNumeralNotInB [TypeNumeralNotInB], 556
 rule TypeNumeralNotInDi [TypeNumeralNotInDi], 535
 rule TypeNumeralNotInDm [TypeNumeralNotInDm], 535
 rule TypeNumeralNotInE [TypeNumeralNotInE], 553
 rule TypeNumeralNotInF [TypeNumeralNotInF], 539
 rule TypeNumeralNotInIi [TypeNumeralNotInIi], 541
 rule TypeNumeralNotInIm [TypeNumeralNotInIm], 541
 rule TypeNumeralNotInMi [TypeNumeralNotInMi], 545
 rule TypeNumeralNotInMm [TypeNumeralNotInMm], 545
 rule TypeNumeralNotInN [TypeNumeralNotInN], 547
 rule TypeNumeralNotInT [TypeNumeralNotInT], 554
 rule TypeNumeralNotInX [TypeNumeralNotInX], 537
 rule TypeNumeralNotInZ [TypeNumeralNotInZ], 543
 rule TypeNumeralNotInZm [TypeNumeralNotInZm], 549

rule TypeNumeralNotInZp [TypeNumeralNotInZp], 551
 rule TypeTInB [TypeTInB], 558
 rule TypeTInT [TypeTInT], 558
 rule TypeXInX [TypeXInX], 511
 rule UnequalPair [UnequalPair], 573
 [x rule y:z] x rule y colon z, 189, 616
 rule XAndStrict [XAndStrict], 561
 rule XBelongsToStrict [XBelongsToStrict], 566
 rule XComputationalEqualStrict [XComputationalEqualStrict], 571
 rule XComputationalUnequalStrict [XComputationalUnequalStrict], 573
 rule XDivideStrict [XDivideStrict], 578
 rule XIfAndOnlyIfStrict [XIfAndOnlyIfStrict], 586
 rule XImpliedByStrict [XImpliedByStrict], 588
 rule XImpliesStrict [XImpliesStrict], 590
 rule XIntegerDivideStrict [XIntegerDivideStrict], 592
 rule XMinusStrict [XMinusStrict], 598
 rule XModuloStrict [XModuloStrict], 600
 rule XOrStrict [XOrStrict], 603
 rule XPlusStrict [XPlusStrict], 608
 rule XPowerStrictEnd [XPowerStrictEnd], 610
 rule XRoundStrict [XRoundStrict], 614
 rule XStronglyGreaterThanStrict [XStronglyGreaterThanStrict], 622
 rule XStronglyLessThanStrict [XStronglyLessThanStrict], 624
 rule XTimesStrict [XTimesStrict], 631
 rule XWeaklyGreaterThanStrict [XWeaklyGreaterThanStrict], 632
 rule XWeaklyLessThanStrict [XWeaklyLessThanStrict], 636
 rule XWeaklyLessThanY [XWeaklyLessThanY], 636
 rule, algebraic, 181
 rule, alpha-, 310
 rule, associativity, 111
 rule, beta-, 310
 rule, derivation, 349
 rule, graph reduction, 628
 rule, inference, 350
 rule, instance of derivation, 350
 rule, macro reduction, 628
 rule, numeral type, 357
 rule, reduction, 120, 628
 rule, subtype, 361
 rule, term reduction, 159, 628

 script a [\mathcal{A}], 103
 semantics, 302
 separable, 419
 sequence, increasing, 425
 set of natural numbers, 339

- [**Set**] the class of sets, 362, 532
- [**SetB**] rule SetB, 556
- [**SetD**] rule SetD, 533
- [**SetDi**] rule SetDi, 535
- [**SetDm**] rule SetDm, 535
- [**SetE**] rule SetE, 553
- [**SetF**] rule SetF, 539
- [**SetIi**] rule SetIi, 541
- [**SetIm**] rule SetIm, 541
- [**SetMi**] rule SetMi, 545
- [**SetMm**] rule SetMm, 545
- [**SetN**] rule SetN, 547
 - sets, the class of all, 362
- [**SetT**] rule SetT, 554
- [**SetX**] rule SetX, 537
- [**SetXCartesianY**] rule SetXCartesianY, 568
- [**SetXListSet**] rule SetXListSet, 595
- [**SetZ**] rule SetZ, 543
- [**SetZm**] rule SetZm, 549
- [**SetZp**] rule SetZp, 551
 - seven [7], 103
 - side condition, 206
 - simple not x [$\sim x$], 335
 - simple term, 213
 - simple tuple x end [$\langle x \rangle$], 328
- [**SimpleHeadBottom**] rule SimpleHeadBottom, 619
- [**SimpleHeadNil**] rule SimpleHeadNil, 619
- [**SimpleHeadPair**] rule SimpleHeadPair, 619
- [**SimpleTailBottom**] rule SimpleTailBottom, 621
- [**SimpleTailNil**] rule SimpleTailNil, 621
- [**SimpleTailPair**] rule SimpleTailPair, 621
 - simplify a term, 213
 - six [6], 103
 - solution, minimal, 427
 - standard, IEEE floating point, 143
 - statement, 115
 - statement, instantiation, 350
 - step, proof, 185
 - step, reduction, 119
 - strict, 128
 - strict in [x], 128
- [**StrictAndX**] rule StrictAndX, 561
- [**StrictAtom**] rule StrictAtom, 565
- [**StrictBelongsToY**] rule StrictBelongsToY, 566
- [**StrictComputationalEqualX**] rule StrictComputationalEqualX, 571
- [**StrictComputationalUnequalX**] rule StrictComputationalUnequalX, 573

[StrictDivideX] rule StrictDivideX, 578
 [StrictExceptional] rule StrictExceptional, 582
 [StrictFaculty] rule StrictFaculty, 583
 [StrictIfAndOnlyIfX] rule StrictIfAndOnlyIfX, 586
 [StrictImpliedByX] rule StrictImpliedByX, 588
 [StrictImpliesX] rule StrictImpliesX, 590
 [StrictIntegerDivideX] rule StrictIntegerDivideX, 592
 [StrictMinusX] rule StrictMinusX, 597
 [StrictModuloX] rule StrictModuloX, 600
 [StrictOrX] rule StrictOrX, 603
 [StrictPair] rule StrictPair, 605
 [StrictPlusX] rule StrictPlusX, 607
 [StrictPowerXEnd] rule StrictPowerXEnd, 610
 [StrictRoundX] rule StrictRoundX, 614
 [StrictStronglyGreaterThanX] rule StrictStronglyGreaterThanX, 622
 [StrictStronglyLessThanX] rule StrictStronglyLessThanX, 624
 [StrictTimesX] rule StrictTimesX, 630
 [StrictWeaklyGreaterThanX] rule StrictWeaklyGreaterThanX, 632
 [StrictWeaklyLessThanX] rule StrictWeaklyLessThanX, 636
 strongly represent, 329
 structurally equal, 210
 [SubB] rule SubB, 556
 [SubBottom] rule SubBottom, 505
 [SubCase] rule SubCase, 506
 [SubClassical] rule SubClassical, 508
 [SubD] rule SubD, 533
 [SubDm] rule SubDm, 535
 [SubE] rule SubE, 553
 [SubEmptyList] rule SubEmptyList, 510
 [SubException] rule SubException, 511
 [SubF] rule SubF, 539
 [SubFalse] rule SubFalse, 514
 [SubFOneOfXEnd] rule SubFOneOfXEnd, 516
 [SubFThreeOfXEnd] rule SubFThreeOfXEnd, 517
 [SubFTwo] rule SubFTwo, 518
 [SubIf] rule SubIf, 519
 [SubIi] rule SubIi, 541
 [SubInfinity] rule SubInfinity, 521
 [SubLambdaXX] rule SubLambdaXX, 522
 [SubLambdaXY] rule SubLambdaXY, 522
 [SubLambdaXYFreeFor] rule SubLambdaXYFreeFor, 522
 [SubMinimumXCommaYEnd] rule SubMinimumXCommaYEnd, 523
 [SubMinusX] rule SubMinusX, 524
 [SubMm] rule SubMm, 545
 [SubN] rule SubN, 547
 [SubNilMap] rule SubNilMap, 525

- [SubNotX] rule SubNotX, 526
- [SubNumeral] rule SubNumeral, 530
- [SubPlusX] rule SubPlusX, 528
- [SubPrecisionX] rule SubPrecisionX, 529
- [SubRound] rule SubRound, 615
- [SubSet] rule SubSet, 532
 - substitute x where y is z end [$\langle x \mid y:=z \rangle$], 203, 530
 - substitution, 203
 - substitution of equals, 104, 219
- [Substitution] rule Substitution, 530
- [SubT] rule SubT, 554
 - subtree, 107
- [SubTrue] rule SubTrue, 558
 - subtype rule, 361
- [SubtypeBC] rule SubtypeBC, 556
- [SubtypeDC] rule SubtypeDC, 533
- [SubtypeDiD] rule SubtypeDiD, 535
- [SubtypeDmD] rule SubtypeDmD, 535
- [SubtypeEC] rule SubtypeEC, 553
- [SubtypeFB] rule SubtypeFB, 539
- [SubtypeFC] rule SubtypeFC, 539
- [SubtypeLiD] rule SubtypeLiD, 541
- [SubtypeImD] rule SubtypeImD, 541
- [SubtypeMiD] rule SubtypeMiD, 545
- [SubtypeMmD] rule SubtypeMmD, 545
- [SubtypeNotBC] rule SubtypeNotBC, 556
- [SubtypeNotDC] rule SubtypeNotDC, 533
- [SubtypeNotEC] rule SubtypeNotEC, 553
- [SubtypeNotFC] rule SubtypeNotFC, 539
- [SubtypeNotSetC] rule SubtypeNotSetC, 566
- [SubtypeNotTC] rule SubtypeNotTC, 554
- [SubtypeNotXC] rule SubtypeNotXC, 537
- [SubtypeNZ] rule SubtypeNZ, 547
- [SubtypeSetC] rule SubtypeSetC, 566
- [SubtypeTB] rule SubtypeTB, 554
- [SubtypeTC] rule SubtypeTC, 554
- [SubtypeXC] rule SubtypeXC, 537
- [SubtypeZD] rule SubtypeZD, 543
- [SubtypeZDi] rule SubtypeZDi, 543
- [SubtypeZmZ] rule SubtypeZmZ, 549
- [SubtypeZpN] rule SubtypeZpN, 551
- [SubtypeZpZ] rule SubtypeZpZ, 551
- [SubX] rule SubX, 537
- [SubXAndY] rule SubXAndY, 561
- [SubXApplyY] rule SubXApplyY, 563
- [SubXAtom] rule SubXAtom, 565

[SubXBelongsToY] rule SubXBelongsToY, 566
 [SubXCartesianY] rule SubXCartesianY, 568
 [SubXComputationalEqualY] rule SubXComputationalEqualY, 571
 [SubXComputationalUnequalY] rule SubXComputationalUnequalY, 573
 [SubXDivideY] rule SubXDivideY, 578
 [SubXExceptional] rule SubXExceptional, 582
 [SubXFaculty] rule SubXFaculty, 583
 [SubXHead] rule SubXHead, 585
 [SubXIfAndOnlyIfY] rule SubXIfAndOnlyIfY, 586
 [SubXImpliedByY] rule SubXImpliedByY, 588
 [SubXImpliesY] rule SubXImpliesY, 590
 [SubXIntegerDivideY] rule SubXIntegerDivideY, 592
 [SubXListSet] rule SubXListSet, 595
 [SubXMinusY] rule SubXMinusY, 598
 [SubXModuloY] rule SubXModuloY, 600
 [SubXOrY] rule SubXOrY, 603
 [SubXPair] rule SubXPair, 605
 [SubXPairY] rule SubXPairY, 606
 [SubXPlusY] rule SubXPlusY, 608
 [SubXPowerYEnd] rule SubXPowerYEnd, 610
 [SubXPredecessor] rule SubXPredecessor, 611
 [SubXSimpleHead] rule SubXSimpleHead, 619
 [SubXSimplePairY] rule SubXSimplePairY, 620
 [SubXSimpleTail] rule SubXSimpleTail, 621
 [SubXStronglyGreaterThanY] rule SubXStronglyGreaterThanY, 622
 [SubXStronglyLessThanY] rule SubXStronglyLessThanY, 624
 [SubXSubX] rule SubXSubX, 530
 [SubXSubY] rule SubXSubY, 530
 [SubXSubYNotFree] rule SubXSubYNotFree, 530
 [SubXSuccessor] rule SubXSuccessor, 626
 [SubXTail] rule SubXTail, 627
 [SubXTimesY] rule SubXTimesY, 631
 [SubXWeaklyGreaterThanY] rule SubXWeaklyGreaterThanY, 632
 [SubXX] rule SubXX, 530
 [SubXY] rule SubXY, 530
 [SubZ] rule SubZ, 543
 [SubZm] rule SubZm, 549
 [SubZp] rule SubZp, 551
 successor, 398
 suffix, 106
 suited for backchaining, 383
 supremum, 425
 syntax, 302
 syntax tree, 107, 161
 system, algebraic, 186
 system, derivation, 366

- [$\overline{\mathbf{T}}$] is true, 336
- [\mathbf{T}] true, 122, 328, 558
- [\mathbf{T}] the set of true maps, 342, 554
 - table, truth, 153, 394
 - tail of list, 174
- [TailBottom] rule TailBottom, 627
- [TailEmpty] rule TailEmpty, 627
- [TailPair] rule TailPair, 627
 - tautology, 394
- [Tautology] rule Tautology, 556
 - term, 115
 - term reduction, 159, 161
 - term reduction rule, 159, 628
 - term, binary, 321
 - term, simple, 213
 - ternary, 105
 - test, 324
 - the class of all sets, 362
 - the class of sets [\mathbf{Set}], 362, 532
 - the set of decimal fractions [\mathbf{D}], 340, 533
 - the set of decimal fractions of precision x end [\mathbf{D}_x^\cdot], 342, 535
 - the set of exceptions [\mathbf{X}], 340, 537
 - the set of false maps [\mathbf{F}], 342, 539
 - the set of infinities of precision x end [\mathbf{D}_x^∞], 342, 541
 - the set of integers [\mathbf{Z}], 342, 543
 - the set of minus infinities of precision x end [$\mathbf{D}_x^{-\infty}$], 342, 545
 - the set of natural numbers [\mathbf{N}], 339, 342, 547
 - the set of negative integers [\mathbf{Z}^-], 342, 549
 - the set of positive integers [\mathbf{Z}^+], 342, 551
 - the set of the empty list [\mathbf{E}], 344, 553
 - the set of true maps [\mathbf{T}], 342, 554
 - the set of truth values [\mathbf{B}], 340, 556
 - theorem, 186
 - theorem, main, 186
 - theory, map, 324, 443, 639
 - three [3], 103
 - [Transitivity] rule Transitivity, 580
 - transparency, referential, 109, 302
 - tree, parse, 110, 161
 - tree, syntax, 107, 161
 - true [\mathbf{T}], 122, 328, 558
 - truth is greater than falsehood, 134
 - truth table, 153, 394
 - truth value, 122
 - [TUnequalF] rule TUnequalF, 407
 - tuple x end [$\langle x \rangle$], 171, 559

Turing machine, 323
 [twice] twice, 314
 twice [twice], 314
 two [2], 103
 type function, 334
 type rule, numeral, 357
 [TypeAll] rule TypeAll, 504
 [TypeCInSet] rule TypeCInSet, 566
 [TypeDiIntegerDivideDi] rule TypeDiIntegerDivideDi, 593
 [TypeDiIntegerDivideDm] rule TypeDiIntegerDivideDm, 593
 [TypeDiModuloDi] rule TypeDiModuloDi, 600
 [TypeDiModuloDm] rule TypeDiModuloDm, 600
 [TypeDmIntegerDivideDi] rule TypeDmIntegerDivideDi, 593
 [TypeDmIntegerDivideDm] rule TypeDmIntegerDivideDm, 593
 [TypeDmIntegerDivideDn] rule TypeDmIntegerDivideDn, 593
 [TypeDmModuloDi] rule TypeDmModuloDi, 600
 [TypeDmModuloDm] rule TypeDmModuloDm, 600
 [TypeDmModuloDn] rule TypeDmModuloDn, 600
 [TypeEmptyListInE] rule TypeEmptyListInE, 510
 [TypeExists] rule TypeExists, 512
 [TypeFInB] rule TypeFInB, 514
 [TypeFInF] rule TypeFInF, 514
 [TypeIiInIi] rule TypeIiInIi, 521
 [TypeImInIm] rule TypeImInIm, 521
 [TypeMiInMi] rule TypeMiInMi, 521
 [TypeMmInMm] rule TypeMmInMm, 521
 [TypeNumeralInD] rule TypeNumeralInD, 533
 [TypeNumeralInDi] rule TypeNumeralInDi, 535
 [TypeNumeralInDm] rule TypeNumeralInDm, 535
 [TypeNumeralInN] rule TypeNumeralInN, 547
 [TypeNumeralInZ] rule TypeNumeralInZ, 543
 [TypeNumeralInZm] rule TypeNumeralInZm, 549
 [TypeNumeralInZp] rule TypeNumeralInZp, 551
 [TypeNumeralNotInB] rule TypeNumeralNotInB, 556
 [TypeNumeralNotInDi] rule TypeNumeralNotInDi, 535
 [TypeNumeralNotInDm] rule TypeNumeralNotInDm, 535
 [TypeNumeralNotInE] rule TypeNumeralNotInE, 553
 [TypeNumeralNotInF] rule TypeNumeralNotInF, 539
 [TypeNumeralNotInIi] rule TypeNumeralNotInIi, 541
 [TypeNumeralNotInIm] rule TypeNumeralNotInIm, 541
 [TypeNumeralNotInMi] rule TypeNumeralNotInMi, 545
 [TypeNumeralNotInMm] rule TypeNumeralNotInMm, 545
 [TypeNumeralNotInN] rule TypeNumeralNotInN, 547
 [TypeNumeralNotInT] rule TypeNumeralNotInT, 554
 [TypeNumeralNotInX] rule TypeNumeralNotInX, 537
 [TypeNumeralNotInZ] rule TypeNumeralNotInZ, 543

- [TypeNumeralNotInZm] rule TypeNumeralNotInZm, 549
- [TypeNumeralNotInZp] rule TypeNumeralNotInZp, 551
- [TypeTInB] rule TypeTInB, 558
- [TypeTInT] rule TypeTInT, 558
- [TypeXInX] rule TypeXInX, 511

- unary, 105
- unary minus, 112
- unary plus, 113
- unequal, computational [$x \neq y$], 124, 141, 573
- [UnequalPair] rule UnequalPair, 573
- universal property, 416
- universal quantification, 435
- up, round [$x@_+y$], 143

- value, truth, 122
- variable fixation, 377
- [**variable** x] variable x, 353, 560
- variable x [**variable** x], 353, 560
- variable, fixed, 377
- variable, fresh, 315

- weakly represent, 329
- wellfounded x end [wf(x)], 345
- [wf(x)] wellfounded x end, 345
- word, 322

- [\bar{X}] is exception, 336
- x and y [$x \wedge y$], 154, 561
- x append y [x append y], 196
- [x append y] x append y, 196
- x apply y [x' y], 304, 563
- x associates as y [$x \dot{\rightarrow} y$], 111, 564
- x atom [x atom], 169, 565
- [x atom] x atom, 169, 565
- x belongs to y [$x \in y$], 339, 566
- x cartesian y [$x \times y$], 343, 568
- x case y else z end $\left[x \left\langle \begin{array}{c} y \\ z \end{array} \right. \right]$, 327
- x colon y [$x : y$], 352, 569
- x comma y [x, y], 162, 570
- x computational equal y [$x = y$], 104, 123, 141, 571
- x computational unequal y [$x \neq y$], 124, 141, 573
- x concludes y [$x \triangleright y$], 191, 350, 575
- x deduces y [$x \rightarrow y$], 369, 576
- x defined equal y [$x \doteq y$], 104, 577

x divide y [x/y], 146, 578
 x equal y [$x \equiv y$], 104, 580
 x exceptional [$x?$], 133, 582
 x faculty [$x!$], 124, 583
 x greater priority y [$x \succ y$], 105, 584
 x head [$x \text{ head}$], 166, 585
 [$x \text{ head}$] x head, 166, 585
 [$x \text{ Head}$] x simple head, 327, 619
 x if and only if y [$x \Leftrightarrow y$], 158, 586
 x implied by y [$x \Leftarrow y$], 158, 588
 x implies y [$x \Rightarrow y$], 158, 590
 x infers y [$x \vdash y$], 349, 591
 x integer divide y [$x // y$], 131, 147, 592
 x lemma y colon z [**x lemma y:z**], 190, 594
 x listset [x^*], 344, 595
 x macro equal y [$x \doteq y$], 109, 596
 x minus y [$x - y$], 103, 144, 597
 x modulo y [$x \% y$], 131, 147, 599
 x modus ponens y [$x \supseteq y$], 375, 601
 x not in y [$x \notin y$], 566
 x optimised equal y [$x \doteq y$], 113, 602
 x or y [$x \vee y$], 156, 603
 x pair [$x \text{ pair}$], 168, 605
 [$x \text{ pair}$] x pair, 168, 605
 x pair y [$x :: y$], 165, 606
 x plus y [$x + y$], 102, 144, 607
 x power y end [x^y], 103, 148, 609
 x predecessor [x^-], 113, 398, 611
 [**x proof of y:z**] x proof of y colon z, 191, 612
 x proof of y colon z [**x proof of y:z**], 191, 612
 x reduces to y [$x \xrightarrow{+} y$], 120, 613
 x round down y [$x@_y$], 143
 x round even y [$x@_2y$], 143
 x round up y [$x@_+y$], 143
 x round y [$x@y$], 142, 614
 x round zero y [$x@_0y$], 143
 x rule y colon z [**x rule y:z**], 189, 616
 x same priority y [$x \doteq y$], 112, 617
 x select y else z end $\left[x \begin{cases} y \\ z \end{cases} \right]$, 122
 x semicolon y [$x; y$], 191, 618
 x separable y [$x \# y$], 419
 x simple and y [$x \wedge y$], 335
 x simple equal y [$x \doteq y$], 335
 x simple head [$x \text{ Head}$], 327, 619

- x simple pair $y [x \dot{=} y]$, 327, 620
- x simple tail $[x \text{ Tail}]$, 327, 621
- x simple unequal $y [x \neq y]$, 335
- x strongly greater than $y [x > y]$, 124, 141, 622
- x strongly less than $y [x < y]$, 123, 141, 624
- x successor $[x^+]$, 113, 398, 626
- x tail $[x \text{ tail}]$, 166, 627
- $[x \text{ Tail}]$ x simple tail, 327, 621
- $[x \text{ tail}]$ x tail, 166, 627
- x term reduces to $y [x \overset{\circ}{\rightarrow} y]$, 159, 628
- $[X]$ the set of exceptions, 340, 537
- x times $y [x \cdot y]$, 102, 144, 630
- x weakly greater than $y [x \geq y]$, 124, 141, 632
- x weakly less information $y [x \preceq y]$, 417, 634
- x weakly less than $y [x \leq y]$, 124, 141, 636
- $[X\text{AndStrict}]$ rule $X\text{AndStrict}$, 561
- $[X\text{BelongsToStrict}]$ rule $X\text{BelongsToStrict}$, 566
- $[X\text{ComputationalEqualStrict}]$ rule $X\text{ComputationalEqualStrict}$, 571
- $[X\text{ComputationalUnequalStrict}]$ rule $X\text{ComputationalUnequalStrict}$, 573
- $[X\text{DivideStrict}]$ rule $X\text{DivideStrict}$, 578
- $[X\text{IfAndOnlyIfStrict}]$ rule $X\text{IfAndOnlyIfStrict}$, 586
- $[X\text{ImpliedByStrict}]$ rule $X\text{ImpliedByStrict}$, 588
- $[X\text{ImpliesStrict}]$ rule $X\text{ImpliesStrict}$, 590
- $[X\text{IntegerDivideStrict}]$ rule $X\text{IntegerDivideStrict}$, 592
- $[X\text{MinusStrict}]$ rule $X\text{MinusStrict}$, 598
- $[X\text{ModuloStrict}]$ rule $X\text{ModuloStrict}$, 600
- $[X\text{OrStrict}]$ rule $X\text{OrStrict}$, 603
- $[X\text{PlusStrict}]$ rule $X\text{PlusStrict}$, 608
- $[X\text{PowerStrictEnd}]$ rule $X\text{PowerStrictEnd}$, 610
- $[X\text{RoundStrict}]$ rule $X\text{RoundStrict}$, 614
- $[X\text{StronglyGreaterThanStrict}]$ rule $X\text{StronglyGreaterThanStrict}$, 622
- $[X\text{StronglyLessThanStrict}]$ rule $X\text{StronglyLessThanStrict}$, 624
- $[X\text{TimesStrict}]$ rule $X\text{TimesStrict}$, 631
- $[X\text{WeaklyGreaterThanStrict}]$ rule $X\text{WeaklyGreaterThanStrict}$, 632
- $[X\text{WeaklyLessThanStrict}]$ rule $X\text{WeaklyLessThanStrict}$, 636
- $[X\text{WeaklyLessThanY}]$ rule $X\text{WeaklyLessThanY}$, 636
- $[Y]$ fixpoint, 317
- $[Z]$ the set of integers, 342, 543
- $[Z^-]$ the set of negative integers, 342, 549
- $[Z^+]$ the set of positive integers, 342, 551
- zero, round $[x@_0y]$, 143

Appendix D

Bibliography

- [1] K. Grue. Map theory. *Theoretical Computer Science*, 102(1):1–133, July 1992.
- [2] Donald E. Knuth. *Fundamental Algorithms*, volume 1 of *The Art of Computer Programming*. Addison-Wesley, Reading, Massachusetts, second edition, 1981.
- [3] J. McCarthy. Recursive functions of symbolic expressions and their computation by machine. *Communications of the ACM*, pages 184–195, 1960.
- [4] E. Mendelson. *Introduction to Mathematical Logic*. Wadsworth and Brooks, Reading, Massachusetts, third edition, 1987.

Appendix E

Summary of syntax

E.1 Associativity and term reduction

[$x' y' z$	\rightarrow	$(x' y)' z$]
[$x@y@z$	\rightarrow	$(x@y)@z$]
[xy^z	\rightarrow	$x(y^z)$]
[$x \cdot y \cdot z$	\rightarrow	$(x \cdot y) \cdot z$]
[$x + y + z$	\rightarrow	$(x + y) + z$]
[$x :: y :: z$	\rightarrow	$x :: (y :: z)$]
[$x \text{ append } y$]
$\text{append } z$	\rightarrow	$x \text{ append } (y \text{ append } z)$]
[x, y, z	\rightarrow	$x, (y, z)$]
[$x \wedge y \wedge z$	\rightarrow	$(x \wedge y) \wedge z$]
[$x \vee y \vee z$	\rightarrow	$(x \vee y) \vee z$]
[$x \vdash y \vdash z$	\rightarrow	$x \vdash (y \vdash z)$]
[$x \triangleright y \triangleright z$	\rightarrow	$(x \triangleright y) \triangleright z$]
[$x : y : z$	\rightarrow	$x : (y : z)$]
[$x; y; z$	\rightarrow	$(x; y); z$]
[$x < y \leq z$	\rightarrow	$(x < y) \leq z$]
[$x < y \leq z$	\rightarrow	$(x < y) \wedge (y \leq z) \succ$	
$x, y < z$	\rightarrow	$x < z \wedge y < z \succ$	
$x < y, z$	\rightarrow	$x < y \wedge x < z$]
[$\forall x, y \in S: a$	\rightarrow	$\forall x \in S: \forall y \in S: a$]
[$x \Rightarrow y \Leftrightarrow z$	\rightarrow	$(x \Rightarrow y) \Leftrightarrow z$]
[$x \Rightarrow y \Leftrightarrow z$	\rightarrow	$(x \Rightarrow y) \wedge (y \Leftrightarrow z)$]
[$x \rightarrow y \rightarrow z$	\rightarrow	$x \rightarrow (y \rightarrow z)$]
[$x \rightarrow y \equiv z$	\rightarrow	$\text{case}(x, y, T) \equiv \text{case}(x, z, T)$]
[$\langle x, y \rangle$	\rightarrow	$x :: \langle y \rangle$]

E.2 Priority

[Lx	\succ
	$x'y$	\succ
	$x@y \doteq x@_2y \doteq x@_+y \doteq x@_-y \doteq x@_0y$	\succ
	$\#x$	\succ
	$x! \doteq x^y \doteq x^+ \doteq x^- \doteq x^* \doteq$	\succ
	$x \text{ head} \doteq x \text{ tail} \doteq x \text{ pair} \doteq x \text{ atom} \doteq x \text{ Head} \doteq x \text{ Tail}$	\succ
	$x \cdot y \doteq x \cdot_+y \doteq x \cdot_-y \doteq x \cdot_0y \doteq x \cdot_2y \doteq$	\succ
	$x/y \doteq x/_+y \doteq x/_-y \doteq x/_0y \doteq x/_2y \doteq$	\succ
	$x // y \doteq x //_+y \doteq x //_-y \doteq x //_0y \doteq x //_2y \doteq$	\succ
	$x \% y \doteq x \%_+y \doteq x \%_-y \doteq x \%_0y \doteq x \%_2y$	\succ
	$+x \doteq -x$	\succ
	$x + y \doteq x ++y \doteq x +-y \doteq x +_0y \doteq x +_2y \doteq$	\succ
	$x - y \doteq x -_+y \doteq x -_-y \doteq x -_0y \doteq x -_2y$	\succ
	$x :: y \doteq x \dot{\cdot} y \doteq x \times y$	\succ
	$x \text{ append } y$	\succ
	x, y	\succ
	$x = y \doteq x \neq y \doteq x < y \doteq x \leq y \doteq x > y \doteq x \geq y \doteq x \in y \doteq x?$	\succ
	$\neg x$	\succ
	$x \wedge y$	\succ
	$x \vee y$	\succ
	$\forall x \in y: z \doteq \exists x \in y: z \doteq \varepsilon x \in y: z$	\succ
	$x \Rightarrow y \doteq x \Leftarrow y \doteq x \Leftrightarrow y$	\succ
	$x \left\{ \begin{array}{l} y \\ z \end{array} \right. \doteq x \left\langle \begin{array}{l} y \\ z \end{array} \right.$	\succ
	$\lambda x. y$	\succ
	$x \equiv y \doteq x \overset{+}{\rightarrow} y \doteq x \overset{-}{\rightarrow} y$	\succ
	$x \rightarrow y$	\succ
	$x \vdash y$	\succ
	$x \triangleright y$	\succ
	$x : y$	\succ
	$x; y$	\succ
	Mac lemma $x:y \doteq$ Mac proof of $x:y \doteq$ Mac rule $x:y$	\succ
]