

Formalizing p-adic arithmetic and dynamics in Lean (theorem prover)

Mario Weitzer

University of Leoben, Austria

Advanced Topics in Discrete Mathematics
Graz, October 11, 2024

A crash course in λ -calculus

```
Mode: 0 | Cursor: 0,0,0 | Items: 0,0,0,0
0|  
1|  
2|  
3|  
4|  
5|  
6|  
7|  
8|  
9|  
10|  
11|  
12|  
13|  
14|  
15|  
16|  
17|  
18|  
19|  
20|  
21|  
22|  
23|  
24|  
25|  
26|  
27|  
28|  
29|  
30|  
31|  
32|  
33|  
34|  
35|  
36|  
37|  
38|  
39|  
40|  
41|  
42|  
43|  
44|  
45|  
46|  
47|  
48|  
49|  
  
Lambda Editor  
Created by Mario Weitzer  
  
[G]/[Left click on index]: go to list item  
[Up]/[Down] (+ [Shift]): move up/down by one line (list item)  
[Page Up]/[Page Down]: move up/down by one page  
[Delete]: delete current list item  
[Back]: delete previous list item  
[Control] + [Left]/[Right]/[Up]: select subterm  
  
[H]: show/hide help  
[S]: toggle short notation  
[C]: toggle variable coloring  
  
[M]: set mode  
Mode 0: lambda  
Mode 1: type  
Mode 2: typed lambda  
Mode 3: typed lambda judgement  
Mode 4: typed lambda derivation  
Mode 5: type 2  
Mode 6: typed lambda 2  
Mode 7: typed lambda 2 judgement  
Mode 8: typed lambda 2 derivation  
  
Selected mode 0: lambda  
replace current (insert) list item by lambda term using...  
[1] (+ [Shift]): variable rule  
[2] (+ [Shift]): application rule  
[3] (+ [Shift]): abstraction rule  
[4] (+ [Shift]): alpha conversion at selected subterm  
[5] (+ [Shift]): beta reduction at selected subterm  
[6] (+ [Shift]): substitution at selected subterm  
[7] (+ [Shift]): selected subterm  
  
[Escape]: exit program
```

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,0 | Item: 0,0 | Nodes Count: 1,0,0,0

var. 0 | 0 **x**

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Create λ -terms by 3 rules (set of λ -terms: **A**)
• var.: *a, b, c, x, y, z, ...* (set of variables: **V**)

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,1 | Item: 1,0 | Nodes Count: 2,0,0,0

var. 0|0 x
var. 1|0 y
2|
3|
4|
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

Create λ -terms by 3 rules (set of λ -terms: A)
• var.: a, b, c, x, y, z, ... (set of variables: V)

A crash course in λ -calculus

```
Mode: 0 | Cursor: 0,0,2 | Item: 2,0 | Nodes Count: 3,0,0,0
var. 0 | 0 | x
var. 1 | 0 | y
var. 2 | 0 | z
3 |
4 |
5 |
6 |
7 |
8 |
9 |
10 |
11 |
12 |
13 |
14 |
15 |
16 |
17 |
18 |
19 |
20 |
21 |
22 |
23 |
24 |
25 |
26 |
27 |
28 |
29 |
30 |
31 |
32 |
33 |
34 |
35 |
36 |
37 |
38 |
39 |
40 |
41 |
42 |
43 |
44 |
45 |
46 |
47 |
48 |
49 |
```

Create λ -terms by 3 rules (set of λ -terms: A)
● var.: a, b, c, x, y, z, ... (set of variables: V)

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,3 | Item: 3,0 | Nodes Count: 4,0,0,0

var.
var.
var.
app. 0, 1

0 0 x
1 0 y
2 0 z
3 0 (xy)
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Create λ -terms by 3 rules (set of λ -terms: Λ)
• var.: a, b, c, x, y, z, \dots (set of variables: V)
• app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,4 | Item: 4,0 | Nodes Count: 5,0,0,0

var.
var.
var.
app. 0, 1
app. 3, 3

0 | 0 | **x**
1 | 0 | **y**
2 | 0 | **z**
3 | 0 | (**xy**)
4 | 0 | ((xy)(xy))
5 |
6 |
7 |
8 |
9 |
10 |
11 |
12 |
13 |
14 |
15 |
16 |
17 |
18 |
19 |
20 |
21 |
22 |
23 |
24 |
25 |
26 |
27 |
28 |
29 |
30 |
31 |
32 |
33 |
34 |
35 |
36 |
37 |
38 |
39 |
40 |
41 |
42 |
43 |
44 |
45 |
46 |
47 |
48 |
49 |

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,5 | Item: 5,0 | Nodes Count: 6,0,0,0

var.	0 0	x
var.	1 0	y
var.	2 0	z
app.	3 0	(xy)
app.	4 0	((xy) (xy))
abs.	5 0	(λx . ((xy) (xy)))
	6	
	7	
	8	
	9	
	10	
	11	
	12	
	13	
	14	
	15	
	16	
	17	
	18	
	19	
	20	
	21	
	22	
	23	
	24	
	25	
	26	
	27	
	28	
	29	
	30	
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	
	43	
	44	
	45	
	46	
	47	
	48	
	49	

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V, M in $\Lambda \Rightarrow (\lambda X. M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V. \Lambda)$

A crash course in λ -calculus

```
Mode: 0 | Cursor: 0,0,6 | Item: 6,0 | Nodes Count: 7,0,0,0
var. 0|0 x
var. 1|0 y
var. 2|0 z
app. 0, 1 3|0 ((xy))
app. 3, 3 4|0 (((xy)(xy)))
abs. 0, 4 5|0 ((xx.(((xy)(xy))))z)
app. 5, 2 6|0 (((xx.(((xy)(xy))))z))z
7|
8|
9|
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V, M in $\Lambda \Rightarrow (\lambda X. M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,7 | Items: 7,0 | Nodes Count: 10,0,0,0

```
var. 0|0 x
var. 1|0 y
var. 2|0 z
app. 3|0 ((xy))
app. 4|0 (((xx)((xy)(xy)))z)
abs. 5|0 ((\x.((xy)(xy)))(z))
\beta-red 6|0 (((zy)(zy)))
7|0
8|
9|
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create λ -terms by 3 rules (set of λ -terms: Λ)
• var.: a, b, c, x, y, z, \dots (set of variables: V)
• app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
• abs.: X in V , M in $\Lambda \Rightarrow (\lambda X. M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$
 β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,2 | Item: 2,0 | Nodes Count: 3,0,0,0

var. 0|0 x
var. 1|0 y
var. 2|0 z
3|
4|
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X. M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,5 | Item: 5,0 | Nodes Count: 6,0,0,0

var.	0 0	x
var.	1 0	y
var.	2 0	z
var.	3 0	a
var.	4 0	b
var.	5 0	c
	6	
	7	
	8	
	9	
	10	
	11	
	12	
	13	
	14	
	15	
	16	
	17	
	18	
	19	
	20	
	21	
	22	
	23	
	24	
	25	
	26	
	27	
	28	
	29	
	30	
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	
	43	
	44	
	45	
	46	
	47	
	48	
	49	

Create λ -terms by 3 rules (set of λ -terms: Λ)
● var.: a, b, c, x, y, z, \dots (set of variables: V)
● app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
● abs.: X in V, M in $\Lambda \Rightarrow (\lambda X. M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,6 | Item: 6,0 | Nodes Count: 7,0,0,0

var.	0 0	x
var.	1 0	y
var.	2 0	z
var.	3 0	a
var.	4 0	b
var.	5 0	c
app. 0, 1	6 0	(xy)
	7	
	8	
	9	
	10	
	11	
	12	
	13	
	14	
	15	
	16	
	17	
	18	
	19	
	20	
	21	
	22	
	23	
	24	
	25	
	26	
	27	
	28	
	29	
	30	
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	
	43	
	44	
	45	
	46	
	47	
	48	
	49	

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V, M in $\Lambda \Rightarrow (\lambda X. M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

A crash course in λ -calculus

```
Mode 0 | Cursor: 0,0,7 | Items: 7,0 | Nodes Count: 8,0,0,0
var. 0 | 0 | x
var. 1 | 0 | y
var. 2 | 0 | z
var. 3 | 0 | a
var. 4 | 0 | b
var. 5 | 0 | c
app. 0, 1 6 | 0 | ((xy)
app. 6, 2 7 | 0 | (((xy)z)
```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V, M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \xrightarrow{\beta} M[X := N]$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,8 | Item: 8,0 | Nodes Count: 9,0,0,0

var.	0 0	x
var.	1 0	y
var.	2 0	z
var.	3 0	a
var.	4 0	b
var.	5 0	c
app.	6 0	(xy)
app.	7 0	((xy)z)
abs.	8 0	(\za.((xy)z))
	9	
	10	
	11	
	12	
	13	
	14	
	15	
	16	
	17	
	18	
	19	
	20	
	21	
	22	
	23	
	24	
	25	
	26	
	27	
	28	
	29	
	30	
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	
	43	
	44	
	45	
	46	
	47	
	48	
	49	

Create λ -terms by 3 rules (set of λ -terms: Λ)
● var.: a, b, c, x, y, z, \dots (set of variables: V)
● app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
● abs.: X in V , M in $\Lambda \Rightarrow (\lambda X. M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,9 | Items: 9,0 | Nodes Count: 10,0,0,0

var.	0 0	x
var.	1 0	y
var.	2 0	z
var.	3 0	a
var.	4 0	b
var.	5 0	c
app.	0, 1	(xy)
app.	6, 2	((xy)z)
abs.	2, 7	(z.((xy)z))
abs.	1, 8	(y.(z.((xy)z)))
	10	
	11	
	12	
	13	
	14	
	15	
	16	
	17	
	18	
	19	
	20	
	21	
	22	
	23	
	24	
	25	
	26	
	27	
	28	
	29	
	30	
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	
	43	
	44	
	45	
	46	
	47	
	48	
	49	

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X. M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

A crash course in λ -calculus

```
Mode: 0 | Cursor: 0,0,10 | Item: 10,0 | Nodes Count: 11,0,0,0,0
var. 0|0 x
var. 1|0 y
var. 2|0 z
var. 3|0 a
var. 4|0 b
var. 5|0 c
app. 6|0 (xy)
app. 6|2 ((xy)z)
abs. 7|0 (xz.((xy)z))
abs. 8|0 (y.((xz.((xy)z))z))
abs. 9|0 (xz.((y.((xz.((xy)z))z))z))
abs. 0, 9 10|0 ((xz.((y.((xz.((xz.((xy)z))))z))z))z)
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- **var.**: a, b, c, x, y, z, \dots (set of variables: V)
- **app.**: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- **abs.**: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \xrightarrow{\beta} M[X := N]$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,10 | Item: 10,0 | Nodes Count: 11,0,0,0

var.	0 0	x
var.	1 0	y
var.	2 0	z
var.	3 0	a
var.	4 0	b
var.	5 0	c
app.	6 0	(xy)
app.	7 0	((xy)z)
abs.	8 0	(\z. ((xy)z))
abs.	9 0	(\y. (\z. ((xy)z)))
abs.	10 0	(\x. (\y. (\z. ((xy)z)))))
	11	
	12	
	13	
	14	
	15	
	16	
	17	
	18	
	19	
	20	
	21	
	22	
	23	
	24	
	25	
	26	
	27	
	28	
	29	
	30	
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	
	43	
	44	
	45	
	46	
	47	
	48	
	49	

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X. M)$ in Λ

$$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$$

β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy. M = (\lambda x.(\lambda y. M))$
- app. takes precedence over abs.: $\lambda x. MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,10 | Item: 10,0 | Nodes Count: 11,0,0,0

var.	0 0	x
var.	1 0	y
var.	2 0	z
var.	3 0	a
var.	4 0	b
var.	5 0	c
app.	6 0	xy
app.	7 0	xyz
abs.	8 0	$\lambda z. xyz$
abs.	9 0	$\lambda yz. xyz$
abs.	10 0	$\lambda xyz. xyz$
	11	
	12	
	13	
	14	
	15	
	16	
	17	
	18	
	19	
	20	
	21	
	22	
	23	
	24	
	25	
	26	
	27	
	28	
	29	
	30	
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	
	43	
	44	
	45	
	46	
	47	
	48	
	49	

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X. M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy. M = (\lambda x.(\lambda y. M))$
- app. takes precedence over abs.: $\lambda x. MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,11 | Item: 11,0 | Nodes Count: 12,0,0,0

```
var. 0|0 x
var. 1|0 y
var. 2|0 z
var. 3|0 a
var. 4|0 b
var. 5|0 c
app. 0,1 6|0 xy
app. 6,2 7|0 xyz
abs. 2,7 8|0 \z,xyz
abs. 1,8 9|0 \yz,xyz
abs. 0,9 10|0 \xyz,xyz
app. 10,3 11|0 (\xyz,xyz)a
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,12 | Item: 12,0 | Nodes Count: 13,0,0,0

var.	0 0	x
var.	1 0	y
var.	2 0	z
var.	3 0	a
var.	4 0	b
var.	5 0	c
app.	6 0	xy
app.	7 0	xyz
abs.	8 0	$\lambda z. xyz$
abs.	9 0	$\lambda yz. xyz$
abs.	10 0	$\lambda xyz. xyz$
app.	11 0	$(\lambda xyz. xyz) a$
app.	12 0	$(\lambda xyz. xyz) ab$
	13	
	14	
	15	
	16	
	17	
	18	
	19	
	20	
	21	
	22	
	23	
	24	
	25	
	26	
	27	
	28	
	29	
	30	
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	
	43	
	44	
	45	
	46	
	47	
	48	
	49	

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X. M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy. M = (\lambda x.(\lambda y. M))$
- app. takes precedence over abs.: $\lambda x. MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,13 | Item: 13,0 | Nodes Count: 14,0,0,0

```

var. 0|0 x
var. 1|0 y
var. 2|0 z
var. 3|0 a
var. 4|0 b
var. 5|0 c
app. 0, 1 6|0 xy
app. 6, 2 7|0 xyz
abs. 2, 7 8|0 \z,xyz
abs. 1, 8 9|0 \yz,xyz
abs. 0, 9 10|0 \xyz,xyz
app. 10, 3 11|0 ((\xyz,xyz)a
app. 11, 4 12|0 ((\xyz,xyz)ab
app. 12, 5 13|0 ((\xyz,xyz)abc
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode 0 | Cursor: 0,0,13 | Item: 13,0 | Nodes Count: 14,0,0,0
var. 0|0 x
var. 1|0 y
var. 2|0 z
var. 3|0 a
var. 4|0 b
var. 5|0 c
app. 0, 1 6|0 (xy)
app. 6, 2 7|0 ((xy)z)
abs. 2, 7 8|0 ((z.((xy)z)))
abs. 1, 8 9|0 ((y.(z.((xy)z))))
abs. 0, 9 10|0 (((x.((y.((z.((xy)z))))))a))
app. 10, 3 11|0 (((Nx.(Ny.((Nz.((xy)z))))))a))
app. 11, 4 12|0 (((Nx.(Ny.((Nz.((xy)z))))))a)b)
app. 12, 5 13|0 (((Nx.(Ny.((Nz.((xy)z))))))a)b)c)
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,13 | Item: 13,0 | Nodes Count: 14,0,0,0

```

var. 0|0 x
var. 1|0 y
var. 2|0 z
var. 3|0 a
var. 4|0 b
var. 5|0 c
app. 0, 1 6|0 xy
app. 6, 2 7|0 xyz
abs. 2, 7 8|0 \z,xyz
abs. 1, 8 9|0 \yz,xyz
abs. 0, 9 10|0 \xyz,xyz
app. 10, 3 11|0 ((\xyz,xyz)a
app. 11, 4 12|0 ((\xyz,xyz)ab
app. 12, 5 13|0 ((\xyz,xyz)abc
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,13 | Item: 13,0 | Nodes Count: 14,0,0,0

var.	0 0	x
var.	1 0	y
var.	2 0	z
var.	3 0	a
var.	4 0	b
var.	5 0	c
app.	6 0	xy
app.	7 0	xyz
abs.	8 0	$\lambda z. xyz$
abs.	9 0	$\lambda yz. xyz$
abs.	10 0	$\lambda xyz. xyz$
app.	11 0	$(\lambda xyz. xyz)a$
app.	12 0	$(\lambda xyz. xyz)ab$
app.	13 0	$(\lambda xyz. xyz)abc$
	14	
	15	
	16	
	17	
	18	
	19	
	20	
	21	
	22	
	23	
	24	
	25	
	26	
	27	
	28	
	29	
	30	
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	
	43	
	44	
	45	
	46	
	47	
	48	
	49	

Create λ -terms by 3 rules (set of λ -terms: Λ)
• var.: a, b, c, x, y, z, \dots (set of variables: V)
• app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
• abs.: X in V , M in $\Lambda \Rightarrow (\lambda X. M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
• drop outermost parenthesis: $MN = (MN)$
• app. left associative: $MNK = ((MN)K)$
• abs. right associative, only one λ : $\lambda xy. M = (\lambda x.(\lambda y. M))$
• app. takes precedence over abs.: $\lambda x. MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,13 | Item: 13,0 | Nodes Count: 14,0,0,0

var.	0 0	x
var.	1 0	y
var.	2 0	z
var.	3 0	a
var.	4 0	b
var.	5 0	c
app.	6 0	xy
app.	7 0	xyz
abs.	8 0	$\lambda z. xyz$
abs.	9 0	$\lambda yz. xyz$
abs.	10 0	$\lambda xyz. xyz$
app.	11 0	$(\lambda xyz. xyz)a$
app.	12 0	$(\lambda xyz. xyz)ab$
app.	13 0	$(\lambda xyz. xyz)abc$
	14	
	15	
	16	
	17	
	18	
	19	
	20	
	21	
	22	
	23	
	24	
	25	
	26	
	27	
	28	
	29	
	30	
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	
	43	
	44	
	45	
	46	
	47	
	48	
	49	

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X. M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy. M = (\lambda x.(\lambda y. M))$
- app. takes precedence over abs.: $\lambda x. MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,14 | Item: 14,0 | Nodes Count: 20,0,0,0

```

var. 0|0 x
var. 1|0 y
var. 2|0 z
var. 3|0 a
var. 4|0 b
var. 5|0 c
app. 6|1 xy
app. 6|2 xyz
abs. 7|2 z,xyz
abs. 8|1 yz,xyz
abs. 9|0 xyz,xyz
app. 10|0 xyz,xyz)a
app. 11|0 (xyz,xyz)ab
app. 12|5 (xyz,xyz)abc
 $\beta$ -red 13 14|0 (yz,zy)bc
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode: 0 | Cursor: 0,0,14 | Item: 14,0 | Nodes Count: 20,0,0,0,0
var. 0|0 x
var. 1|0 y
var. 2|0 z
var. 3|0 a
var. 4|0 b
var. 5|0 c
app. 6|1 xy
app. 6|2 xyz
abs. 7|2 z,xyz
abs. 8|1 yz,xyz
abs. 9|0 xyz,xyz
app. 10|0 xyz,xyz)a
app. 11|0 (xyz,xyz)ab
app. 12|5 (xyz,xyz)ab
β-red 13 14|0 (yz,jyz)b
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode: 0 | Cursor: 0,0,15 | Item: 15,0 | Nodes Count: 24,0,0,0,0
var. 0|0 x
var. 1|0 y
var. 2|0 z
var. 3|0 a
var. 4|0 b
var. 5|0 c
app. 0,1 6|0 xy
app. 6,2 7|0 xyz
abs. 2,7 8|0 \z,xyz
abs. 1,8 9|0 \yz,xyz
abs. 0,9 10|0 \xyz,xyz
app. 10,3 11|0 ((\xyz,xyz)a
app. 11,4 12|0 ((\xyz,xyz)ab
app. 12,5 13|0 ((\xyz,xyz)a)b
β-red 13 14|0 ((\yz,\yz))b
β-red 14 15|0 (\z,abz)c
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- **var.**: a, b, c, x, y, z, \dots (set of variables: V)
- **app.**: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- **abs.**: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode: 0 | Cursor: 0,0,16 | Item: 16,0 | Nodes Count: 25,0,0,0,0
var. 0|0 x
var. 1|0 y
var. 2|0 z
var. 3|0 a
var. 4|0 b
var. 5|0 c
app. 0,1 6|0 xy
app. 6,2 7|0 xyz
abs. 2,7 8|0 \z,xyz
abs. 1,8 9|0 \yz,xyz
abs. 0,9 10|0 \xyz,xyz
app. 10,3 11|0 ((\xyz,xyz)a
app. 11,4 12|0 ((\xyz,xyz)ab
app. 12,5 13|0 ((\xyz,xyz)a b
β-red 13 14|0 ((\yz,xyz)b
β-red 14 15|0 ((\z,abz)c
β-red 15 16|0 abc
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```
Mode: 0 | Cursor: 0,0,5 | Item: 5,0 | Nodes Count: 6,0,0,0
var. 0 | 0 | x
var. 1 | 0 | y
var. 2 | 0 | z
var. 3 | 0 | a
var. 4 | 0 | b
var. 5 | 0 | c
6 |
7 |
8 |
9 |
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```
Mode: 0 | Cursor: 0,0,6 | Item: 6,0 | Nodes Count: 7,0,0,0
var. 0 | 0 x
var. 1 | 0 y
var. 2 | 0 z
var. 3 | 0 a
var. 4 | 0 b
var. 5 | 0 c
app. 1, 0 6 | 0 yx
7 |
8 |
9 |
10 |
11 |
12 |
13 |
14 |
15 |
16 |
17 |
18 |
19 |
20 |
21 |
22 |
23 |
24 |
25 |
26 |
27 |
28 |
29 |
30 |
31 |
32 |
33 |
34 |
35 |
36 |
37 |
38 |
39 |
40 |
41 |
42 |
43 |
44 |
45 |
46 |
47 |
48 |
49 |
```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,7 | Items: 7,0 | Nodes Count: 8,0,0,0

var.	0 0	x
var.	1 0	y
var.	2 0	z
var.	3 0	a
var.	4 0	b
var.	5 0	c
app.	6 0	yx
abs.	7 0	$\lambda z.yx$
	8	
	9	
	10	
	11	
	12	
	13	
	14	
	15	
	16	
	17	
	18	
	19	
	20	
	21	
	22	
	23	
	24	
	25	
	26	
	27	
	28	
	29	
	30	
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	
	43	
	44	
	45	
	46	
	47	
	48	
	49	

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```
Mode: 0 | Cursor: 0,0,8 | Item: 8,0 | Nodes Count: 9,0,0,0
var. 0 | 0 x
var. 1 | 0 y
var. 2 | 0 z
var. 3 | 0 a
var. 4 | 0 b
var. 5 | 0 c
app. 6 | 0 yx
abs. 7 | 2,6
app. 8 | 0 ((\z.yx)z)
9 |
10 |
11 |
12 |
13 |
14 |
15 |
16 |
17 |
18 |
19 |
20 |
21 |
22 |
23 |
24 |
25 |
26 |
27 |
28 |
29 |
30 |
31 |
32 |
33 |
34 |
35 |
36 |
37 |
38 |
39 |
40 |
41 |
42 |
43 |
44 |
45 |
46 |
47 |
48 |
49 |
```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,9 | Items: 9,0 | Nodes Count: 10,0,0,0

var.	0 0	x
var.	1 0	y
var.	2 0	z
var.	3 0	a
var.	4 0	b
var.	5 0	c
app.	6 0	yx
abs.	7 0	$\lambda z. yx$
app.	8 0	$(\lambda z. yx) z$
abs.	9 0	$\lambda x. (\lambda z. yx) z$
	10	
	11	
	12	
	13	
	14	
	15	
	16	
	17	
	18	
	19	
	20	
	21	
	22	
	23	
	24	
	25	
	26	
	27	
	28	
	29	
	30	
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	
	43	
	44	
	45	
	46	
	47	
	48	
	49	

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X. M)$ in Λ

$$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$$

β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy. M = (\lambda x.(\lambda y. M))$
- app. takes precedence over abs.: $\lambda x. MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,10 | Item: 10,0 | Nodes Count: 11,0,0,0

var.	0 0	x
var.	1 0	y
var.	2 0	z
var.	3 0	a
var.	4 0	b
var.	5 0	c
app.	6 0	yx
abs.	7 0	$\lambda z.yx$
app.	8 0	$(\lambda z.yx)z$
abs.	9 0	$\lambda x.(\lambda z.yx)z$
app.	10 0	$(\lambda x.(\lambda z.yx))z$
	11	
	12	
	13	
	14	
	15	
	16	
	17	
	18	
	19	
	20	
	21	
	22	
	23	
	24	
	25	
	26	
	27	
	28	
	29	
	30	
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	
	43	
	44	
	45	
	46	
	47	
	48	
	49	

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode 0 | Cursor: 0,0,11 | Item: 11,0 | Nodes Count: 14,0,0,0,0
var. 0|0 x
var. 1|0 y
var. 2|0 z
var. 3|0 a
var. 4|0 b
var. 5|0 c
app. 6|0 yx
abs. 7|0 xz.yx
app. 8|0 ((xz.yx)z)
abs. 9|0 xz.((xz.yx)z)
app. 10|0 ((xz.((xz.yx)z))z)
β-red 11|0 ((xz.ya)z)
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode 0 | Cursor: 0,0,12 | Item: 12,0 | Nodes Count: 14,0,0,0,0
var. 0|0 x
var. 1|0 y
var. 2|0 z
var. 3|0 a
var. 4|0 b
var. 5|0 c
app. 6|0 yx
abs. 7|0 xz.yx
app. 8|0 ((xz.yx)z)
abs. 9|0 xz.((xz.yx)z)
app. 10|0 ((xz.((xz.yx)z))z)
β-red 11|0 ((xz.ya)z)
β-red 12|0 ya
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode 0 | Cursor: 0,0,13 | Item: 13,0 | Nodes Count: 14,0,0,0
var. 0|0 x
var. 1|0 y
var. 2|0 z
var. 3|0 a
var. 4|0 b
var. 5|0 c
app. 6|0 yx
abs. 7|0 xz.yx
app. 8|0 (\xz.yx)z
abs. 9|0 xz.(\xz.yx)z
app. 10|0 (xz.(\xz.yx)z)z
β-red 11|0 (\xz.ya)z
β-red 12|0 ya
copy 13|0 (\xz.(\xz.yx)z)z
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- **var.**: a, b, c, x, y, z, \dots (set of variables: V)
- **app.**: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- **abs.**: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode 0 | Cursor: 0,0,13 | Item: 13,0 | Nodes Count: 14,0,0,0
var. 0|0 x
var. 1|0 y
var. 2|0 z
var. 3|0 a
var. 4|0 b
var. 5|0 c
app. 6|0 yx
abs. 7|0 xz.yx
app. 8|0 (\xz.yx)z
abs. 9|0 xz.(\xz.yx)z
app. 10|0 (xz.(\xz.yx)z)z
β-red 11|0 (\xz.ya)z
β-red 12|0 ya
copy 13|0 xz.(\xz.yx)z
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- **var.**: a, b, c, x, y, z, \dots (set of variables: V)
- **app.**: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- **abs.**: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,13 | Item: 13,0 | Nodes Count: 14,0,0,0

```

var. 0|0 x
var. 1|0 y
var. 2|0 z
var. 3|0 a
var. 4|0 b
var. 5|0 c
app. 6|0 yx
abs. 7|0 \z.yx
app. 8|0 (\z.yx)z
abs. 9|0 \x.(\z.yx)z
app. 10|0 (\x.(\z.yx)z)a
\beta-red 11|0 (\z.ya)z
copy 12|0 ya
13|0 \z.(\z.yx)z
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode 0 | Cursor: 0,0,14 | Item: 14,0 | Nodes Count: 16,0,0,0,0
var. 0|0 x
var. 1|0 y
var. 2|0 z
var. 3|0 a
var. 4|0 b
var. 5|0 c
app. 6|0 yx
abs. 7|0 xz.yx
app. 8|0 (\xz.yx)z
abs. 9|0 xz.(\xz.yx)z
app. 10|0 (xz.(\xz.yx)z)a
β-red 11|0 (\xz.ya)z
β-red 12|0 ya
copy 13|0 (\xz.yx)z
β-red 14|0 (\xz.yx)a
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- **var.**: a, b, c, x, y, z, \dots (set of variables: V)
- **app.**: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- **abs.**: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode: 0 | Cursor: 0,0,15 | Item: 15,0 | Nodes Count: 17,0,0,0,0
var. 0|0 x
var. 1|0 y
var. 2|0 z
var. 3|0 a
var. 4|0 b
var. 5|0 c
app. 1,0
abs. 2,6
app. 7,2
abs. 0,8
app. 9,3
β-red 10
β-red 11
copy 10
β-red 13
β-red 14
15|0 ya
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- **var.**: a, b, c, x, y, z, \dots (set of variables: V)
- **app.**: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- **abs.**: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode: 0 | Cursor: 0,0,24 | Item: 21,0 | Nodes Count: 139,0,0,0
Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
21]
22]
23]
24]
25]
26]
27]
28]
29]
30]
31]
32]
33]
34]
35]
36]
37]
38]
39]
40]
41]
42]
43]
44]
45]
46]

0|0 \y.(\nx.y(\nx))(\nx.y(\nx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfix)
9|0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)
10|0 \mfix.m(fix)
11|0 \m.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12|0 \mfix.m(fix)
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \m.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15|0 \m.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16|0 (\sxy.y(\nx))(\sxy(\nx))(\sfnm.(\suv.suv))((\m.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
17|0 (\sxy.y(\nx))(\sxy(\nx))(\sfnm.(\suv.suv))((\m.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
f((\m.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m)mn)n)m

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \cup (\Lambda\Lambda) \cup (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode: 0 | Cursor: 0,0,24 | Item: 21,0 | Nodes Count: 140,0,0,0
Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
succ zero
0|0 \y.(\nx.y(\nx))(\nx.y(\nx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfix)
9|0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)
10|0 \mfix.m(mfix)
11|0 \m.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12|0 \mfix.m(nfix)
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \m.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15|0 \m.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16|0 (\sxy.y(\nx))(\sxy(\nx))(\sfnm.(\suv.suv))((\m.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
17|0 (\sxy.y(\nx))(\sxy(\nx))(\sfnm.(\suv.suv))((\m.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
18|0 \fix.xfs
19|0 \x.x(\sxy.x)
20|0 \sxy.x(\sxy.y)
21|0 (\mfix.f(mfix))(\fix.x)
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \cup (\Lambda\Lambda) \cup (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
succ zero,  $\beta x_1$ 
  0 |  $\lambda y. (\lambda x. y (xx)) (\lambda x. y (xx))$ 
  1 |  $\lambda xy. x$ 
  2 |  $\lambda xy. y$ 
  3 |  $\lambda z. z (\lambda xy. y) (\lambda xy. x)$ 
  4 |  $\lambda xy. x y x$ 
  5 |  $\lambda xy. x x y$ 
  6 |  $\lambda uv. u v v$ 
  7 |  $\lambda f x. f x$ 
  8 |  $\lambda mfx. f (mfx)$ 
  9 |  $\lambda mfx. m (\lambda gh. h (gf)) (\lambda u. x) (\lambda u. u)$ 
 10 |  $\lambda mfx. m (mfix)$ 
 11 |  $\lambda mn. n (\lambda mfx. m (\lambda gh. h (gf)) (\lambda u. x) (\lambda u. u)) m$ 
 12 |  $\lambda mn. n (mfx) x$ 
 13 |  $\lambda z. z (\lambda xy. y) (\lambda xy. x)$ 
 14 |  $\lambda mn. n (\lambda mfx. m (\lambda gh. h (gf)) (\lambda u. x) (\lambda u. u)) m$ 
 15 |  $\lambda mn. n (\lambda mfx. m (\lambda gh. h (gf)) (\lambda u. x) (\lambda u. u)) m$ 
    |  $((n (\lambda mfx. m (\lambda gh. h (gi)))) (\lambda u. x) (\lambda u. u)) m$ 
 16 |  $(\lambda y. (\lambda x. y (xx)) (\lambda x. y (xx))) (\lambda mn. (\lambda uv. u v v) ((\lambda mfx. m (mfx)) ((\lambda mfx. f (mfx)) (\lambda f x. f x)))$ 
 17 |  $(\lambda y. (\lambda x. y (xx)) (\lambda x. y (xx))) (\lambda mn. (\lambda uv. u v v) f ((\lambda mn. (\lambda mfx. m (\lambda gh. h (gf)))) (\lambda u. x) (\lambda u. u)) m))$ 
 18 |  $\lambda tsx. x is$ 
 19 |  $\lambda x. x (\lambda xy. x)$ 
 20 |  $\lambda x. x (\lambda xy. y)$ 
 21 |  $\lambda fx. f ((\lambda fx. x) fx)$ 
 22 |
 23 |
 24 |
 25 |
 26 |
 27 |
 28 |
 29 |
 30 |
 31 |
 32 |
 33 |
 34 |
 35 |
 36 |
 37 |
 38 |
 39 |
 40 |
 41 |
 42 |
 43 |
 44 |
 45 |
 46 |

```

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in A \Rightarrow (MN) \text{ in } A$
 • abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \text{ in } A$

$\Lambda = V \upharpoonright A \upharpoonright (\lambda V. \Lambda)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = ((MN))$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy. M = ((\lambda x. (\lambda y. M)))$
 • app. takes precedence over abs.: $\lambda x. MN = ((\lambda x. M)N)$

A crash course in λ -calculus

```

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
succ zero,  $\beta\lambda 2$ 
0 | \y.(\lambda x.y(xx))(\lambda x.y(xx))
1 | \xy.x
2 | \xy.y
3 | \z.z(\lambda xy.y)(\lambda xy.x)
4 | \xy.yxy
5 | \xy.xxy
6 | \xuv.xuv
7 | \fx.x
8 | \mfix.f(mfix)
9 | \mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)
10 | \mfix.m(mfix)
11 | \mn.n(\mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m
12 | \mfix.m(mfix)x
13 | \z.z(\lambda xxy.y)(\lambda xy.x)
14 | \mn.n(\mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n
15 | \mn.n(\mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m
    ((n(\mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n
    ((\lambda y.(\lambda x.y(xx))(\lambda x.y(xx)))(\lambda mnn.(\lambda xuv.
    (\lambda nnn.(\lambda mfix.m(mfix))(\lambda mfix.f(mfix))(\lambda fx.x)))
    (\lambda y.(\lambda x.y(xx))(\lambda x.y(xx)))(\lambda mnn.(\lambda xuv.
    f((\lambda mn.n(\mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m
    18 | \lfix.x1s
19 | \x.x.(\lambda xy.x)
20 | \x.x.(\lambda xy.y)
21 | \lfix.f((\lambda x.x)x)
22 |
23 |
24 |
25 |
26 |
27 |
28 |
29 |
30 |
31 |
32 |
33 |
34 |
35 |
36 |
37 |
38 |
39 |
40 |
41 |
42 |
43 |
44 |
45 |
46 |

```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
 • abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X. M) \in \Lambda$

$\Lambda = V \uplus (\Lambda \uplus (\lambda V. \Lambda))$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = ((M)N)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
- app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (M))N$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
succ zero, $\beta \times 3$

```

0|0 \y.(\nx.y(\xx))(\nx.y(\xx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfix)
9|0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)
10|0 \mfix.m(fix)
11|0 \mfix.m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12|0 \mfix.m(fix)
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15|0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(\m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16|0 (\sxy.y(\xx))(\sxy.x(\xx))(\m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))n(\sxy.y)(\sxy.x)
17|0 (\sxy.y(\xx))(\sxy.x(\xx))(\m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))n(\sxy.y)(\sxy.x)
18|0 \fix.xls
19|0 \sxy.x(\sxy.x)
20|0 \sxy.x(\sxy.y)
21|0 \fix.fx
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one

```
0|0 \y.(\nx.y(\xx))(\nx.y(\xx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfix)
9|0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)
10|0 \mfix.m(fix)
11|0 \m.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12|0 \mfix.m(fix)
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \m.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15|0 \m.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16|0 (\sxy.y(\xx))(\sxy(\xx))(\sfnm.(\suv.suv))((\m.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
17|0 (\sxy.y(\xx))(\sxy(\xx))(\sfnm.(\suv.suv))((\m.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
18|0 \fix.xls
19|0 \x.x(\sxy.x)
20|0 \x.x(\sxy.y)
21|0 \fix.fx
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode: 0 | Cursor: 0,0,25 | Item: 22,0 | Nodes Count: 143,0,0,0
Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
succ one
0|0 \y.(\nx.y(\nx))(\nx.y(\nx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfix)
9|0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)
10|0 \mfix.m(fix)
11|0 \mfix.m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12|0 \mfix.m(nfix)
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15|0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16|0 (\sy.(\sx.y(\sx))(\sx.y(\sx)))(\sfm.(\suv.suv))((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
17|0 (\sy.(\sx.y(\sx))(\sx.y(\sx)))(\sfm.(\suv.suv))((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
18|0 \fix.xfs
19|0 \sx.x(\sxy.x)
20|0 \sx.x(\sxy.y)
21|0 \fix.fx
22|0 (\mfix.f(mfix))(\fix.fx)
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
succ one, $\beta x 1$

```

0|0 \y.(\nx.y(\nx))(\nx.y(\nx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfx)
9|0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)
10|0 \mfix.m(fix)
11|0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12|0 \mfix.m(nfx)
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15|0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16|0 (\sxy.y(\nx))(\sxy(\nx))((\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))n((\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))
17|0 (\sxy.y(\nx))(\sxy(\nx))((\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))n((\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm((\mfix.m(nfx))((\mfix.f(mfx))(\fx.x))((f((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm))(\fx.x)))
18|0 \fx.xfs
19|0 \x.x(\sxy.x)
20|0 \sxy.x(\sxy.y)
21|0 \fx.fx
22|0 \fx.f((\fx.fx)fx)
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
succ one,  $\beta\lambda x$ 
  0 | 0  \text{y}.(\text{nx.y}(\text{xx}))(\text{nx.y}(\text{xx}))
  1 | 0  \text{xy}.x
  2 | 0  \text{xy}.y
  3 | 0  \text{z}.z(\text{xy.y})(\text{xy.x})
  4 | 0  \text{xy}.xyx
  5 | 0  \text{xy}.xxy
  6 | 0  \text{xyuv}.xuv
  7 | 0  \text{nf}.x
  8 | 0  \text{nmfix}.f(\text{mfx})
  9 | 0  \text{nmfix}.m(\text{gh}.h(\text{gf}))(\text{nu}.x)(\text{nu}.u)
 10 | 0  \text{nmn}.m(\text{nmfix}.m(\text{gh}.h(\text{gf})))(\text{nu}.x)(\text{nu}.u))m
 11 | 0  \text{nmn}.n(\text{nmfix}.m(\text{gh}.h(\text{gf})))(\text{nu}.x)(\text{nu}.u))m
 12 | 0  \text{nmfix}.m(\text{nf}).x
 13 | 0  \text{z}.z(\text{nxxy.y})(\text{ny.x})
 14 | 0  \text{nmn}.n(\text{nmfix}.m(\text{gh}.h(\text{gf})))(\text{nu}.x)(\text{nu}.u))m
 15 | 0  \text{nmn}.n(\text{nmfix}.m(\text{gh}.h(\text{gf})))(\text{nu}.x)(\text{nu}.u))m
    ))(n(\text{nmfix}.m(\text{gh}.h(\text{gf})))(\text{nu}.x)(\text{nu}.u))m
 16 | 0  (\text{ny}.(\text{nx.y}(\text{xx}))) (\text{nx.y}(\text{xx}))) (\text{nmn}.(\text{nxuv}.
  (\text{nmnfix}.m(\text{nf})) ((\text{nmix}.f(\text{mfx})) (\text{nx}.x)))
 17 | 0  (\text{ny}.(\text{nx.y}(\text{xx}))) (\text{nx.y}(\text{xx}))) (\text{nmn}.(\text{nxuv}.
  f((\text{nmn}.n(\text{nmfix}.m(\text{gh}.h(\text{gf})))(\text{nu}.x))(\text{nu}.
  18 | 0  \text{nts}.x
 19 | 0  \text{nx}.x(\text{nx}.x)
 20 | 0  \text{nx}.x(\text{nx}.y)
 21 | 0  \text{nf}.fx
 22 | 0  \text{fx}.f((\text{nx}.fx)x)
 23 |
 24 |
 25 |
 26 |
 27 |
 28 |
 29 |
 30 |
 31 |
 32 |
 33 |
 34 |
 35 |
 36 |
 37 |
 38 |
 39 |
 40 |
 41 |
 42 |
 43 |
 44 |
 45 |
 46 |

```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
 • abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X. M) \in \Lambda$

$\Lambda = V \uplus (\Lambda \uplus (\lambda V. \Lambda))$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = ((M)N)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
- app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (M))N$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
succ one, $\beta x 3$

```

0| 0 \y.(\nx.y(\nx))(\nx.y(\nx))
1| 0 \xy.x
2| 0 \xy.y
3| 0 \z.z(\sxy.y)(\sxy.x)
4| 0 \sxy.sxy
5| 0 \sxy.sxy
6| 0 \suv.suv
7| 0 \fx.x
8| 0 \mfix.f(mfx)
9| 0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10| 0 \mfix.m(f(nix))
11| 0 \mfix.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12| 0 \mfix.m(nfx)
13| 0 \z.z(\sxy.y)(\sxy.x)
14| 0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15| 0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16| 0 (\sy.(\sx.y(\sx))(\sx.y(\sx)))(\sfm.(\suv.suv))((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
17| 0 (\sy.(\sx.y(\sx))(\sx.y(\sx)))(\sfm.(\suv.suv))((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
18| 0 \fix.xfs
19| 0 \sx.x(\sxy.x)
20| 0 \sx.x(\sxy.y)
21| 0 \fix.fx
22| 0 \fix.f(fx)
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode: 0 | Cursor: 0,0,25 | Item: 22,0 | Nodes Count: 146,0,0,0
Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
0|0 \y.(\nx.y(\nx))(\nx.y(\nx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\nx.y)(\nx.x)
4|0 \xy.xyx
5|0 \xy.xxy
6|0 \xuv.xuv
7|0 \fx.x
8|0 \mfix.f(mfx)
9|0 \mfix.m(\gh.h(gf))(\nuu.x)(\nuu.u)m
10|0 \mfix.m(fix)
11|0 \mfix.m(\mfix.m(\gh.h(gf))(\nuu.x)(\nuu.u))m
12|0 \mfix.m(fix)
13|0 \z.z(\nxxy.y)(\nx.y)
14|0 \mn.n(\mfix.m(\gh.h(gf))(\nuu.x)(\nuu.u))m(\nxxy.y)(\nx.y)
15|0 \mn.n(\mfix.m(\gh.h(gf))(\nuu.x)(\nuu.u))m(\nxxy.y)(\nx.y)(m(\mfix.m(\gh.h(gf))(\nuu.x)(\nuu.u))n(\nxxy.y)(\nx.y))
16|0 (\ny.(\nx.y(\nx))(\nx.y(\nx)))(\mfix.(\nxuv.xuv))((\mn.n(\mfix.m(\gh.h(gf))(\nuu.x)(\nuu.u))m(\nxxy.y)(\nx.y))nm)
17|0 (\ny.(\nx.y(\nx))(\nx.y(\nx)))((\mfix.(\nxuv.xuv))((\mn.n(\mfix.m(\gh.h(gf))(\nuu.x)(\nuu.u))m(\nxxy.y)(\nx.y))nm)
18|0 \fix.xfs
19|0 \x.x(\nx.y)
20|0 \x.x(\nx.y)
21|0 \fix.fx
22|0 \fix.f(fx)
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$
 β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$
 Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
succ two

```
0|0 \y.(\nx.y(\xx))(\nx.y(\xx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfx)
9|0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10|0 \mfix.m(mfix)
11|0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
12|0 \mfix.m(nfix)
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15|0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16|0 (\sxy.y(\xx))(\sxy.y(\xx))(\m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))m
17|0 (\sxy.y(\xx))(\sxy.y(\xx))(\m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))m
18|0 \fix.xfs
19|0 \x.x(\sxy.x)
20|0 \x.x(\sxy.y)
21|0 \fix.fx
22|0 \fix.f(fx)
23|0 (\mfix.f(mfx))(\fix.f(fx))
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Y-combinator
true
false
not
and
or
ife
zero
succ
pred
add
sub
mul
iszero
leg
eq
div
mod
pair
fst
sec
one
two
succ two,  $\beta x_1$ 
  0 | 0  $\lambda y. (\lambda x. y(xx))(\lambda x. y(xx))$ 
  1 | 0  $\lambda xy. x$ 
  2 | 0  $\lambda xy. y$ 
  3 | 0  $\lambda z. z(\lambda xy. y)(\lambda xy. x)$ 
  4 | 0  $\lambda xy. xy$ 
  5 | 0  $\lambda xy. xxy$ 
  6 | 0  $\lambda xuv. xuv$ 
  7 | 0  $\lambda ffx. x$ 
  8 | 0  $\lambda mfix. f(mfix)$ 
  9 | 0  $\lambda mfix. m(\lambda gh. h(gf))(\lambda u. x)(\lambda u. u)$ 
 10 | 0  $\lambda mnfix. m(mfix)$ 
 11 | 0  $\lambda mn. n(\lambda mfix. m(\lambda gh. h(gf))(\lambda u. x)(\lambda u. u))m$ 
 12 | 0  $\lambda mnfix. m(mfix)x$ 
 13 | 0  $\lambda z. z(\lambda xxy. y)(\lambda xy. x)$ 
 14 | 0  $\lambda mn. n(\lambda fix. m(\lambda gh. h(gf))(\lambda u. x)(\lambda u. u))m$ 
 15 | 0  $\lambda mn. n(\lambda fix. m(\lambda gh. h(gf))(\lambda u. x)(\lambda u. u))m$ 
 16 | 0  $(\lambda y. (\lambda x. y(xx))(\lambda x. y(xx)))(\lambda mn. (\lambda xuv.$ 
 17 | 0  $(\lambda y. (\lambda x. y(xx))(\lambda x. y(xx))))(\lambda mn. (\lambda xuv.$ 
 $f((\lambda mn. n(\lambda mfix. m(\lambda gh. h(gf))(\lambda u. x)(\lambda u. u))m$ 
 18 | 0  $\lambda fix. xis$ 
 19 | 0  $\lambda x. x(\lambda xxy. x)$ 
 20 | 0  $\lambda x. x(\lambda xxy. y)$ 
 21 | 0  $\lambda fix. fix$ 
 22 | 0  $\lambda fix. f(fx)$ 
 23 | 0  $\lambda fix. f((\lambda fix. f(fx))fx)$ 
 24 |
 25 |
 26 |
 27 |
 28 |
 29 |
 30 |
 31 |
 32 |
 33 |
 34 |
 35 |
 36 |
 37 |
 38 |
 39 |
 40 |
 41 |
 42 |
 43 |
 44 |
 45 |
 46 |

```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
 • abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X. M) \in \Lambda$
 $\Lambda = V \uplus (\Lambda \uplus (\lambda V. \Lambda))$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariable functions)
 • drop outermost parenthesis: $MN = ((MN))$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy. M = ((\lambda x.(\lambda y. M)))$
 • app. takes precedence over abs.: $\lambda xy. MN = (\lambda x. (y. M))N$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
succ two, $\beta x 2$

```

0|0 \y.(\nx.y(\xx))(\nx.y(\xx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfx)
9|0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10|0 \mfix.m(mfix)(nfix)
11|0 \mfix.m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12|0 \mfix.m(nfix)
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15|0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16|0 (\sxy.y(\xx))(\sxy(\xx))(\mfix.(\suv.suv))((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
17|0 (\sxy.y(\xx))(\sxy(\xx))(\mfix.(\suv.suv))((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
18|0 \fix.xfs
19|0 \x.x(\sxy.x)
20|0 \x.x(\sxy.y)
21|0 \fix.fx
22|0 \fix.f(fx)
23|0 \fix.f((\nx.f(fx))x)
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
succ two, $\beta x 3$

```

0| 0 \y.(\nx.y(\nx))(\nx.y(\nx))
1| 0 \xy.x
2| 0 \xy.y
3| 0 \z.z(\sxy.y)(\sxy.x)
4| 0 \sxy.sxy
5| 0 \sxy.sxy
6| 0 \suv.suv
7| 0 \fx.x
8| 0 \mfix.f(mfx)
9| 0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10| 0 \mfix.m(mfix.n(nix))
11| 0 \mfix.m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12| 0 \mfix.m(nfx)
13| 0 \z.z(\sxy.y)(\sxy.x)
14| 0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15| 0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16| 0 (\sxy.y(\sxy.y))(\sxy.y(\sxy.y))(\sfix.(\suv.suv)((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm))
17| 0 (\sxy.y(\sxy.y))(\sxy.y(\sxy.y))(\sfix.(\suv.suv)((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm))
18| 0 \fix.xfs
19| 0 \x.x(\sxy.x)
20| 0 \x.x(\sxy.y)
21| 0 \fix.fx
22| 0 \fix.f(fx)
23| 0 \fix.f(f(fx))
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three

```
0| 0 \y.(\nx.y(\nx))(\nx.y(\nx))
1| 0 \xy.x
2| 0 \xy.y
3| 0 \z.z(\sxy.y)(\sxy.x)
4| 0 \sxy.sxy
5| 0 \sxy.sxy
6| 0 \suv.suv
7| 0 \fx.x
8| 0 \mfix.f(mfx)
9| 0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10| 0 \mfix.m(mfix)(nfix)
11| 0 \mfix.m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12| 0 \mfix.m(nfix)
13| 0 \z.z(\sxy.y)(\sxy.x)
14| 0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15| 0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16| 0 (\sxy.x.y(\sxy.y))(\sxy.y(\sxy.x))(\sfnm.(\suv.suv)((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)(\sfnx.m(nfix))((\mfix.f(mfx))(\sfx.x))(\f((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)(\sfx.x))
17| 0 (\sxy.x.y(\sxy.y))(\sxy.y(\sxy.x))(\sfnm.(\suv.suv)((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)(\f((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)(\sfx.x)))
18| 0 \fix.xfs
19| 0 \sxy.x(\sxy.x)
20| 0 \sxy.x(\sxy.y)
21| 0 \fix.fx
22| 0 \fix.f(fx)
23| 0 \fix.f(f(fx))
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$$\Lambda = V \mid (\Lambda A) \mid (\lambda V.\Lambda)$$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
add two
24|0
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in A \Rightarrow (MN) \in A$
 • abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \in A$

$\Lambda = V \uplus A \uplus (\lambda V. A)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = ((MN))$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
 • app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

```

Mode: 0 | Cursor: 0,0,27 | Item: 24,0 | Nodes Count: 153,0,0,0
Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
add two three
0|0 \y.(\nx.y(\nx))(\nx.y(\nx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfx)
9|0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10|0 \mfix.m(mfix)
11|0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12|0 \mfix.m(nfix)
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15|0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16|0 (\sxy.y(\nx))(\sxy(\nx))(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
17|0 (\sxy.y(\nx))(\sxy(\nx))(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)nm
18|0 \fx.xfs
19|0 \nx.x(\sxy.x)
20|0 \sxy.x(\sxy.y)
21|0 \fx.fx
22|0 \fx.f(fx)
23|0 \fx.f(f(fx))
24|0 (\mn.mfix.m(nfix))(\fx.f(fx))(\fx.f(f(fx)))
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
add two three, $\beta x 1$

```

0| 0 `y.(\$x.y(\$x))(\$x.y(\$x))
1| 0 \$xy.x
2| 0 \$xy.y
3| 0 \$z.z(\$xy.y)(\$xy.x)
4| 0 \$xy.xyx
5| 0 \$xy.xxy
6| 0 \$xuv.xuv
7| 0 \$fx.x
8| 0 \$mfix.f(mfx)
9| 0 \$mfix.m(\$gh.h(gf))(\$u.x)(\$u.u)m
10| 0 \$mfix.m(mfix)
11| 0 \$m.n(\$mfix.m(\$gh.h(gf))(\$u.x)(\$u.u))m
12| 0 \$mfix.m(nfix)
13| 0 \$z.z(\$xxy.y)(\$xy.x)
14| 0 \$m.n(\$mfix.m(\$gh.h(gf))(\$u.x)(\$u.u))m(\$xxy.y)(\$xy.x)
15| 0 \$m.n(\$mfix.m(\$gh.h(gf))(\$u.x)(\$u.u))m(\$xxy.y)(\$xy.x)(m(\$mfix.m(\$gh.h(gf))(\$u.x)(\$u.u))n(\$xxy.y)(\$xy.x))
16| 0 (\$y.(\$x.y(\$x))(\$x.y(\$x))) (\$fmn.(\$xuv.xuv)) (\$mn.n(\$mfix.m(\$gh.h(gf))(\$u.x)(\$u.u))m(\$xxy.y)(\$xy.x))nm
17| 0 (\$y.(\$x.y(\$x))(\$x.y(\$x))) (\$fmn.(\$xuv.xuv)) (\$mn.n(\$mfix.m(\$gh.h(gf))(\$u.x)(\$u.u))m(\$xxy.y)(\$xy.x))nm
18| 0 \$fx.xfs
19| 0 \$x.x(\$xy.x)
20| 0 \$x.x(\$xy.y)
21| 0 \$fx.fx
22| 0 \$fx.f(fx)
23| 0 \$fx.f(f(fx))
24| 0 (\$mfix.(\$fx.f(fx))f(nfx))(\$fx.f(f(fx)))
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: $\textcolor{blue}{\Lambda}$)
 • var.: a, b, c, x, y, z, \dots (set of variables: $\textcolor{blue}{V}$)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = (MN)$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
 • app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode: 0 | Cursor: 0,0,27 | Item: 24,0 | Nodes Count: 157,0,0,0
Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
add two three,  $\beta x 2$ 
0| 0 \y.(\nx.y(\nx))(\nx.y(\nx))
1| 0 \xy.x
2| 0 \xy.y
3| 0 \z.z(\sxy.y)(\sxy.x)
4| 0 \sxy.sxy
5| 0 \sxy.sxy
6| 0 \suv.suv
7| 0 \fx.x
8| 0 \mfix.f(mfx)
9| 0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10| 0 \mfix.m(mfix)
11| 0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12| 0 \mfix.m(nfix)
13| 0 \z.z(\sxy.y)(\sxy.x)
14| 0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15| 0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16| 0 (\sxy.y(\sxy.y))(\sxy.y(\sxy.y))(\sfix.(\suv.suv))((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
17| 0 (\sxy.y(\sxy.y))(\sxy.y(\sxy.y))(\sfix.(\suv.suv))((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
18| 0 \fx.xs
19| 0 \x.x(\sxy.x)
20| 0 \x.x(\sxy.y)
21| 0 \fx.fx
22| 0 \fx.f(fx)
23| 0 \fx.f(f(fx))
24| 0 \fx.(\fx.f(fx))f(((\fx.f(f(fx))))fx)
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda \Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode: 0 | Cursor: 0,0,27 | Item: 24,0 | Nodes Count: 159,0,0,0
Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
add two three,  $\beta x 3$ 
0| 0 \y.(\nx.y(\nx))(\nx.y(\nx))
1| 0 \xy.x
2| 0 \xy.y
3| 0 \z.z(\nx.y)(\nx.x)
4| 0 \xy.xyx
5| 0 \xy.xxy
6| 0 \xuv.xuv
7| 0 \fx.x
8| 0 \mfix.f(mfx)
9| 0 \mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u)
10| 0 \mfix.m(mfix)
11| 0 \mn.n(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m
12| 0 \mfix.m(nfix)
13| 0 \z.z(\nxxy.y)(\nx.y)
14| 0 \mn.n(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\nxxy.y)(\nx.y)
15| 0 \mn.n(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\nxxy.y)(\nx.y)(m(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))n(\nxxy.y)(\nx.y))
16| 0 (\ny.(\nx.y(\nx))(\nx.y(\nx)))(\mfix.(\nxuv.xuv))((\mn.n(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\nxxy.y)(\nx.y))nm)
17| 0 (\ny.(\nx.y(\nx))(\nx.y(\nx)))(\mfix.(\nxuv.xuv))((\mn.n(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\nxxy.y)(\nx.y))nm)
18| 0 \fix.xfs
19| 0 \nx.x(\nx.y.x)
20| 0 \nx.x(\nx.y.y)
21| 0 \fix.fx
22| 0 \fix.f(fx)
23| 0 \fix.f(f(fx))
24| 0 \fix.(\nx.f(fx))((\fix.f(f(fx)))fx)
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda M) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode: 0 | Cursor: 0,0,27 | Item: 24,0 | Nodes Count: 157,0,0,0
Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
add two three,  $\beta x 4$ 
0| 0 \y.(\nx.y(\nx))(\nx.y(\nx))
1| 0 \xy.x
2| 0 \xy.y
3| 0 \z.z(\sxy.y)(\sxy.x)
4| 0 \sxy.sxy
5| 0 \sxy.sxy
6| 0 \suv.suv
7| 0 \fx.x
8| 0 \mfix.f(mfx)
9| 0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10| 0 \mfix.m(mfix)
11| 0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12| 0 \mfix.m(nfix)
13| 0 \z.z(\sxy.y)(\sxy.x)
14| 0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15| 0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16| 0 (\sxy.y(\sxy.y))(\sxy.y(\sxy.y))(\sfix.(\suv.suv))((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
17| 0 (\sxy.y(\sxy.y))(\sxy.y(\sxy.y))(\sfix.(\suv.suv))((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
18| 0 \fx.xfs
19| 0 \x.x(\sxy.x)
20| 0 \x.x(\sxy.y)
21| 0 \fx.fx
22| 0 \fx.f(fx)
23| 0 \fx.f(f(fx))
24| 0 \fx.f(f((\fx.f(f(fx))))fx))
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: $\textcolor{blue}{A}$)

- var.: a, b, c, x, y, z, \dots (set of variables: $\textcolor{blue}{V}$)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda A) \mid (\lambda V.\Lambda)$
 β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$
Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
add two three, $\beta x 5$

```

0| 0 `y.(`x.y(xx))(xy.y(xx))
1| 0 `xy.x
2| 0 `xy.y
3| 0 `z.z(`xy.y)(xy.x)
4| 0 `xy.xyx
5| 0 `xy.xxy
6| 0 `xuv.xuv
7| 0 `fx.x
8| 0 `mixx.f(mfx)
9| 0 `mixx.m(`gh.h(gf))(`u.x)(`u.u)m
10| 0 `mixx.m(mix)
11| 0 `mn.n(`mixx.m(`gh.h(gf))(`u.x)(`u.u))m
12| 0 `mixx.m(nfx)
13| 0 `z.z(`xy.y)(xy.x)
14| 0 `mn.n(`mixx.m(`gh.h(gf))(`u.x)(`u.u))m(`xy.y)(xy.x)
15| 0 `mn.n(`mixx.m(`gh.h(gf))(`u.x)(`u.u))m(`xy.y)(xy.x)(m(`mixx.m(`gh.h(gf))(`u.x)(`u.u))n(`xy.y)(xy.x))
16| 0 `(xy.(`x.y(xx))(xy.y(xx)))(`fmn.(`xuv.xuv)((`mn.n(`mixx.m(`gh.h(gf))(`u.x)(`u.u))m(`xy.y)(xy.x))nm)(`mnfx.m(nfx))((`mnfx.f(mfx))(`fx.x))(f((`mn.n(`mixx.m(`gh.h(gf))(`u.x)(`u.u))m(`xy.y)(xy.x))n))(`fx.x))
17| 0 `(xy.(`x.y(xx))(xy.y(xx)))(`fmn.(`xuv.xuv)((`mn.n(`mixx.m(`gh.h(gf))(`u.x)(`u.u))m(`xy.y)(xy.x))nm)f((`mn.n(`mixx.m(`gh.h(gf))(`u.x)(`u.u))m(`xy.y)(xy.x))nm))
24| 0 `fx.f(f(f(fx)))
24| 0 `fx.f(f(f(f(f(f(fx))))x))
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
add two three, βx_6

```

0|0 \y.(\nx.y(\nx))(\nx.y(\nx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfx)
9|0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10|0 \mfix.m(mfix)
11|0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12|0 \mfix.m(nfix)
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15|0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16|0 (\sxy.y(\sxy.x))(\sxy.y(\sxy.x))(\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm
17|0 (\sxy.y(\sxy.x))(\sxy.y(\sxy.x))(\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm
18|0 \fx.xfs
19|0 \x.x(\sxy.x)
20|0 \x.x(\sxy.y)
21|0 \fx.fx
22|0 \fx.f(fx)
23|0 \fx.f(f(fx))
24|0 \fx.f(f(f(f(fx))))
```

25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = (MN)$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
 • app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five

```
0|0 \y.(\nx.y(\nx))(\nx.y(\nx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfx)
9|0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10|0 \mfix.m(mfix)
11|0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12|0 \mfix.m(nfix)
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15|0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16|0 (\sxy.y(\sxy.y))(\sxy.y(\sxy.y))(\sfix.(\suv.suv)((\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm))
17|0 (\sxy.y(\sxy.y))(\sxy.y(\sxy.y))(\sfix.(\suv.suv)((\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm))
18|0 \fix.xfs
19|0 \x.x(\sxy.x)
20|0 \x.x(\sxy.y)
21|0 \fix.fx
22|0 \fix.f(fx)
23|0 \fix.f(f(fx))
24|0 \fix.f(f(f(fx)))
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
mul three

```
0|0 \y.(\nx.y(\nx))(\nx.y(\nx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfx)
9|0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10|0 \mfix.m(mfix)
11|0 \mn.m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12|0 \mfix.m(nfix)
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15|0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16|0 (\sxy.y(\nx))(\sxy(\nx))(\sfnm.(\suv.suv))((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
17|0 (\sxy.y(\nx))(\sxy(\nx))(\sfnm.(\suv.suv))((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
18|0 \fx.xfs
19|0 \nx.x(\sxy.x)
20|0 \sx.x(\sxy.y)
21|0 \fx.fx
22|0 \fx.f(fx)
23|0 \fx.f(f(f))
24|0 \fx.f(f(f(f)))
25|0 (\mfix.m(nfx)(\fx.f(f)))
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode: 0 | Cursor: 0,0,28 | Item: 25,0 | Nodes Count: 160,0,0,0
Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
mul three five
0| 0 y.(\(x,y(xx))(xx,y(xx)))
1| 0 \xy.x
2| 0 \xy.y
3| 0 \z.z(\(xy.y)(xy.x))
4| 0 \xy.xyx
5| 0 \xy.xxy
6| 0 \xuv.xuv
7| 0 \fx.x
8| 0 \mfix.f(mfx)
9| 0 \mfix.m(\gh.h(gf))(\u.x)(\u.u)m
10| 0 \mfix.m(mfix)
11| 0 \mn.m(\mfix.m(\gh.h(gf))(\u.x)(\u.u))m
12| 0 \mfix.m(nfix)
13| 0 \z.z(\(xy.y)(xy.x))
14| 0 \mn.m(\mfix.m(\gh.h(gf))(\u.x)(\u.u))m(\(xy.y)(xy.x))
15| 0 \mn.n(\mfix.m(\gh.h(gf))(\u.x)(\u.u))m(\(xy.y)(xy.x))m(\(m(\mfix.m(\gh.h(gf))(\u.x)(\u.u))n(\(xy.y)(xy.x)))m
16| 0 (\(y.(xy.y(xx))(xy.y(xx))))(\(m(\mfix.m(\gh.h(gf))(\u.x)(\u.u))m(\(xy.y)(xy.x)))m
17| 0 (\(y.(xy.y(xx))(xy.y(xx))))(\(m(\mfix.m(\gh.h(gf))(\u.x)(\u.u))m(\(xy.y)(xy.x)))m)(f((\mn.n(\mfix.m(\gh.h(gf))(\u.x)(\u.u))m(\(xy.y)(xy.x)))m)(\mfix.m(nfix)))(\mfix.f(mfx)))(\fx.x))(f((\mn.n(\mfix.m(\gh.h(gf))(\u.x)(\u.u))m(\(xy.y)(xy.x)))m)(\mfix.m(nfix)))(\mfix.f(f(mfx)))(\fx.x))(f((\mn.n(\mfix.m(\gh.h(gf))(\u.x)(\u.u))m(\(xy.y)(xy.x)))m)(\mfix.m(nfix)))(\mfix.f(f(f(f(fx)))))(\fx.x))(f((\mn.n(\mfix.m(\gh.h(gf))(\u.x)(\u.u))m(\(xy.y)(xy.x)))m)(\mfix.m(nfix)))(\mfix.f(f(f(f(f(fx))))))(f((\mn.n(\mfix.m(\gh.h(gf))(\u.x)(\u.u))m(\(xy.y)(xy.x)))m)(\mfix.m(nfix)))(\mfix.f(f(f(f(f(f(fx)))))))
18| 0 \fix.xfs
19| 0 \x.x(\xy.x)
20| 0 \x.x(\xy.y)
21| 0 \fix.fx
22| 0 \fix.f(fx)
23| 0 \fix.f(f(f))
24| 0 \fix.f(f(f(f(f(fx)))))
25| 0 (\mn.m(\nf)x)(\fix.f(f(f(f(fx)))))(\fix.f(f(f(f(f(fx)))))))
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
mul three five, $\beta x 1$

```

0| 0 \y.(\nx.y(\xx))(\nx.y(\xx))
1| 0 \xy.x
2| 0 \xy.y
3| 0 \z.z(\sxy.y)(\sxy.x)
4| 0 \sxy.sxy
5| 0 \sxy.sxy
6| 0 \suv.suv
7| 0 \fx.x
8| 0 \mfix.f(mfx)
9| 0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10| 0 \mfix.m(mfix)
11| 0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12| 0 \mfix.m(nfix)
13| 0 \z.z(\sxy.y)(\sxy.x)
14| 0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15| 0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16| 0 (\sxy.x.y(\xx))(\sxy.y(\xx))(\sfnm.(\suv.suv))((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
17| 0 (\sxy.x.y(\xx))(\sxy.y(\xx))(\sfnm.(\suv.suv))((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
18| 0 \fx.xfs
19| 0 \x.x(\sxy.x)
20| 0 \x.x(\sxy.y)
21| 0 \fx.fx
22| 0 \fx.f(fx)
23| 0 \fx.f(f(fx))
24| 0 \fx.f(f(f(f(fx))))
25| 0 ((\mfix.(\fx.f(f(fx))))(nf)x)(\fx.f(f(I(f(fx)))))
```

26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = (MN)$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
 • app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
mul three five, $\beta x 2$

```

0| 0 \y.(\nx.y(\nx))(\nx.y(\nx))
1| 0 \xy.x
2| 0 \xy.y
3| 0 \z.z(\sxy.y)(\sxy.x)
4| 0 \sxy.sxy
5| 0 \sxy.sxy
6| 0 \suv.suv
7| 0 \fx.x
8| 0 \mfix.f(mfx)
9| 0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10| 0 \mfix.m(mfix)
11| 0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12| 0 \mfix.m(nfix)
13| 0 \z.z(\sxy.y)(\sxy.x)
14| 0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15| 0 \mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16| 0 (\sxy.y(\sxy.y)(\sxy.y(\sxy.y)))((\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm((\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))
17| 0 (\sxy.y(\sxy.y)(\sxy.y(\sxy.y)))((\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm(f((\mn.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm((\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm))
18| 0 \fx.xfs
19| 0 \x.x(\sxy.x)
20| 0 \x.x(\sxy.y)
21| 0 \fx.fx
22| 0 \fx.f(fx)
23| 0 \fx.f(f(fx))
24| 0 \fx.f(f(f(f(fx))))
25| 0 \fx.(\sfx.(\sf.(\sf(f(fx))))((\sfx.\sf.(\sf(f(f(f(fx))))))\sf)x
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: $\textcolor{blue}{A}$)
 • var.: a, b, c, x, y, z, \dots (set of variables: $\textcolor{yellow}{V}$)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = (MN)$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
 • app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
mul three five, $\beta x 3$

```

0| 0 \y.(\nx.y(\xx))(\nx.y(\xx))
1| 0 \xy.x
2| 0 \xy.y
3| 0 \z.z(\sxy.y)(\sxy.x)
4| 0 \sxy.sxy
5| 0 \sxy.sxy
6| 0 \suv.suv
7| 0 \fx.x
8| 0 \mfix.f(mfx)
9| 0 \mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u)m
10| 0 \mfix.m(mfix)
11| 0 \mnm.n(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m
12| 0 \mfix.m(nfix)
13| 0 \z.z(\sxy.y)(\sxy.x)
14| 0 \mnm.n(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x)
15| 0 \mnm.n(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x))
16| 0 (\sxy.y(\xx))(\sxy.y(\xx))(\mfix.m(\suv.suv)(\mnm.n(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x))nm)(\sfix.f(mfix))(\sfix.f(mfx))(\sfix.x))((f(\mnm.n(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x)))n)(\sfix.x))
17| 0 (\sxy.y(\xx))(\sxy.y(\xx))(\mfix.m(\suv.suv)(\mnm.n(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x))nm)f(\mnm.n(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x))nm)
18| 0 \fix.xfs
19| 0 \x.x(\sxy.x)
20| 0 \x.x(\sxy.y)
21| 0 \fix.fx
22| 0 \fix.f(fx)
23| 0 \fix.f(f(fx))
24| 0 \fix.f(f(f(f(fx))))
25| 0 \fix.(\sx.(\sfix.\(f(f(f(f(f(fx)))))))f((\sfix.\f(f(f(f(f(fx)))))))f((\sfix.\f(f(f(f(f(fx)))))))fx)))x
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = (MN)$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
 • app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
mul three five, $\beta \times 4$

```

0|0 \y.(\nx.y(\xx))(\nx.y(\xx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfx)
9|0 \nmix.m(\ngh.h(gf))(\nuu.x)(\nuu.u)m
10|0 \nmix.m(nfix)
11|0 \nm. n(\nmix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m
12|0 \nmix.m(nfx)
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \nm. n(\nmix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x)
15|0 \nm. n(\nmix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x)(m(\nmix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))n(\sxy.y)(\sxy.x))
16|0 (\ny.(\nx.y(\xx))(\nx.y(\xx)))(\nf m.(\nxuv.xuv))((\nm. n(\nmix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x))nm)
17|0 (\ny.(\nx.y(\xx))(\nx.y(\xx)))(\nf m.(\nxuv.xuv))((\nm. n(\nmix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x))nm)
18|0 \fx.xfs
19|0 \nx.x(\sxy.x)
20|0 \sx.x(\sxy.y)
21|0 \fx.fx
22|0 \fx.f(fx)
23|0 \fx.f(f(fx))
24|0 \fx.f(f(f(f(fx))))
25|0 \fx.(\sfx.(\sf.(\sf.(\sf.(\sf.(f(f(f(f(fx)))))))))(\sfx.(\sf.(\sf.(\sf.(\sf.(\sf.(f(f(f(f(f(fx)))))))))))))(\sfx.(\sf.(\sf.(\sf.(\sf.(\sf.(f(f(f(f(f(fx))))))))))))f(x))
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = (MN)$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
 • app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
mul three five, $\beta x 5$

```

0| 0 \y.(\nx.y(\xx))(\nx.y(\xx))
1| 0 \xy.x
2| 0 \xy.y
3| 0 \z.z(\sxy.y)(\sxy.x)
4| 0 \sxy.sxy
5| 0 \sxy.sxy
6| 0 \suv.suv
7| 0 \fx.x
8| 0 \mfix.f(mfx)
9| 0 \nmix.m(\ngh.h(gf))(\nuu.x)(\nuu.u)m
10| 0 \nmix.m(nfix)
11| 0 \nmn.n(\nmix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m
12| 0 \nmix.m(nfx)
13| 0 \z.z(\sxy.y)(\sxy.x)
14| 0 \nmn.n(\nmix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x)
15| 0 \nmn.n(\nmix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x)(m(\nmix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))n(\sxy.y)(\sxy.x))
16| 0 (\ny.(\nx.y(\xx))(\nx.y(\xx)))(\nf m.(\nxuv.xuv))((\nmn.n(\nmix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x))nm)
17| 0 (\ny.(\nx.y(\xx))(\nx.y(\xx)))(\nf m.(\nxuv.xuv))((\nmn.n(\nmix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x))nm)
18| 0 \fx.xfs
19| 0 \nx.x(\sxy.x)
20| 0 \sx.x(\sxy.y)
21| 0 \fx.fx
22| 0 \fx.f(fx)
23| 0 \fx.f(f(fx))
24| 0 \fx.f(f(f(f(fx))))
25| 0 \fx.(\sx.n(f(f(f(f(fx))))))((\fx.f(f(f(f(f(fx))))))f((\fx.f(f(f(f(f(f(fx))))))fx)))
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: $\textcolor{blue}{A}$)
 • var.: a, b, c, x, y, z, \dots (set of variables: $\textcolor{blue}{V}$)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = (MN)$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
 • app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator	0 0	\y. (\x. y(xx)) (\x. y(xx))
true	1 0	\xy. x
false	2 0	\xy. y
not	3 0	\z. z(\xy. y) (\xy. x)
and	4 0	\xy. xyx
or	5 0	\xy. xxy
ite	6 0	\xuv. xuv
zero	7 0	\fx. x
succ	8 0	\mix. f(mfx)
pred	9 0	\mix. m(\gh. h(gf)) (\u. x) (\u. u)
add	10 0	\mix. mf(nfx)
sub	11 0	\mn. n(\mix. m(\gh. h(gf)) (\u. x) (\u. u)) m
mul	12 0	\mn. n(mf) n
iszero	13 0	\z. z(\xxy. y) (\xy. x)
leq	14 0	\mn. n(\mix. m(\gh. h(gf)) (\u. x) (\u. u)) m
eq	15 0	\mn. n(\mix. m(\gh. h(gf)) (\u. x) (\u. u)) m
div	16 0	(\y. (\nx. y(xx)) (\x. y(xx))) (\tm. (\nxuv. (\anfx. mf(nfx)) ((\mix. f(mfx)) (\fx. x))
mod	17 0	(\y. (\nx. y(xx)) (\x. y(xx))) (\tmn. (\nxuv. f((\mn. n(\mix. m(\gh. h(gf)) (\u. x) (\u. u)) m
pair	18 0	\fsx. xis
fst	19 0	\x. x(\xy. x)
sec	20 0	\x. x(\xy. y)
one	21 0	\fx. fx
two	22 0	\fx. f(fx)
three	23 0	\fx. f(f(fx))
five	24 0	\fx. f(f(f(f(fx))))
mul three five, $\beta x 6$	25 0	\fx. f(f(f(f(f((\fx. f(f(f(f(fx))))))))))
	26	
	27	
	28	
	29	
	30	
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	
	43	
	44	
	45	
	46	

Create λ -terms by 3 rules	(set of λ -terms: A)
• var.: a, b, c, x, y, z, \dots	(set of variables: V)
• app.: $M, N \in A \Rightarrow (MN) \text{ in } A$	
• abs.: $X \in V, M \text{ in } A \Rightarrow (\lambda X. M) \text{ in } A$	
$A = V \mid (A A) \mid (\lambda X. A)$	
β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$	
Notation (guided by Currying of multivariate functions)	
• drop outermost parenthesis: $MN = (MN)$	
• app. left associative: $MNK = ((MN)K)$	
• abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$	
• app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$	

A crash course in λ -calculus

Create λ -terms by 3 rules	(set of λ -terms: A)
• var.: a, b, c, x, y, z, \dots	(set of variables: V)
• app.: $M, N \in A \Rightarrow (MN) \text{ in } A$	
• abs.: $X \in V, M \text{ in } A \Rightarrow (\lambda X. M) \text{ in } A$	
$A = V \mid (A A) \mid (\lambda X. A)$	
β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$	
Notation (guided by Currying of multivariate functions)	
• drop outermost parenthesis: $MN = (MN)$	
• app. left associative: $MNK = ((MN)K)$	
• abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$	
• app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$	

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
 • abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X. M) \in \Lambda$
 $\Lambda = V \uparrow (\Lambda) \uparrow (\lambda V. \Lambda)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = ((M)N)$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
 • app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (M))N$

A crash course in λ -calculus

Create λ -terms by 3 rules	(set of λ -terms: Λ)
• var.: a, b, c, x, y, z, \dots	(set of variables: V)
• app.: $M, N \in \Lambda \Rightarrow (MN) \text{ in } \Lambda$	
• abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X. M) \text{ in } \Lambda$	
$\Lambda = V \cup (\Lambda)$ $(\lambda V. \Lambda)$	
β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$	
Notation (guided by Currying of multivariate functions)	
• drop outermost parenthesis: $MN = (MN)$	
• app. left associative: $MNK = ((MN)K)$	
• abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$	
• app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$	

A crash course in λ -calculus

Y-combinator	Code
true	$\lambda y.(\lambda x.y(xx))(xx)$
false	$\lambda x.x$
not	$\lambda z.z(\lambda xy.y)(xy)$
and	$\lambda xy.yxy$
or	$\lambda xy.xxy$
ife	$\lambda uvxuv$
zero	$\lambda fx.f$
succ	$\lambda mfx.f(mfx)$
pred	$\lambda mfx.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)$
add	$\lambda mnx.m(fix)$
sub	$\lambda mnx.n(\lambda mfx.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m$
mul	$\lambda mnx.m(nfx)$
iszero	$\lambda z.z(\lambda xy.y)(xy)$
leg	$\lambda mnx.n(\lambda mfx.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(xx)$
eq	$\lambda mnx.n(\lambda mfx.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(xx))n(\lambda mfx.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(xx))$
div	$\lambda y.(\lambda x.y(xx))(\lambda x.y(xx))(\lambda mn.(\lambda xuv.xuv)((\lambda mn.(\lambda mnx.m(fix))((\lambda mix.f(mfx))(\lambda fx)))f)(\lambda mn.(\lambda mnx.m(fix))((\lambda mix.f(mfx))(\lambda fx)))f))$
mod	$\lambda y.(\lambda x.y(xx))(\lambda x.y(xx))(\lambda mn.(\lambda xuv.xuv)((\lambda mn.(\lambda mnx.m(fix))((\lambda mix.f(mfx))(\lambda fx)))f)(\lambda mn.(\lambda mnx.m(fix))((\lambda mix.f(mfx))(\lambda fx)))f))m))n)m$
pair	$\lambda xsx.xs$
fst	$\lambda x.x(\lambda y.y)$
sec	$\lambda x.x(\lambda y.y)$
one	$\lambda fix.fix$
two	$\lambda fix.f(fx)$
three	$\lambda fix.f(f(fx))$
five	$\lambda fix.f(f(f(f(fx))))$
mul three five, $\beta \times 10$	$\lambda xix.i(f(f(f(f(f(f(f(f(f(f(f(f(f(f(ix)))))))))))))))$
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	
37	
38	
39	
40	
41	
42	
43	
44	
45	
46	

Create λ -terms by 3 rules	(set of λ -terms: Λ)
• var.: a, b, c, x, y, z, \dots	(set of variables: V)
• app.: $M, N \in \Lambda \Rightarrow (MN) \text{ in } \Lambda$	
• abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X. M) \text{ in } \Lambda$	
$\Lambda = V \cup (\Lambda)$ $(\lambda V. \Lambda)$	
β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$	
Notation (guided by Currying of multivariate functions)	
• drop outermost parenthesis: $MN = (MN)$	
• app. left associative: $MNK = ((MN)K)$	
• abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$	
• app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$	

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
 • abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X. M) \in \Lambda$
 $\Lambda = V \uplus (\Lambda \uplus (\lambda V. \Lambda))$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariable functions)
 • drop outermost parenthesis: $MN = ((MN))$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy. M = ((\lambda x.(\lambda y. M)))$
 • app. takes precedence over abs.: $\lambda xy. MN = (\lambda x. (y. M))N$

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in A \Rightarrow (MN) \in A$
 • abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \in A$
 $\Lambda = V \uplus A \uplus (\lambda V. A)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = ((MN))$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
 • app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in A \Rightarrow (MN) \text{ in } A$
 • abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \text{ in } A$

$\Lambda = V \mid A \mid (\lambda X. \Lambda)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis:
 $MN = (MN)$
 • app. left associative:
 $MNK = ((MN)K)$
 • abs. right associative, only one λ :
 $\lambda xy. M = ((\lambda x. (\lambda y. M))$
 • app. takes precedence over abs.:
 $\lambda x. MN = ((\lambda x. M)N)$

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, z, \dots (set of variables: **V**)
 • app.: M, N in Λ \Rightarrow (MN) in Λ
 • abs.: X in V, M in Λ \Rightarrow $(\lambda X. M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = (MN)$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
 • app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

Create λ -terms by 3 rules	(set of λ -terms: A)
• var.: a, b, c, x, y, z, \dots	(set of variables: V)
• app.: M, N in Λ	$\Rightarrow (MN)$ in Λ
• abs.: X in V, M in Λ	$\Rightarrow (\lambda X.M)$ in Λ
$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$	
β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$	
Notation (guided by Currying of multivariate functions)	
• drop outermost parenthesis:	$MN = (MN)$
• app. left associative:	$MNK = ((MN)K)$
• abs. right associative, only one λ :	$\lambda xy.M = (\lambda x.(\lambda y.M))$
• app. takes precedence over abs.:	$\lambda x.MN = ((\lambda x.M)N)$

A crash course in λ -calculus

Create λ -terms by 3 rules	(set of λ -terms: A)
• var.: a, b, c, x, y, z, \dots	(set of variables: V)
• app.: M, N in Λ	$\Rightarrow (MN)$ in Λ
• abs.: X in V, M in Λ	$\Rightarrow (\lambda X.M)$ in Λ
$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$	
β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$	
Notation (guided by Currying of multivariate functions)	
• drop outermost parenthesis:	$MN = (MN)$
• app. left associative:	$MNK = ((MN)K)$
• abs. right associative, only one λ :	$\lambda xy.M = (\lambda x.(\lambda y.M))$
• app. takes precedence over abs.:	$\lambda x.MN = ((\lambda x.M)N)$

A crash course in λ -calculus

Create λ -terms by 3 rules	(set of λ -terms: A)
• var.: a, b, c, x, y, z, \dots	(set of variables: V)
• app.: M, N in Λ	$\Rightarrow (MN)$ in Λ
• abs.: X in V, M in Λ	$\Rightarrow (\lambda X.M)$ in Λ
$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$	
β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$	
Notation (guided by Currying of multivariate functions)	
• drop outermost parenthesis:	$MN = (MN)$
• app. left associative:	$MNK = ((MN)K)$
• abs. right associative, only one λ :	$\lambda xy.M = (\lambda x.(\lambda y.M))$
• app. takes precedence over abs.:	$\lambda x.MN = ((\lambda x.M)N)$

A crash course in λ -calculus

Create λ -terms by 3 rules	(set of λ -terms: A)
• var.: a, b, c, x, y, z, \dots	(set of variables: V)
• app.: M, N in Λ	$\Rightarrow (MN)$ in Λ
• abs.: X in V, M in Λ	$\Rightarrow (\lambda X.M)$ in Λ
$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$	
β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$	
Notation (guided by Currying of multivariate functions)	
• drop outermost parenthesis:	$MN = (MN)$
• app. left associative:	$MNK = ((MN)K)$
• abs. right associative, only one λ :	$\lambda xy.M = (\lambda x.(\lambda y.M))$
• app. takes precedence over abs.:	$\lambda x.MN = ((\lambda x.M)N)$

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, z, \dots (set of variables: **V**)
 • app.: M, N in Λ \Rightarrow (MN) in Λ
 • abs.: X in V, M in Λ \Rightarrow $(\lambda X. M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = (MN)$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
 • app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

Create λ -terms by 3 rules	(set of λ -terms: A)
• var.: a, b, c, x, y, z, \dots	(set of variables: V)
• app.: M, N in Λ	$\Rightarrow (MN)$ in Λ
• abs.: X in V, M in Λ	$\Rightarrow (\lambda X.M)$ in Λ
$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$	
β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$	
Notation (guided by Currying of multivariate functions)	
• drop outermost parenthesis:	$MN = (MN)$
• app. left associative:	$MNK = ((MN)K)$
• abs. right associative, only one λ :	$\lambda xy.M = (\lambda x.(\lambda y.M))$
• app. takes precedence over abs.:	$\lambda x.MN = ((\lambda x.M)N)$

A crash course in λ -calculus

Create λ -terms by 3 rules	(set of λ -terms: A)
• var.: a, b, c, x, y, z, \dots	(set of variables: V)
• app.: M, N in Λ	$\Rightarrow (MN)$ in Λ
• abs.: X in V, M in Λ	$\Rightarrow (\lambda X.M)$ in Λ
$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$	
β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$	
Notation (guided by Currying of multivariate functions)	
• drop outermost parenthesis:	$MN = (MN)$
• app. left associative:	$MNK = ((MN)K)$
• abs. right associative, only one λ :	$\lambda xy.M = (\lambda x.(\lambda y.M))$
• app. takes precedence over abs.:	$\lambda x.MN = ((\lambda x.M)N)$

A crash course in λ -calculus

Y-combinator	0 0	\y. (\x. y (xx)) (\x. y (xx))
true	1 0	\xy. x
false	2 0	\xy. y
not	3 0	\z. z (\xxy. y) (\xxy. x)
and	4 0	\xy. xyx
or	5 0	\xy. xxy
ifte	6 0	\xuv. xuv
zero	7 0	\fx. x
succ	8 0	\mfix. f (mfx)
pred	9 0	\mfix. m (\gh. h (gf)) (\u. x) (\u. u)
add	10 0	\mfix. m (nfx)
sub	11 0	\mn. n (\mfix. m (\gh. h (gf)) (\u. x) (\u. u)) \m
mul	12 0	\mnfix. m (nfx) \x
iszero	13 0	\z. z (\xxy. y) (\xxy. x)
leq	14 0	\mn. n (\mfix. m (\gh. h (gf)) (\u. x) (\u. u)) \m
eq	15 0	\mn. n (\mfix. m (\gh. h (gf)) (\u. x) (\u. u)) \m
div	16 0	(\y. (\x. y (xx)) (\x. y (xx))) (\fm. (\xuv. x) (\mfix. mf (nfx)) ((\mfix. f (mfx)) (\fx. x)))
mod	17 0	(\y. (\x. y (xx)) (\x. y (xx))) (\fm. (\xuv. x) f ((\mn. n (\mfix. m (\gh. h (gf)) (\u. x) (\u. u))
pair	18 0	\fx. xfs
fst	19 0	\x. x (\xxy. x)
sec	20 0	\x. x (\xxy. y)
one	21 0	\fx. fx
two	22 0	\fx. f (fx)
three	23 0	\fx. f (f (fx))
five	24 0	\fx. f (f (f (f (fx))))
fifteen	25 0	\fx. f (f (fx))))
copy	26 0	(\z. z (\xxy. y) (\xxy. x)) ((\mnfix. m (nfx) (\xxy. x) (\xxy. x))
iszero (mul zero five)	27 0	\xy. x
	28	
	29	
	30	
	31	
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	
	43	
	44	
	45	
	46	

Create λ -terms by 3 rules	(set of λ -terms: A)
• var.: a, b, c, x, y, z, \dots	(set of variables: V)
• app.: M, N in Λ	$\Rightarrow (MN)$ in Λ
• abs.: X in V, M in Λ	$\Rightarrow (\lambda X.M)$ in Λ
$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$	
β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$	
Notation (guided by Currying of multivariate functions)	
• drop outermost parenthesis:	$MN = (MN)$
• app. left associative:	$MNK = ((MN)K)$
• abs. right associative, only one λ :	$\lambda xy.M = (\lambda x.(\lambda y.M))$
• app. takes precedence over abs.:	$\lambda x.MN = ((\lambda x.M)N)$

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, z, \dots (set of variables: **V**)
 • app.: M, N in Λ \Rightarrow (MN) in Λ
 • abs.: X in V, M in Λ \Rightarrow $(\lambda X. M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = (MN)$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
 • app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in A \Rightarrow (MN) \text{ in } A$
 • abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \text{ in } A$

$\Lambda = V \mid A \mid (\lambda X. A)$

β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
- app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

```

Y-combinator
true
false
not
and
or
ifte
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
fifteen

0|0 n.y.(n.x.y(xx))(n.x.y(xx))
1|0 nxy.x
2|0 nxy.y
3|0 nz.z(nxy.y)(nxy.x)
4|0 nxy.yxy
5|0 nxy.xxy
6|0 nxyuv.xuv
7|0 nfx.x
8|0 nmfx.f(mfix)
9|0 nmfx.m(ngh.h(gf))(nux.x)(n.u.u)
10|0 nmfx.mf(nfix)
11|0 nmn.n(nmfx.m(ngh.h(gf)))(nux.x)(n.u.u)nm
12|0 nmfx.m(nfx)x
13|0 nz.z(nxy.y)(nxy.x)
14|0 nmn.n(nmfx.m(ngh.h(gf)))(nux.x)(n.u.u)nm
15|0 nmn.n(nmfx.m(ngh.h(gf)))(nux.x)(n.u.u)nm
16|0 ((n(nmfx.m(ngh.h(gf)))(nux.x)(n.u.u))nm
((ny.(n.x.y(xx))(n.x.y(xx)))((nmn.(nxyuv
((nmnf.mf(nix))((nmfx.f(mfix))(\nfx.x)))
17|0 f((nmn.n(nmfx.m(ngh.h(gf)))(nux.x)(n.u.u)nm
nfx.xis
19|0 nxy.x(nxy.x)
20|0 nxy.x(nxy.y)
21|0 nfx.fx
22|0 nfx.f(fx)
23|0 nfx.f(f(fx))
24|0 nfx.f(f(f(f(fx))))
25|0 nfx.f(f(f(f(f(fx))))))
26|0 ((nxyuv.xuv)((nz.z(nxy.y)(nxy.x))((nm
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules	(set of λ -terms: Λ)
• var.: a, b, c, x, y, z, \dots	(set of variables: V)
• app.: $M, N \in \Lambda \Rightarrow (MN) \text{ in } \Lambda$	
• abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X. M) \text{ in } \Lambda$	
$\Lambda = V \cup (\Lambda)$ $(\lambda V. \Lambda)$	
β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$	
Notation (guided by Currying of multivariate functions)	
• drop outermost parenthesis:	$MN = (MN)$
• app. left associative:	$MNK = ((MN)K)$
• abs. right associative, only one λ :	$\lambda xy. M = (\lambda x. (\lambda y. M))$
• app. takes precedence over abs.:	$\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

```

Y-combinator
true
false
not
and
or
ifte
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
fifteen
0|0 | \z. (\x. y (xx)) (\x. y (xx))
1|0 | \xy. x
2|0 | \xy. y
3|0 | \z. z (\xy. y) (\xy. x)
4|0 | \xy. xyx
5|0 | \xy. xxy
6|0 | \xuv. xuv
7|0 | \fix. x
8|0 | \mfix. f (mix)
9|0 | \mfix. m (\gh. h (gf)) (\u. x) (\u. u)
10|0 | \mfix. m f (nfix)
11|0 | \mn. n (\mfix. m (\gh. h (gf))) (\u. x) (\u. u)
12|0 | \mfix. m (nfix)
13|0 | \z. z (\xxy. y) (\xy. x)
14|0 | \mn. n (\mfix. m (\gh. h (gf))) (\u. x) (\u. u)
15|0 | \mn. n (\mfix. m (\gh. h (gf))) (\u. x) (\u. u)
16|0 | ((\mfix. m (\gh. h (gf))) (\u. x) (\u. u)))
17|0 | ((\y. (\x. y (xx)) (\x. y (xx))) (\mn. (\xuv.
18|0 | \fix. xis
19|0 | \x. x (\xy. x)
20|0 | \x. x (\xy. y)
21|0 | \fix. fx
22|0 | \fix. f (fx)
23|0 | \fix. f (f (fx))
24|0 | \fix. f (f (f (f (fx))))
25|0 | \fix. f (f (fx)))))))
26|0 | (\xuv. xuv) ((\z. z (\xxy. y) (\xy. x)) ((
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in A \Rightarrow (MN) \in A$
 • abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \in A$

$\Lambda = V \uparrow (A \uparrow (\lambda V. A))$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
- app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

($\lambda x_1 x_2$ x)
 (x₁ x₂) x
 (x₁ x₂) (x₃ (S(m f x m) (S(g h, h (g f)))) (S(u v) (S(u u))) (S(xxx v) (S(xv x) x)))

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in A \Rightarrow (MN) \in A$
 • abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \in A$

$\Lambda = V \uparrow (A \uparrow (\lambda V. A))$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
- app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

($\lambda x_1 x_2$ x)
 ((λx_1 x) (λ (λ y₁ f₁ m₁ (λ g₁ h₁ (g₁ f₁))) (λ u₁ x₁) (λ u₁ u₂))) (λ x₂ y₂) (λ x₂ y₂))

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in A \Rightarrow (MN) \text{ in } A$
 • abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \text{ in } A$

$\Lambda = V \mid A \mid (\lambda X. A)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy. M = ((\lambda x.(\lambda y. M))$
- app. takes precedence over abs.: $\lambda x. MN = ((\lambda x. M)N)$

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in A \Rightarrow (MN) \text{ in } A$
 • abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \text{ in } A$

$\Lambda = V \mid A \mid (\lambda X. A)$

β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
- app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in A \Rightarrow (MN) \in A$
 • abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \in A$

$\Lambda = V \uparrow (A \uparrow (\lambda V. A))$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
- app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

($\lambda x_1 x_2$ x)
 ((λx_1 x) (λ (λ (λ f x m) (λ g h) h (g f)) (λ u x) (λ u u)) (λ x y) (λ x y) x)

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in A \Rightarrow (MN) \text{ in } A$
 • abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \text{ in } A$

$\Lambda = V \mid A \mid (\lambda X. A)$

β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
- app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in A \Rightarrow (MN) \text{ in } A$
 • abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \text{ in } A$

$\Lambda = V \mid A \mid (\lambda X. A)$

β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
- app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in A \Rightarrow (MN) \in A$
 • abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \in A$
 $\Lambda = V \upharpoonright \{ \} \cup (V \upharpoonright \Lambda)$

β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = (MN)$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
 • app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

```

Y-combinator
true
false
not
and
or
ifte
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
fifteen
0| 0 y.((x.y(xx))(xx.y(xx)))
1| 1 xxy.x
2| 2 xxy.y
3| 3 xz.z((xy.y)(xy.x))
4| 4 xxy.xy
5| 5 xxy.xxy
6| 6 xxuv.xuv
7| 7 fix.x
8| 8 Smfix.f(mfix)
9| 9 Smfix.m((gh.h(gf))(u.x)(u.u))
10| 10 Smfix.mf(mfix)
11| 11 Smn.n(Smfix.m((gh.h(gf))(u.x)(u.u))m)
12| 12 Smfix.m(nfix)x
13| 13 nz.z((xxy.y)(xy.x))
14| 14 Smn.n(Smfix.m((gh.h(gf))(u.x)(u.u))m)
15| 15 Smn.n(Smfix.m((gh.h(gf))(u.x)(u.u))m
16| 16 ))(n(Smfix.m((gh.h(gf))(u.x)(u.u))m
17| 17 ((y.((x.y(xx))(xy.y(xx))))(Smn.(xxuv.
18| 18 xxy.xis
19| 19 xxy.x((xy.x)
20| 20 xxy.x((xy.y)
21| 21 fix.fix
22| 22 fix.f(fx)
23| 23 fix.f(f(fx))
24| 24 fix.f(f(f(f(fx))))
25| 25 fix.f(f(f(f(f(fx))))))
26| 26 ((fix.x)((fix.f(f(f(f(f(fx)))))))(xxy.y))
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
 • abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X. M) \in \Lambda$

$\Lambda = V \cup \{\Lambda\} \cup (\lambda V. \Lambda)$

β -reduction: $((\lambda X. M)N) \rightarrow M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
- app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

$((\lambda xy. x)(\lambda xy. y))(\lambda (ab. a)(\lambda gh. h(g(f))))((\lambda u. u)(\lambda v. v))(\lambda x. x)(\lambda y. y)$

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in A \Rightarrow (MN) \in A$
 • abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \in A$

$\Lambda = V \mid A \mid (\lambda X. A)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis:
 $MN = (MN)$
 • app. left associative:
 $MNK = ((MN)K)$
 • abs. right associative, only one λ :
 $\lambda xy. M = ((\lambda x. (\lambda y. M))$
 • app. takes precedence over abs.:
 $\lambda x. MN = ((\lambda x. M)N)$

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in A \Rightarrow (MN) \in A$
 • abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \in A$

$\Lambda = V \uplus A \uplus (\lambda V. A)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = ((MN))$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
 • app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in A \Rightarrow (MN) \in A$
 • abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \in A$
 $\Lambda = V \uplus (A) \uplus (\lambda V. A)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = ((MN))$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
 • app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
 • abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X. M) \in \Lambda$
 $\Lambda = V \cup \{\lambda\} \cup (\Lambda V. \Lambda)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = (MN)$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
 • app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five

```
0|0 \y.(\nx.y(\nx))(\nx.y(\nx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfx)
9|0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10|0 \mfix.m(mfix)
11|0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12|0 \mfix.m(nfix)
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15|0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16|0 (\sxy.y(\sxy.y))(\sxy.y(\sxy.y))(\sfix.(\suv.suv)((\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm))
17|0 (\sxy.y(\sxy.y))(\sxy.y(\sxy.y))(\sfix.(\suv.suv)((\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm))
18|0 \fix.xfs
19|0 \x.x(\sxy.x)
20|0 \x.x(\sxy.y)
21|0 \fix.fx
22|0 \fix.f(fx)
23|0 \fix.f(f(fx))
24|0 \fix.f(f(f(fx)))
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.

```

0| 0 \y.(\nx.y(\nx))(\nx.y(\nx))
1| 0 \xy.x
2| 0 \xy.y
3| 0 \z.z(\sxy.y)(\sxy.x)
4| 0 \sxy.sxy
5| 0 \sxy.sxy
6| 0 \suv.suv
7| 0 \fx.x
8| 0 \mfix.f(mfx)
9| 0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10| 0 \mfix.m(mfix)
11| 0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12| 0 \mfix.m(nfix)
13| 0 \z.z(\sxy.y)(\sxy.x)
14| 0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15| 0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16| 0 (\sxy.y(\sxy.y)(\sxy.y(\sxy.y)))(\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm
17| 0 (\sxy.y(\sxy.y)(\sxy.y(\sxy.y)))(\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm
18| 0 \fx.xfs
19| 0 \x.x(\sxy.x)
20| 0 \x.x(\sxy.y)
21| 0 \fx.fx
22| 0 \fx.f(fx)
23| 0 \fx.f(f(fx))
24| 0 \fx.f(f(f(fx))))
25| 0 f
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.

```

0| 0 \y.(\nx.y(\xx))(\nx.y(\xx))
1| 0 \xy.x
2| 0 \xy.y
3| 0 \z.z(\sxy.y)(\sxy.x)
4| 0 \sxy.sxy
5| 0 \sxy.sxy
6| 0 \suv.suv
7| 0 \fx.x
8| 0 \mfix.f(mfix)
9| 0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10| 0 \mfix.m(mfix)
11| 0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12| 0 \mfix.m(nfix)
13| 0 \z.z(\sxy.y)(\sxy.x)
14| 0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15| 0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16| 0 (\sxy.y(\xx))(\sxy.y(\xx))(\sfnm.(\suv.suv))((\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
17| 0 (\sxy.y(\xx))(\sxy.y(\xx))(\sfnm.(\suv.suv))((\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
18| 0 \fix.xfs
19| 0 \x.x(\sxy.x)
20| 0 \x.x(\sxy.y)
21| 0 \fix.fx
22| 0 \fix.f(fx)
23| 0 \fix.f(f(fx))
24| 0 \fix.f(f(f(fx)))
25| 0 f
26| 0 n
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
0|0 \y.(\nx.y(\nx))(\nx.y(\nx))
1|0 \xy.x
2|0 \xy.y
3|0 \xz.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfx)
9|0 \mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u)m
10|0 \mfix.m(mfix)
11|0 \mn.m(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m
12|0 \mfix.m(nfix)
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \mn.n(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x)
15|0 \mn.n(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))n(\sxy.y)(\sxy.x))
16|0 (\sxy.y(\nx))(\sxy.y(\nx))((\mfix.(\suv.suv))((\mn.n(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x))nm)
17|0 (\sxy.y(\nx))(\sxy.y(\nx))((\mfix.(\suv.suv))((\mn.n(\mfix.m(\ngh.h(gf))(\nuu.x)(\nuu.u))m(\sxy.y)(\sxy.x))nm)
18|0 \fx.xfs
19|0 \nx.x(\sxy.x)
20|0 \sx.x(\sxy.y)
21|0 \fx.fx
22|0 \fx.f(fx)
23|0 \fx.f(f(fx))
24|0 \fx.f(f(f(f(fx))))
25|0 f
26|0 n
27|0 (\sz.z(\sxy.y)(\sxy.x))n
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode: 0 | Cursor: 0,0,31 | Item: 28,0 | Nodes Count: 162,0,0,0
Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
pred n
0|0 \y.(\nx.y(\xx))(\nx.y(\xx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfx)
9|0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10|0 \mfix.m(fix)
11|0 \m.(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12|0 \mfix.m(nfx)
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \m.(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15|0 \m.(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16|0 (\sxy.y(\xx))(\sxy.x(\xx))(\mfix.(\suv.suv))((\m.(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)(\mfix.m(nfx))((\mfix.f(mfx))(\fx.x))(f((\m.(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)(\fx.x))
17|0 (\sxy.y(\xx))(\sxy.x(\xx))(\mfix.(\suv.suv))((\m.(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)f((\m.(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)
18|0 \fx.xfs
19|0 \x.x(\sxy.x)
20|0 \x.x(\sxy.y)
21|0 \fx.fx
22|0 \fx.f(fx)
23|0 \fx.f(f(fx))
24|0 \fx.f(f(f(fx))))
25|0 f
26|0 n
27|0 (\z.z(\sxy.y)(\sxy.x))n
28|0 (\mfix.m(\gh.h(gf))(\su.x)(\su.u))n
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode: 0 | Cursor: 0,0,31 | Item: 28,0 | Nodes Count: 163,0,0,0
Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
f (pred n)
0|0 \y.(\nx.y(\xx))(\nx.y(\xx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfx)
9|0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10|0 \mfix.m(mfix)
11|0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12|0 \mfix.m(nfix)
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15|0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16|0 (\sxy.y(\xx))(\sxy.x(\xx))(\mfix.m(\suv.suv)(\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)(\mfix.m(nfix))(\mfix.f(mfx))(\fx.x))((f(\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))n)(\fx.x))
17|0 (\sxy.y(\xx))(\sxy.x(\xx))(\mfix.m(\suv.suv)(\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)f((\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m)mn)n)m
18|0 \fx.xfs
19|0 \x.x(\sxy.x)
20|0 \x.x(\sxy.y)
21|0 \fx.fx
22|0 \fx.f(fx)
23|0 \fx.f(f(fx))
24|0 \fx.f(f(f(fx)))
25|0 f
26|0 n
27|0 (\z.z(\sxy.y)(\sxy.x))n
28|0 f((\mfix.m(\gh.h(gf))(\su.x)(\su.u))n)
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Mode: 0 | Cursor: 0,0,32 | Item: 29,0 | Nodes Count: 164,0,0,0
Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
f (pred n)
mul n
0|0 \y.(\nx.y(\xx))(\nx.y(\xx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfx)
9|0 \mfix.m(\gh.h(gf))(\su.x)(\su.u)m
10|0 \mfix.m(mfix)
11|0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m
12|0 \mfix.m(nfx)
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)
15|0 \mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\su.x)(\su.u))n(\sxy.y)(\sxy.x))
16|0 (\sxy.y(\xx))(\sxy.y(\xx))(\sfnm.(suv.suv)((\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)(\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)))
17|0 (\sxy.y(\xx))(\sxy.y(\xx))(\sfnm.(suv.suv)((\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)(f((\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)(\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)))f((\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)(\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)))f((\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x))nm)(\mnm.n(\mfix.m(\gh.h(gf))(\su.x)(\su.u))m(\sxy.y)(\sxy.x)))
18|0 \fx.xfs
19|0 \x.x(\sxy.x)
20|0 \x.x(\sxy.y)
21|0 \fx.fx
22|0 \fx.f(fx)
23|0 \fx.f(f(fx))
24|0 \fx.f(f(f(fx))))
25|0 f
26|0 n
27|0 (\z.z(\sxy.y)(\sxy.x))n
28|0 f((\mfix.m(\gh.h(gf))(\su.x)(\su.u))n)
29|0 (\mnm.n(\mfix.m(nfx))n)n
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,32 | Item: 29,0 | Nodes Count: 165,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
f (pred n)
mul n (f (pred n))

```

0|0 $\lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$
1|0 $\lambda xy.x$
2|0 $\lambda xy.y$
3|0 $\lambda z.z(\lambda xy.y)(\lambda xy.x)$
4|0 $\lambda xy.xyx$
5|0 $\lambda xy.xxy$
6|0 $\lambda xuv.xuv$
7|0 $\lambda fx.x$
8|0 $\lambda mfix.f(mfx)$
9|0 $\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)$
10|0 $\lambda mfix.m(mfix)$
11|0 $\lambda m.m(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m$
12|0 $\lambda mfix.m(nfix)$
13|0 $\lambda z.z(\lambda xy.y)(\lambda xy.x)$
14|0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x)$
15|0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x)m(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n(\lambda xy.y)(\lambda xy.x)$
16|0 $(\lambda y.(\lambda xy.y)(\lambda xy.y))(\lambda mfix.(xuv))((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))nm)(\lambda mfix.m(nfix))((\lambda mfix.f(mfx))(\lambda fx.x))(f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m)mn)n)(\lambda fx.x))$
17|0 $(\lambda y.(\lambda xy.y)(\lambda xy.y))(\lambda mfix.(xuv))((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))nm)f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m)mn)n)m$
18|0 $\lambda fix.xfs$
19|0 $\lambda x.x(\lambda xy.x)$
20|0 $\lambda x.x(\lambda xy.y)$
21|0 $\lambda fix.fx$
22|0 $\lambda fix.f(fx)$
23|0 $\lambda fix.f(f(fx))$
24|0 $\lambda fix.f(f(f(fx))))$
25|0 f
26|0 n
27|0 $(\lambda z.z(\lambda xy.y)(\lambda xy.x))n$
28|0 $f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)$
29|0 $(\lambda mfix.m(nfix))n((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))$
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,33 | Item: 30,0 | Nodes Count: 166,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
mul n f (pred n)
f (pred n)
ite (iszero n)
  
```

0| 0 $\lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$
1| 0 $\lambda xy.x$
2| 0 $\lambda xy.y$
3| 0 $\lambda z.z(\lambda xy.y)(\lambda xy.x)$
4| 0 $\lambda xy.xyx$
5| 0 $\lambda xy.xxy$
6| 0 $\lambda xuv.xuv$
7| 0 $\lambda fx.x$
8| 0 $\lambda mfix.f(mfx)$
9| 0 $\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)$
10| 0 $\lambda mfix.m(mfix)(nfix)$
11| 0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m$
12| 0 $\lambda mfix.m(nfix)$
13| 0 $\lambda z.z(\lambda xy.y)(\lambda xy.x)$
14| 0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x)$
15| 0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x)(m(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n(\lambda xy.y)(\lambda xy.x))$
16| 0 $(\lambda y.(\lambda x.y(xx))(\lambda x.y(xx)))(\lambda mfix.(\lambda uv.xuv)((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))nm)(\lambda mfix.m(nfix))((\lambda mfix.f(mfx))(\lambda fx.x))(f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m)mn)n)(\lambda fx.x))$
17| 0 $(\lambda y.(\lambda x.y(xx))(\lambda x.y(xx)))(\lambda mfix.(\lambda uv.xuv)((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))nm)f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m)mn)n)m$
18| 0 $\lambda fix.xfs$
19| 0 $\lambda x.x(\lambda xy.x)$
20| 0 $\lambda x.x(\lambda xy.y)$
21| 0 $\lambda fix.fx$
22| 0 $\lambda fix.f(fx)$
23| 0 $\lambda fix.f(f(fx))$
24| 0 $\lambda fix.f(f(f(fx))))$
25| 0 f
26| 0 n
27| 0 $(\lambda z.z(\lambda xy.y)(\lambda xy.x))n$
28| 0 $f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)$
29| 0 $(\lambda mfix.m(nf)x)n((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)$
30| 0 $(\lambda xuv.xuv)((\lambda z.z(\lambda xy.y)(\lambda xy.x))n)$
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

Create λ -terms by 3 rules (set of λ -terms: **A**)

- var.: a, b, c, x, y, z, \dots (set of variables: **V**)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,33 | Item: 30,0 | Nodes Count: 167,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n
f (pred n)
ifte (iszero n) one
    
```

0 | $\lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$

1 | $\lambda xy.x$

2 | $\lambda xy.y$

3 | $\lambda z.z((\lambda xy.y)(\lambda xy.x))$

4 | $\lambda xy.xy$

5 | $\lambda xy.xxy$

6 | $\lambda xuv.xuv$

7 | $\lambda fx.x$

8 | $\lambda mfix.f(mfx)$

9 | $\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)$

10 | $\lambda mfix.m(mfix)(nfix)$

11 | $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m$

12 | $\lambda mfix.m(nfix)$

13 | $\lambda z.z((\lambda xy.y)(\lambda xy.x))$

14 | $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m((\lambda xy.y)(\lambda xy.x))$

15 | $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m((\lambda xy.y)(\lambda xy.x))m((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m((\lambda xy.y)(\lambda xy.x)))$

16 | $(\lambda y.(\lambda xy.y)(\lambda xy(xx)))((\lambda mfix.(\lambda uv.xuv))((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m((\lambda xy.y)(\lambda xy.x)))m((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m((\lambda xy.y)(\lambda xy.x))))$

17 | $(\lambda y.(\lambda xy.y)(\lambda xy(xx)))((\lambda mfix.(\lambda uv.xuv))((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m((\lambda xy.y)(\lambda xy.x)))m((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m((\lambda xy.y)(\lambda xy.x))))m((\lambda f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(mn)n)m)m))$

18 | $\lambda fix.xfs$

19 | $\lambda x.x((\lambda xy.x)$

20 | $\lambda x.x((\lambda xy.y)$

21 | $\lambda fix.fx$

22 | $\lambda fix.f(fx)$

23 | $\lambda fix.f(f(fx))$

24 | $\lambda fix.f(f(f(fx))))$

25 | f

26 | n

27 | $(\lambda z.z((\lambda xy.y)(\lambda xy.x)))n$

28 | $f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)))n$

29 | $(\lambda mfix.m(nfix))n((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)))n$

30 | $(\lambda xuv.xuv)((\lambda z.z((\lambda xy.y)(\lambda xy.x)))n)(\lambda fix.fx)$

31 |

32 |

33 |

34 |

35 |

36 |

37 |

38 |

39 |

40 |

41 |

42 |

43 |

44 |

45 |

46 |

Create λ -terms by 3 rules (set of λ -terms: $\textcolor{blue}{A}$)

- var.: a, b, c, x, y, z, \dots (set of variables: $\textcolor{blue}{V}$)
- app.: M, N in $\Lambda \Rightarrow (\lambda M.N)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda A) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
f (pred n)
 0 | 0 \|_y.(\lambda x.y(\lambda x))( \lambda x.y(\lambda x))
 1 | 0 \lambda xy.x
 2 | 0 \lambda xy.y
 3 | 0 \lambda z.z(\lambda xy.y)( \lambda xy.x)
 4 | 0 \lambda xy.xyx
 5 | 0 \lambda xy.xxy
 6 | 0 \lambda xuv.xuv
 7 | 0 \lambda fx.x
 8 | 0 \lambda mfix.f(mfix)
 9 | 0 \lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)
10 | 0 \lambda mfix.m(mfix)
11 | 0 \lambda nn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m
12 | 0 \lambda nnfix.m(nfix)x
13 | 0 \lambda z.z(\lambda xy.y)( \lambda xy.x)
14 | 0 \lambda nn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m
15 | 0 \lambda nn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m
  ) (n (\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m)
16 | 0 \lambda y.(\lambda x.y(\lambda x))( \lambda x.y(\lambda x))( \lambda xy.(\lambda mfix.m(mfix)((\lambda fx.f(mfix))(\lambda fx.x)))
17 | 0 \lambda y.(\lambda x.y(\lambda x))( \lambda x.y(\lambda x))( \lambda fmn.(\lambda xuv.
f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m
\lambda fx.xis
19 | 0 \lambda x.x(\lambda xy.x)
20 | 0 \lambda x.x(\lambda xy.y)
21 | 0 \lambda fx.fx
22 | 0 \lambda fx.f(fx)
23 | 0 \lambda fx.f(f(f(fx)))
24 | 0 \lambda fx.f(f(f(f(f(fx))))))
25 | 0 f
26 | 0 n
27 | 0 (\lambda z.z(\lambda xy.y)( \lambda xy.x))n
28 | 0 f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)
29 | 0 (\lambda nnfix.m(nfix)x)n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)
30 | 0 \lambda xuv.xuv)((\lambda z.z(\lambda xy.y)( \lambda xy.x))n)(\lambda
 31 |
 32 |
 33 |
 34 |
 35 |
 36 |
 37 |
 38 |
 39 |
 40 |
 41 |
 42 |
 43 |
 44 |
 45 |
 46 |

```

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in A \Rightarrow (MN) \text{ in } A$
 • abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \text{ in } A$
 $\Lambda = V \mid (A \Lambda) \mid (\lambda V. A)$

β -reduction: $((\lambda X. M) N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = (MN)$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
 • app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

```
(\nxy.x)
(\nxy.x)(\m(\nfix.m(\ngh.h(gf))(\nux.x)(\nuu.u))\n(\nxy.y)(\nxy.x)(\nxy.y))
(\nxy.x)(\m(\nfix.m(\ngh.h(gf))(\nux.x)(\nuu.u))\m(\nxy.y)(\nxy.x))nm
(\nfix.x)(\m(\nfix.m(\ngh.h(gf))(\nux.x)(\nuu.u))\m(\nxy.y)(\nxy.x))nm
(\nfix.x)(\m(\nfix.m(\ngh.h(gf))(\nux.x)(\nuu.u))\m(\nxy.y)(\nxy.x))nm
(\nfix.x)(\m(\nfix.m(\ngh.h(gf))(\nux.x)(\nuu.u))\m(\nxy.y)(\nxy.x))nm
(\nfix.x)(\m(\nfix.m(\ngh.h(gf))(\nux.x)(\nuu.u))\m(\nxy.y)(\nxy.x))nm
```

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,33 | Item: 30,0 | Nodes Count: 169,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
    
```

0|0 $\lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$

1|0 $\lambda xy.x$

2|0 $\lambda xy.y$

3|0 $\lambda z.z(\lambda xy.y)(\lambda xy.x)$

4|0 $\lambda xy.xyx$

5|0 $\lambda xy.xxy$

6|0 $\lambda xuv.xuv$

7|0 $\lambda fx.x$

8|0 $\lambda mfix.f(mfx)$

9|0 $\lambda nmix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)$

10|0 $\lambda nmix.m(fix)$

11|0 $\lambda nm.n(\lambda mix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m$

12|0 $\lambda nmix.m(nfix)$

13|0 $\lambda z.z(\lambda xxy.y)(\lambda xy.x)$

14|0 $\lambda nm.n(\lambda mix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x)$

15|0 $\lambda nm.n(\lambda mix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x)(m(\lambda mix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x))$

16|0 $(\lambda y.(\lambda xy.y)(\lambda xy.y)(\lambda x.(y(xx))))(\lambda mfix.(\lambda uv.xuv)((\lambda nm.n(\lambda mix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x))nm)(\lambda nmfix.m(fix)))(\lambda fix.((\lambda f.(\lambda nm.n(\lambda mix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x))nm)(\lambda fix.x)))$

17|0 $(\lambda y.(\lambda xy.y)(\lambda xy.y)(\lambda x.(y(xx))))(\lambda mfix.(\lambda uv.xuv)((\lambda nm.n(\lambda mix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x))nm)(\lambda f((\lambda nm.n(\lambda mix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m)mn)n)m)$

18|0 $\lambda fix.xfs$

19|0 $\lambda x.x(\lambda xy.x)$

20|0 $\lambda x.x(\lambda xy.y)$

21|0 $\lambda fix.fx$

22|0 $\lambda fix.f(fx)$

23|0 $\lambda fix.f(f(fx))$

24|0 $\lambda fix.f(f(f(fx))))$

25|0 f

26|0 n

27|0 $(\lambda z.z(\lambda xxy.y)(\lambda xy.x))n$

28|0 $f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)$

29|0 $(\lambda nmfix.m(nfix)x)n((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)$

30|0 $\lambda u.(\lambda xuv.xuv)((\lambda z.z(\lambda xxy.y)(\lambda xy.x))n)(\lambda fix.(f((\lambda mfix.m(nfix)x)n((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))))$

31| ↗ $\lambda n.\text{ite}(\text{iszero } n \text{ one} (\text{mul } n (f (\text{pred } n))))$

32|

33|

34|

35|

36|

37|

38|

39|

40|

41|

42|

43|

44|

45|

46|

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,33 | Item: 30,0 | Nodes Count: 170,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
f (pred n)
mul n (f (pred n))

```

0|0 $\lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$
1|0 $\lambda xy.x$
2|0 $\lambda xy.y$
3|0 $\lambda z.z(\lambda xy.y)(\lambda xy.x)$
4|0 $\lambda xy.xyx$
5|0 $\lambda xy.xxy$
6|0 $\lambda xuv.xuv$
7|0 $\lambda fx.x$
8|0 $\lambda mfix.f(mfx)$
9|0 $\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)$
10|0 $\lambda mfix.m(mfix)$
11|0 $\lambda m.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m$
12|0 $\lambda mfix.m(nfix)$
13|0 $\lambda z.z(\lambda xy.y)(\lambda xy.x)$
14|0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x)$
15|0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x)m(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n(\lambda xy.y)(\lambda xy.x)$
16|0 $(\lambda y.(\lambda xy.y)(\lambda xy.y))(\lambda mfix.(\lambda uv.xuv)((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))nm)(\lambda mfix.m(nfix))((\lambda mfix.f(mfx))(\lambda fx.x))(\lambda f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m)mn)n)(\lambda fx.x))$
17|0 $(\lambda y.(\lambda xy.y)(\lambda xy.y))(\lambda mfix.(\lambda uv.xuv)((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))nm)f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m)mn)n)m$
18|0 $\lambda fix.xfs$
19|0 $\lambda x.x(\lambda xy.x)$
20|0 $\lambda x.x(\lambda xy.y)$
21|0 $\lambda fix.fx$
22|0 $\lambda fix.f(fx)$
23|0 $\lambda fix.f(f(fx))$
24|0 $\lambda fix.f(f(f(fx))))$
25|0 f
26|0 n
27|0 $(\lambda z.z(\lambda xy.y)(\lambda xy.x))n$
28|0 $f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)$
29|0 $(\lambda mfix.m(nfix))n((\lambda ((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))n))$
30|0 $\lambda fn.(\lambda xuv.xuv)((\lambda z.z(\lambda xy.y)(\lambda xy.x))n)(\lambda fix.fx)((\lambda mn.m(mfix))x)n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)))$
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

Create λ -terms by 3 rules (set of λ -terms: **A**)

- var.: a, b, c, x, y, z, \dots (set of variables: **V**)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,33 | Item: 30,0 | Nodes Count: 171,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
    
```

Y-combinator

```

0|0 \y.(\nx.y(\nx))(\nx.y(\nx))
1|0 \xy.x
2|0 \xy.y
3|0 \z.z(\sxy.y)(\sxy.x)
4|0 \sxy.sxy
5|0 \sxy.sxy
6|0 \suv.suv
7|0 \fx.x
8|0 \mfix.f(mfx)
9|0 \mn.m(\gh.h(gf))(\nu.u)(\nu.u)
10|0 \mn.m(m(nix))
11|0 \mn.n(\mfix.m(\gh.h(gf))(\nu.u)(\nu.u))m
12|0 \mn.m(m(nix))
13|0 \z.z(\sxy.y)(\sxy.x)
14|0 \mn.n(\mfix.m(\gh.h(gf))(\nu.u)(\nu.u))m(\sxy.y)(\sxy.x)
15|0 \mn.n(\mfix.m(\gh.h(gf))(\nu.u)(\nu.u))m(\sxy.y)(\sxy.x)(m(\mfix.m(\gh.h(gf))(\nu.u)(\nu.u))n(\sxy.y)(\sxy.x))
16|0 (\sxy.(\sxy.y)(\sxy.x))(\mfix.(\suv.suv))((\mn.n(\mfix.m(\gh.h(gf))(\nu.u)(\nu.u))m(\sxy.y)(\sxy.x))n)(\suv.suv)
17|0 (\sxy.(\sxy.y)(\sxy.x))(\mfix.(\suv.suv))((\mn.n(\mfix.m(\gh.h(gf))(\nu.u)(\nu.u))m(\sxy.y)(\sxy.x))n)(f((\mn.n(\mfix.m(\gh.h(gf))(\nu.u)(\nu.u))m)mn)n)m
18|0 \fx.xs
19|0 \nx.x(\sxy.x)
20|0 \nx.x(\sxy.y)
21|0 \fx.fx
22|0 \fx.f(fx)
23|0 \fx.f(f(fx))
24|0 \fx.f(f(f(fx))))
25|0 f
26|0 n
27|0 (\z.z(\sxy.y)(\sxy.x))n
28|0 f((\mfix.m(\gh.h(gf))(\nu.u)(\nu.u))n)
29|0 (\mfix.m(nf)x)n(f((\mfix.m(\gh.h(gf))(\nu.u)(\nu.u))n))
30|0 (\sxy.(\sxy.y)(\sxy.x))(\mfix.(\suv.suv))((\z.z(\sxy.y)(\sxy.x))n)(\fx.fx)((\mfix.m(nf)x)n(f((\mfix.m(\gh.h(gf))(\nu.u)(\nu.u))n)))
    
```

fac = $\text{Y}(\lambda\text{fn.ite (iszero n) one (mul n (f (pred n)))})$

Create λ -terms by 3 rules (set of λ -terms: **A**)

- var.: a, b, c, x, y, z, \dots (set of variables: **V**)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

```

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
f (pred n)
mul n (f (pred n))
fac

0 | 0 \|y. (\$x.y(\$x)) (\$x.y(\$x))
1 | 0 \xy.x
2 | 0 \xy.y
3 | 0 \z. z(\$xy.y) (\$xy.x)
4 | 0 \xy. xyx
5 | 0 \xy. xxy
6 | 0 \xuv. xuv
7 | 0 \fx.x
8 | 0 \$mfix f(mfix)
9 | 0 \$mfix.m(\$gh.h(gf)) (\$u.x) (\$u.u)
10 | 0 \$mfix.m(mfix)
11 | 0 \$mn.n(\$mfix.m(\$gh.h(gf)) (\$u.x) (\$u.u))m
12 | 0 \$mnfix.m(nfix)x
13 | 0 \z. z(\$xy.y) (\$xy.x)
14 | 0 \$mn.n(\$mfix.m(\$gh.h(gf)) (\$u.x) (\$u.u))m
15 | 0 \$mn.n(\$mfix.m(\$gh.h(gf)) (\$u.x) (\$u.u))m
    ) (n (\$mfix.m(\$gh.h(gf)) (\$u.x) (\$u.u))m
16 | 0 (\$y. (\$x.y(\$x)) (\$x.y(\$x))) (\$mfix.m(\$gh.h(gf)) (\$x.x) (\$x.u))
    ) (\$mfix.m(mfix)) ((\$mfix.m(mfix)) (\$fx.x))
17 | 0 (\$y. (\$x.y(\$x)) (\$x.y(\$x))) (\$mn.n(\$xuv.
    f((\$mn.n(\$mfix.m(\$gh.h(gf)) (\$u.x) (\$u.u))m
18 | 0 \fx.xis
19 | 0 \x. x(\$xy.x)
20 | 0 \x. (\$xy.y)
21 | 0 \fx.fx
22 | 0 \fx. f(fx)
23 | 0 \fx. f(f(fx))
24 | 0 \fx. f(f(f(f(fx))))
25 | 0 f
26 | 0 n
27 | 0 (\$z. z(\$xy.y) (\$xy.x))n
28 | 0 f((\$mfix.m(\$gh.h(gf)) (\$u.x) (\$u.u))n)
29 | 0 (\$mnfix.m(nfix)x)n (f((\$mfix.m(\$gh.h(gf)) (
30 | 0 (\$y. (\$x.y(\$x)) (\$x.y(\$x))) (\$fn. (\$xuv.x
    gh.h(gf)) (\$u.x) (\$u.u))n)))
31 |
32 |
33 |
34 |
35 |
36 |
37 |
38 |
39 |
40 |
41 |
42 |
43 |
44 |
45 |

```

Create λ -terms by 3 rules	(set of λ -terms: A)
• var.: a, b, c, x, y, z, \dots	(set of variables: V)
• app.: $M, N \in A \Rightarrow (MN) \text{ in } A$	
• abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \text{ in } A$	
$A = V \mid (A A) \mid (\lambda V. A)$	
β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$	
Notation (guided by Currying of multivariate functions)	
• drop outermost parenthesis:	$MN = (MN)$
• app. left associative:	$MNK = ((MN)K)$
• abs. right associative, only one λ :	$\lambda xy. M = (\lambda x. (\lambda y. M))$
• app. takes precedence over abs.:	$\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 172,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
mul n (f (pred n))
f (pred n)
fac
fac five
  
```

0| 0 $\lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$
1| 0 $\lambda xy.x$
2| 0 $\lambda xy.y$
3| 0 $\lambda z.z(\lambda xy.y)(\lambda xy.x)$
4| 0 $\lambda xy.xyx$
5| 0 $\lambda xy.xxy$
6| 0 $\lambda xuv.xuv$
7| 0 $\lambda fx.x$
8| 0 $\lambda mfix.f(mfx)$
9| 0 $\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)$
10| 0 $\lambda mfix.m(mfix)$
11| 0 $\lambda m.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m$
12| 0 $\lambda mfix.m(nfix)$
13| 0 $\lambda z.z(\lambda xxy.y)(\lambda xy.x)$
14| 0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x)$
15| 0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x)m(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n(\lambda xxy.y)(\lambda xy.x)$
16| 0 $(\lambda y.(\lambda xy.yxx)(\lambda xy.yxx))(\lambda fn.(\lambda xuv.xuv)((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x))nm)(\lambda mfix.m(nfix))((\lambda mfix.m(nfix))(\lambda fx.x))(\lambda f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x))mn))(\lambda fx.x))$
17| 0 $(\lambda y.(\lambda xy.yxx)(\lambda xy.yxx))(\lambda fn.(\lambda xuv.xuv)((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x))nm)f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m)mn)n)m$
18| 0 $\lambda fix.fx$
19| 0 $\lambda x.x(\lambda xy.x)$
20| 0 $\lambda x.x(\lambda xy.y)$
21| 0 $\lambda fix.fx$
22| 0 $\lambda fx.f(fx)$
23| 0 $\lambda fx.f(f(fx))$
24| 0 $\lambda fx.f(f(f(fx))))$
25| 0 f
26| 0 n
27| 0 $(\lambda z.z(\lambda xxy.y)(\lambda xy.x))n$
28| 0 $f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)$
29| 0 $(\lambda mfix.m(nfix))n(\lambda ((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))$
30| 0 $(\lambda y.(\lambda xy.yxx)(\lambda xy.yxx))(\lambda fn.(\lambda xuv.xuv)((\lambda z.z(\lambda xxy.y)(\lambda xy.x))n)(\lambda fix.fx)((\lambda mfix.m(nfix))(\lambda fx.f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))))(\lambda fx.f(f(f(fx))))$
31| 0 $(\lambda y.(\lambda xy.yxx)(\lambda xy.yxx))(\lambda fn.(\lambda xuv.xuv)((\lambda z.z(\lambda xxy.y)(\lambda xy.x))n)(\lambda fix.fx)((\lambda mfix.m(nfix))n(\lambda f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))))(\lambda fx.f(f(f(fx))))$
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|

Create λ -terms by 3 rules (set of λ -terms: $\textcolor{blue}{A}$)

- var.: a, b, c, x, y, z, \dots (set of variables: $\textcolor{blue}{V}$)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$

$\Lambda = V \cup (\Lambda\Lambda) \cup (\Lambda V \Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 177,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
fac
fac five,  $\beta$ x1

```

0| 0 | $\lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$
1| 0 | $\lambda xy.x$
2| 0 | $\lambda xy.y$
3| 0 | $\lambda z.z(\lambda xy.y)(\lambda xy.x)$
4| 0 | $\lambda xy.xyx$
5| 0 | $\lambda xy.xxy$
6| 0 | $\lambda xuv.xuv$
7| 0 | $\lambda fx.x$
8| 0 | $\lambda mfix.f(mfx)$
9| 0 | $\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)$
10| 0 | $\lambda mfix.m(mfix)(nfix)$
11| 0 | $\lambda m.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m$
12| 0 | $\lambda mfix.m(nfix)$
13| 0 | $\lambda z.z(\lambda xy.y)(\lambda xy.x)$
14| 0 | $\lambda m.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x)$
15| 0 | $\lambda m.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x)(m(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))$
16| 0 | $(\lambda y.(\lambda xy.y)(\lambda xy.y)(\lambda xy.y)(\lambda xy.y)(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))m$
17| 0 | $(\lambda y.(\lambda xy.y)(\lambda xy.y)(\lambda xy.y)(\lambda xy.y)(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))m$
18| 0 | $\lambda fix.fx$
19| 0 | $\lambda x.x(\lambda xy.x)$
20| 0 | $\lambda x.x(\lambda xy.y)$
21| 0 | $\lambda fix.fx$
22| 0 | $\lambda fix.fff$
23| 0 | $\lambda fix.ffff$
24| 0 | $\lambda fix.ffff(fix))$
25| 0 | f
26| 0 | n
27| 0 | $(\lambda z.z(\lambda xy.y)(\lambda xy.x))n$
28| 0 | $f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)$
29| 0 | $(\lambda mfix.m(nfix))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))$
30| 0 | $(\lambda y.(\lambda xy.y)(\lambda xy.y)(\lambda xy.y)(\lambda xy.y)(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)$
31| 0 | $(\lambda x.(\lambda fin.(\lambda xuv.xuv)((\lambda z.z(\lambda xy.y)(\lambda xy.x))n)(\lambda fix.fx)((\lambda mfix.m(nfix))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))))(xx))(\lambda x.(\lambda fin.(\lambda xuv.xuv)((\lambda z.z(\lambda xy.y)(\lambda xy.x))n)(\lambda fix.fx)((\lambda mfix.m(nfix))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))))(xx))(\lambda fix.ffff(fix)))$
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$
- $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 176,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
fac
fac five,  $\beta$ x2

```

0|0 $\lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$
1|0 $\lambda xy.x$
2|0 $\lambda xy.y$
3|0 $\lambda z.z((\lambda xy.y)(\lambda xy.x))$
4|0 $\lambda xy.xy$
5|0 $\lambda xy.xxy$
6|0 $\lambda xuv.xuv$
7|0 $\lambda fx.x$
8|0 $\lambda mfix.f(mfx)$
9|0 $\lambda mn.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)$
10|0 $\lambda mnfix.m(nfix)$
11|0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m$
12|0 $\lambda mnfix.m(nfix)$
13|0 $\lambda z.z((\lambda xy.y)(\lambda xy.x))$
14|0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m((\lambda xy.y)(\lambda xy.x))$
15|0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m((\lambda xy.y)(\lambda xy.x))m((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m((\lambda xy.y)(\lambda xy.x)))$
16|0 $(\lambda y.(\lambda xy.yxx)(\lambda xy.yxx))(\lambda fm.(\lambda xuv.xuv)((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m((\lambda xy.y)(\lambda xy.x)))m((\lambda mnfix.m(nfix))((\lambda mfix.f(mfx))(\lambda fx.x)))(f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m((\lambda xy.y)(\lambda xy.x))))m))$
17|0 $(\lambda y.(\lambda xy.yxx)(\lambda xy.yxx))(\lambda fm.(\lambda xuv.xuv)((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m((\lambda xy.y)(\lambda xy.x)))m(f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(mn)n)m)))$
18|0 $\lambda fx.fxs$
19|0 $\lambda x.x((\lambda xy.x)$
20|0 $\lambda x.x((\lambda xy.y)$
21|0 $\lambda fx.fx$
22|0 $\lambda fx.f(fx)$
23|0 $\lambda fx.f(f(fx))$
24|0 $\lambda fx.f(f(f(f(fx))))$
25|0 f
26|0 n
27|0 $(\lambda z.z((\lambda xy.y)(\lambda xy.x)))n$
28|0 $f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)$
29|0 $(\lambda mnfix.m(nfix))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))$
30|0 $(\lambda y.(\lambda xy.yxx)(\lambda xy.yxx))(\lambda fm.(\lambda xuv.xuv)((\lambda z.z((\lambda xy.y)(\lambda xy.x)))n)(\lambda fx.fx)((\lambda mnfix.m(nfx)x)n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)))$
31|0 $(\lambda n.(\lambda xuv.xuv)((\lambda z.z((\lambda xy.y)(\lambda xy.x)))n)(\lambda fx.fx)((\lambda mnfix.m(nfx)x)n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)))$
 $((\lambda x.(\lambda fm.(\lambda xuv.xuv)((\lambda z.z((\lambda xy.y)(\lambda xy.x)))n)(\lambda fx.fx)((\lambda mnfix.m(nfx)x)n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))))(xx))(\lambda x.(\lambda fm.(\lambda xuv.xuv)((\lambda z.z((\lambda xy.y)(\lambda xy.x)))n)(\lambda fx.fx)((\lambda mnfix.m(nfx)x)n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))))(xx)))$
32|0
33|0
34|0
35|0
36|0
37|0
38|0
39|0
40|0
41|0
42|0

Create λ -terms by 3 rules (set of λ -terms: $\textcolor{blue}{A}$)

- var.: a, b, c, x, y, z, \dots (set of variables: $\textcolor{blue}{V}$)
- app.: $M, N \in \Lambda \Rightarrow (\textcolor{red}{MN}) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$

$\Lambda = V \cup (\Lambda\Lambda) \cup (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Y-combinator	0 0	$\lambda y. (\lambda x. y(\bar{x}x)) (\lambda x. y(\bar{x}x))$
true	1 0	$\lambda xy. x$
false	2 0	$\lambda xy. y$
not	3 0	$\lambda z. z(\lambda xy. y) (\lambda xy. x)$
and	4 0	$\lambda xy. xy$
or	5 0	$\lambda xy. xxy$
ife	6 0	$\lambda xuv. xuv$
zero	7 0	$\lambda fx. x$
succ	8 0	$\lambda mfix. f(mfix)$
pred	9 0	$\lambda mfix. m(\lambda gh. h(gf)) (\lambda u. x) (\lambda u. u)$
add	10 0	$\lambda mfix. m(mfix)$
sub	11 0	$\lambda mn. n(\lambda mfix. m(\lambda gh. h(gf)) (\lambda u. x) (\lambda u. u)) m$
mul	12 0	$\lambda mnfx. m(nfix)$
iszero	13 0	$\lambda z. z(\lambda xy. y) (\lambda xy. x)$
leq	14 0	$\lambda mn. n(\lambda mfix. m(\lambda gh. h(gf)) (\lambda u. x) (\lambda u. u)) m$
eq	15 0	$\lambda mn. n(\lambda mfix. m(\lambda gh. h(gf)) (\lambda u. x) (\lambda u. u)) m$
div	16 0	$(\lambda y. (\lambda x. y(\bar{x}x)) (\lambda x. y(\bar{x}x))) (\lambda mn. (\lambda xuv.$ $(\lambda mfix. m(nfix)) ((\lambda mfix. f(mfix)) (\lambda fx. x))$
mod	17 0	$(\lambda y. (\lambda x. y(\bar{x}x)) (\lambda x. y(\bar{x}x))) (\lambda mn. (\lambda xuv.$ $f((\lambda mn. n(\lambda mfix. m(\lambda gh. h(gf)) (\lambda u. x) (\lambda u. u)) m$
pair	18 0	$\lambda fsx. fsx$
fst	19 0	$\lambda x. x(\lambda xy. x)$
sec	20 0	$\lambda x. x(\lambda xy. y)$
one	21 0	$\lambda fx. fx$
two	22 0	$\lambda fx. f(fx)$
three	23 0	$\lambda fx. f(f(fx))$
five	24 0	$\lambda fx. f(f(f(f(fx))))$
var.	25 0	f
var.	26 0	n
iszero n	27 0	$(\lambda z. z(\lambda xxy. y) (\lambda xy. x)) n$
f (pred n)	28 0	$f((\lambda mfix. m(\lambda gh. h(gf)) (\lambda u. x) (\lambda u. u)) n)$
mul n (f (pred n))	29 0	$(\lambda mnfx. m(nfix)) n(f((\lambda mfix. m(\lambda gh. h(gf)) (\lambda u. x) (\lambda u. u)) m$
fac	30 0	$(\lambda y. (\lambda x. y(\bar{x}x)) (\lambda x. y(\bar{x}x))) (\lambda fn. (\lambda xuv. x$ $gh. h(gf)) (\lambda u. x) (\lambda u. u)) n))$
fac five, $\beta x3$	31 0	$(\lambda n. (\lambda xuv. xuv) ((\lambda z. z(\lambda xxy. y) (\lambda xy. x)) n$ $(\lambda x) n)) (\lambda fx. fx) ((\lambda mnfx. m(nfix)) n(f((\lambda mfix. m(nfix))$ $(\lambda xxy. y) (\lambda xy. x)) n)) (\lambda fx. fx) ((\lambda mnfx. m(nfix))$ $gf)) (\lambda u. x) (\lambda u. u)) n))))) (\lambda fx. f(f(f(f(f(x$
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
 • abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X. M) \in \Lambda$
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = (MN)$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
 • app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

```

(\lambda xy.x)
((xy)x)(m(\$mix,m(\$gh,h(gf)))(\u.x)(\u.u))n(\$xxy,y)(\xy,x)(\xy,x)
mn,n(\$mix,m(\$gh,h(gf)))(\u.x)(\u.u)m(\$xxy,y)(\xy,x))nm(
)n(\$mix,m(\$gh,h(gf)))(\u.x)(\u.u)m(mn)n))(\$fx,x))
mn,n(\$mix,m(\$gh,h(gf)))(\u.x)(\u.u)m(\$xxy,y)(\xy,x))nm(
)n)m

```

```

(u,u))n)
.z(\$xxy,y)(\$xy,x))n)(\$fx,fx)((\$mnfix,m(nf)x)n(f((\$mix,m(\$x
x))((\$mnfix,m(nf)x)n((\$x,(\$tn,(\$uuv,xuv))(\$z,z)(\$xxy,y)(\$xy
.h(gf)))(\u.x)(\u.u))n))))(xx))(\$x,(\$tn,(\$uuv,xuv))((\$z,z
(\$mix,m(\$gh,h(gf)))(\u.x)(\u.u))n))))(x))((\$mix,m(\$gh,h(

```

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 182,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
f
fac
fac five,  $\beta$ x4

```

0| 0 `y.(\(x,y)(xx))(xy,y(xx))`
1| 0 `xy.x`
2| 0 `xy.y`
3| 0 `z,z(\(xy,y)(xy,x))`
4| 0 `xy.xyx`
5| 0 `xy.xxy`
6| 0 `xuv.xuv`
7| 0 `fx.x`
8| 0 `mfix.f(mfx)`
9| 0 `mfix.m(\(gh,h(gf)))(u,x)(u,u)`
10| 0 `mfix.m(mfix.m(nfix.m(\(gh,h(gf)))(u,x)(u,u)))`
11| 0 `mfix.m(mfix.m(\(gh,h(gf)))(u,x)(u,u))m`
12| 0 `mfix.m(nfix.m(\(gh,h(gf)))(u,x)(u,u))`
13| 0 `z,z(\(xy,y)(xy,x))`
14| 0 `mfix.m(mfix.m(\(gh,h(gf)))(u,x)(u,u))m(\(xy,y)(xy,x))`
15| 0 `mfix.m(mfix.m(\(gh,h(gf)))(u,x)(u,u))m(\(xy,y)(xy,x))m(\(mfix.m(\(gh,h(gf)))(u,x)(u,u))n(\(xy,y)(xy,x))`
16| 0 `(y,(\(xy,y)(xx))(xy,y(xx)))\(\(mfix.m(xuv,xuv)((mfix.m(mfix.m(\(gh,h(gf)))(u,x)(u,u))m(\(xy,y)(xy,x)))nm)\)`
17| 0 `(y,(\(xy,y)(xx))(xy,y(xx)))\(\(mfix.m(xuv,xuv)((mfix.m(mfix.m(\(gh,h(gf)))(u,x)(u,u))m(\(xy,y)(xy,x)))nm)\)`
18| 0 `fx,fx`
19| 0 `x,x(\(xy,x))`
20| 0 `x,x(\(xy,y))`
21| 0 `fx,fx`
22| 0 `fx,f(fx)`
23| 0 `fx,f(f(fx))`
24| 0 `fx,f(f(f(f(fx))))`
25| 0 `f`
26| 0 `n`
27| 0 `(z,z(\(xy,y)(xy,x)))n`
28| 0 `f((mfix.m(\(gh,h(gf)))(u,x)(u,u)))`
29| 0 `(mfix.m(nfix.m(\(gh,h(gf)))(u,x)(u,u)))n`
30| 0 `(y,(\(xy,y)(xx))(xy,y(xx)))\(\(mfix.m(xuv,xuv)((z,z(\(xy,y)(xy,x)))n)(fx,fx)((mfix.m(nfix.m(fx))n(f((mfix.m(\(gh,h(gf)))(u,x)(u,u))))`
31| 0 `(xuv,xuv)((z,z(\(xy,y)(xy,x))(fx,f(f(f(f(fx)))))(fx,fx)((mfix.m(nfix.m(fx))n(f((mfix.m(\(gh,h(gf)))(u,x)(u,u))))`
32| 0 `(xuv,xuv)((z,z(\(xy,y)(xy,x)))n)(fx,fx)((mfix.m(nfix.m(fx))n(f((mfix.m(\(gh,h(gf)))(u,x)(u,u))))`
33| 0 `(xuv,xuv)((z,z(\(xy,y)(xy,x)))n)(fx,fx)((mfix.m(nfix.m(fx))n(f((mfix.m(\(gh,h(gf)))(u,x)(u,u))))`
34| 0 `(xuv,xuv)((z,z(\(xy,y)(xy,x)))n)(fx,fx)((mfix.m(nfix.m(fx))n(f((mfix.m(\(gh,h(gf)))(u,x)(u,u))))`
35| 0 `(xuv,xuv)((z,z(\(xy,y)(xy,x)))n)(fx,fx)((mfix.m(nfix.m(fx))n(f((mfix.m(\(gh,h(gf)))(u,x)(u,u))))`
36| 0 `(xuv,xuv)((z,z(\(xy,y)(xy,x)))n)(fx,fx)((mfix.m(nfix.m(fx))n(f((mfix.m(\(gh,h(gf)))(u,x)(u,u))))`
37| 0 `(xuv,xuv)((z,z(\(xy,y)(xy,x)))n)(fx,fx)((mfix.m(nfix.m(fx))n(f((mfix.m(\(gh,h(gf)))(u,x)(u,u))))`
38| 0 `(xuv,xuv)((z,z(\(xy,y)(xy,x)))n)(fx,fx)((mfix.m(nfix.m(fx))n(f((mfix.m(\(gh,h(gf)))(u,x)(u,u))))`
39| 0 `(xuv,xuv)((z,z(\(xy,y)(xy,x)))n)(fx,fx)((mfix.m(nfix.m(fx))n(f((mfix.m(\(gh,h(gf)))(u,x)(u,u))))`
40| 0 `(xuv,xuv)((z,z(\(xy,y)(xy,x)))n)(fx,fx)((mfix.m(nfix.m(fx))n(f((mfix.m(\(gh,h(gf)))(u,x)(u,u))))`
41| 0 `(xuv,xuv)((z,z(\(xy,y)(xy,x)))n)(fx,fx)((mfix.m(nfix.m(fx))n(f((mfix.m(\(gh,h(gf)))(u,x)(u,u))))`
42| 0 `(xuv,xuv)((z,z(\(xy,y)(xy,x)))n)(fx,fx)((mfix.m(nfix.m(fx))n(f((mfix.m(\(gh,h(gf)))(u,x)(u,u))))`

Create λ -terms by 3 rules (set of λ -terms: A)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
- $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

A crash course in λ -calculus

Create λ -terms by 3 rules	(set of λ -terms: A)
• var.: a, b, c, x, y, z, \dots	(set of variables: V)
• app.: $M, N \in A \Rightarrow (MN) \text{ in } A$	
• abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \text{ in } A$	
$A = V \mid (A A) \mid (\lambda V. A)$	
β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$	
Notation (guided by Currying of multivariate functions)	
• drop outermost parenthesis:	$MN = (MN)$
• app. left associative:	$MNK = ((MN)K)$
• abs. right associative, only one λ :	$\lambda xy. M = (\lambda x. (\lambda y. M))$
• app. takes precedence over abs.:	$\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 181,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
fac
fac five,  $\beta\eta$ 7

```

0| 0 `y.(\(x.y(xx))(xy.y(xx))`
1| 0 `xy.x`
2| 0 `xy.y`
3| 0 `z.z(\(xy.y)(xy.x)`
4| 0 `xy.xy`
5| 0 `xy.xxy`
6| 0 `xuv.xuv`
7| 0 `fx.x`
8| 0 `mfix.f(mfx)`
9| 0 `mfix.m(\(gh.h(gf))(\(u.x)(u.u))`
10| 0 `mfix.m(mfix.m(\(gh.h(gf))(\(u.x)(u.u))))`
11| 0 `mfix.m(\(mfix.m(\(gh.h(gf))(\(u.x)(u.u))))`
12| 0 `mfix.m(mfix.m(\(gh.h(gf))(\(u.x)(u.u))))`
13| 0 `z.z(\(xxy.y)(xy.x)`
14| 0 `mfix.m(\(mfix.m(\(gh.h(gf))(\(u.x)(u.u))m(\(xxy.y)(xy.x))`
15| 0 `mfix.m(\(mfix.m(\(gh.h(gf))(\(u.x)(u.u))m(\(xxy.y)(xy.x))`
16| 0 `(mfix.m(\(mfix.m(\(gh.h(gf))(\(u.x)(u.u))m(\(xxy.y)(xy.x))`
17| 0 `(mfix.m(\(mfix.m(\(gh.h(gf))(\(u.x)(u.u))m(\(xxy.y)(xy.x))`
18| 0 `fx.fxs`
19| 0 `x.x(\(xy.x)`
20| 0 `x.x(\(xy.y)`
21| 0 `fx.fx`
22| 0 `fx.f(fx)`
23| 0 `fx.f(f(fx))`
24| 0 `fx.f(f(f(f(fx))))`
25| 0 `f`
26| 0 `n`
27| 0 `(z.z(\(xxy.y)(xy.x))`
28| 0 `f((mfix.m(\(gh.h(gf))(\(u.x)(u.u))))`
29| 0 `(mfix.m(nfx))`
30| 0 `(mfix.m(\(mfix.m(\(gh.h(gf))(\(u.x)(u.u))))`
31| 0 `(z.z(\(xxy.y)(xy.x))`
32| 33| 34| 35| 36| 37| 38| 39| 40| 41| 42|

Create λ -terms by 3 rules (set of λ -terms: A)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 182,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
fac
fac five,  $\beta$ x8

```

0| 0| $\lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$
1| 0| $\lambda xy.x$
2| 0| $\lambda xy.y$
3| 0| $\lambda z.z(\lambda xy.y)(\lambda xy.x)$
4| 0| $\lambda xy.xyx$
5| 0| $\lambda xy.xxy$
6| 0| $\lambda xuv.xuv$
7| 0| $\lambda fx.x$
8| 0| $\lambda mfix.f(mfx)$
9| 0| $\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)$
10| 0| $\lambda mnfix.mf(nfix)$
11| 0| $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m$
12| 0| $\lambda mnfix.m(nfix)$
13| 0| $\lambda z.z(\lambda xy.y)(\lambda xy.x)$
14| 0| $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x)$
15| 0| $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x)m(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n(\lambda xy.y)(\lambda xy.x)$
16| 0| $(\lambda y.(\lambda xy.y)(\lambda xy.x))(\lambda mfix.(lambda uv)(\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))nm)(\lambda mnfix.mf(nfix))((\lambda mfix.f(mfx))(\lambda fx.x))(f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m)n))(\lambda fx.x))$
17| 0| $(\lambda y.(\lambda xy.y)(\lambda xy.x))(\lambda mfix.(lambda uv)(\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))nm)f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m)mn)n)m$
18| 0| $\lambda fx.fxs$
19| 0| $\lambda x.x(\lambda xy.x)$
20| 0| $\lambda x.x(\lambda xy.y)$
21| 0| $\lambda fx.fx$
22| 0| $\lambda fx.f(fx)$
23| 0| $\lambda fx.f(f(fx))$
24| 0| $\lambda fx.f(f(f(f(fx))))$
25| 0| f
26| 0| n
27| 0| $(\lambda z.z(\lambda xy.y)(\lambda xy.x))n$
28| 0| $f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)$
29| 0| $(\lambda mnfix.m(nfix))n((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))$
30| 0| $(\lambda y.(\lambda xy.y)(\lambda xy.x))(\lambda mfix.(lambda uv)(\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))n)(\lambda fx.fx)((\lambda mnfix.m(nfix))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)))$
31| 0| $((\lambda z.z(\lambda xy.y)(\lambda xy.x))(\lambda fx.fx)((\lambda mnfix.m(nfix))n(\lambda fx.f(f(f(f(fx))))))((\lambda x.(\lambda fx.(\lambda xuv.xuv))(\lambda z.z(\lambda xy.y)(\lambda xy.x))n)(\lambda fx.fx)((\lambda mnfix.m(nfix))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))))(xx))(\lambda x.(\lambda fx.(\lambda xuv.xuv))(\lambda z.z(\lambda xy.y)(\lambda xy.x))n)(\lambda fx.fx)((\lambda mnfix.m(nfix))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))))(xx))$
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$

$\Lambda = V \cup (\Lambda\Lambda) \cup (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Create λ -terms by 3 rules	(set of λ -terms: A)
• var.: a, b, c, x, y, z, \dots	(set of variables: V)
• app.: M, N in Λ	$\Rightarrow (MN)$ in Λ
• abs.: X in V, M in Λ	$\Rightarrow (\lambda X.M)$ in Λ
$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$	
β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$	
Notation (guided by Currying of multivariate functions)	
• drop outermost parenthesis:	$MN = (MN)$
• app. left associative:	$MNK = ((MN)K)$
• abs. right associative, only one λ :	$\lambda xy.M = (\lambda x.(\lambda y.M))$
• app. takes precedence over abs.:	$\lambda x.MN = ((\lambda x.M)N)$

A crash course in λ -calculus

Y-combinator	0 0	$\lambda y. (\lambda x. y(\bar{x}x))(\lambda x. y(\bar{x}x))$
true	1 0	$\lambda xy. x$
false	2 0	$\lambda xy. y$
not	3 0	$\lambda z. z(\lambda xy. y)(\lambda xy. x)$
and	4 0	$\lambda xy. xxy$
or	5 0	$\lambda xy. xcy$
ite	6 0	$\lambda uv. xuv$
zero	7 0	$\lambda fx. x$
succ	8 0	$\lambda mix. f(mfx)$
pred	9 0	$\lambda mix. m(\lambda gh. h(gf))(\lambda u. x)(\lambda u. u)$
add	10 0	$\lambda mix. m(mfx)$
sub	11 0	$\lambda mix. m(\lambda mix. m(\lambda gh. h(gf))(\lambda u. x)(\lambda u. u))m$
mul	12 0	$\lambda mix. m(nfx)$
iszero	13 0	$\lambda z. z(\lambda xy. y)(\lambda xy. x)$
leq	14 0	$\lambda mn. n(\lambda mix. m(\lambda gh. h(gf))(\lambda u. x)(\lambda u. u))m$
eq	15 0	$\lambda mn. n(\lambda mix. m(\lambda gh. h(gf))(\lambda u. x)(\lambda u. u))m$))((n(\lambda mix. m(\lambda gh. h(gf))(\lambda u. x)(\lambda u. u))m) ((Ny. (\lambda xy. y)(\lambda xy. x)))((Nx. y(\bar{y}y))(\lambda mix. m(\lambda gh. h(gf))(\lambda u. x)(\lambda u. u))m))
div	16 0	$(\lambda mix. m(nfx))((\lambda mix. m(nfx))((\lambda mix. f(mfx))((\lambda fx. x)))$
mod	17 0	$(\lambda y. (\lambda xy. y)(\lambda xy. x)))((\lambda mix. (\lambda mix. m(\lambda gh. h(gf))(\lambda u. x)(\lambda u. u))m))$ $f((\lambda mn. n(\lambda mix. m(\lambda gh. h(gf))(\lambda u. x)(\lambda u. u))m))$
pair	18 0	$\lambda xs. xfs$
fst	19 0	$\lambda x. x(\lambda xy. x)$
sec	20 0	$\lambda x. x(\lambda xy. y)$
one	21 0	$\lambda fx. fx$
two	22 0	$\lambda fx. f(fx)$
three	23 0	$\lambda fx. f(f(fx))$
five	24 0	$\lambda fx. f(f(f(f(fx))))$
var.	25 0	f
var.	26 0	\bar{n}
iszero n	27 0	$(\lambda z. z(\lambda xy. y)(\lambda xy. x))n$
f (pred n)	28 0	$f((\lambda mix. m(\lambda gh. h(gf))(\lambda u. x)(\lambda u. u))n)$
mul n (f (pred n))	29 0	$(\lambda mix. m(nfx))n f((\lambda mix. m(\lambda gh. h(gf))(\lambda u. x)(\lambda u. u))n)$
fac	30 0	$(\lambda y. (\lambda xy. y)(\lambda xy. x)))((\lambda mix. (\lambda mix. m(\lambda gh. h(gf))(\lambda u. x)(\lambda u. u))m))$ $((\lambda xy. y)(\lambda xy. y))$
fac five, $\beta \times 10$	31 0	$((\lambda xy. y)(\lambda xy. y))$ $((\lambda x. (\lambda fin. (\lambda uv. xuv))((\lambda z. z(\lambda xy. y)(\lambda xy. x))n)))((\lambda x. (\lambda fin. (\lambda uv. xuv))((\lambda z. z(\lambda xy. y)(\lambda xy. x))n)))n))$ $((\lambda x. ((\lambda fin. (\lambda uv. xuv))((\lambda z. z(\lambda xy. y)(\lambda xy. x))n)))n))((\lambda x. ((\lambda fin. (\lambda uv. xuv))((\lambda z. z(\lambda xy. y)(\lambda xy. x))n)))n)))n))$
	32	
	33	
	34	
	35	
	36	
	37	
	38	
	39	
	40	
	41	
	42	

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 180,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
fac
fac five,  $\beta x 11$ 

```

0| 0 $\lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$
1| 0 $\lambda xy.x$
2| 0 $\lambda xy.y$
3| 0 $\lambda z.z(\lambda xy.y)(\lambda xy.x)$
4| 0 $\lambda xy.xyx$
5| 0 $\lambda xy.xxy$
6| 0 $\lambda xuv.xuv$
7| 0 $\lambda fx.x$
8| 0 $\lambda mfix.f(mfx)$
9| 0 $\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)$
10| 0 $\lambda mfix.m(fix)$
11| 0 $\lambda m.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m$
12| 0 $\lambda mfix.m(nfix)$
13| 0 $\lambda z.z(\lambda xy.y)(\lambda xy.x)$
14| 0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x)$
15| 0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x)m(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n(\lambda xy.y)(\lambda xy.x)$
16| 0 $(\lambda y.(\lambda xy.y)(\lambda xy.y))(\lambda mfix.(\lambda fx.(xuv))((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))nm)(\lambda mfix.m(fix))((\lambda mfix.m(fix))(\lambda fx.x))(\lambda f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m)n))(\lambda fx.x))$
17| 0 $(\lambda y.(\lambda xy.y)(\lambda xy.y))(\lambda mfix.(\lambda fx.(xuv))((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))nm)f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m)mn)n)m$
18| 0 $\lambda fx.fx$
19| 0 $\lambda x.x(\lambda xy.x)$
20| 0 $\lambda x.x(\lambda xy.y)$
21| 0 $\lambda fx.fx$
22| 0 $\lambda fx.f(fx)$
23| 0 $\lambda fx.f(f(fx))$
24| 0 $\lambda fx.f(f(f(fx))))$
25| 0 f
26| 0 n
27| 0 $(\lambda z.z(\lambda xy.y)(\lambda xy.x))n$
28| 0 $f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)$
29| 0 $(\lambda mfix.m(nfix))n(\lambda f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))$
30| 0 $(\lambda y.(\lambda xy.y)(\lambda xy.y))(\lambda mfix.(\lambda fx.(xuv))((\lambda z.z(\lambda xy.y)(\lambda xy.x))n)(\lambda fx.fx)((\lambda mfix.m(nfix))n(\lambda f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)))$
31| 0 $(\lambda z.z(\lambda xy.y)(\lambda xy.x))((\lambda mfix.m(nfix))n(\lambda f((\lambda f(f(fx))))((\lambda x.(\lambda mfix.(\lambda fx.(xuv))((\lambda z.z(\lambda xy.y)(\lambda xy.x))n)(\lambda fx.fx)((\lambda mfix.m(nfix))n(\lambda f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))))(xx))(\lambda x.(\lambda mfix.(\lambda fx.(xuv))((\lambda z.z(\lambda xy.y)(\lambda xy.x))n)(\lambda fx.fx)((\lambda mfix.m(nfix))n(\lambda f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))))(xx))((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))((\lambda fx.f(f(f(fx)))))))$
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 179,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
fac
fac five,  $\beta x 12$ 

```

0|0 $\lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$

1|0 $\lambda xy.x$

2|0 $\lambda xy.y$

3|0 $\lambda z.z(\lambda xy.y)(\lambda xy.x)$

4|0 $\lambda xy.xyx$

5|0 $\lambda xy.xxy$

6|0 $\lambda xuv.xuv$

7|0 $\lambda fx.x$

8|0 $\lambda mfix.f(mfx)$

9|0 $\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)$

10|0 $\lambda mfix.m(mfix)$

11|0 $\lambda m.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m$

12|0 $\lambda mfix.m(nfix)$

13|0 $\lambda z.z(\lambda xy.y)(\lambda xy.x)$

14|0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x)$

15|0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x)(m(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))$

16|0 $(\lambda y.(\lambda xy.y)(\lambda xy.x))(\lambda mfix.(\lambda fx.(xuv))((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))nm)(\lambda mfix.m(nfix))((\lambda mfix.f(mfx))(\lambda fx.x)))(f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))nm)(\lambda mfix.m(nfix))((\lambda mfix.f(mfx))(\lambda fx.x)))$

17|0 $(\lambda y.(\lambda xy.y)(\lambda xy.x))(\lambda mfix.(\lambda fx.(xuv))((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))nm)(f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))nm)(\lambda mfix.m(nfix))((\lambda mfix.f(mfx))(\lambda fx.x))))$

18|0 $\lambda fx.fxs$

19|0 $\lambda x.x(\lambda xy.x)$

20|0 $\lambda x.x(\lambda xy.y)$

21|0 $\lambda fx.fx$

22|0 $\lambda fx.f(fx)$

23|0 $\lambda fx.f(f(fx))$

24|0 $\lambda fx.f(f(f(f(fx))))$

25|0 f

26|0 n

27|0 $(\lambda z.z(\lambda xy.y)(\lambda xy.x))n$

28|0 $f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)$

29|0 $(\lambda mfix.m(nfx))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))$

30|0 $(\lambda y.(\lambda xy.y)(\lambda xy.x))(\lambda mfix.m(nfx))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))$

31|0 $(\lambda y.(\lambda mfix.m(nfx))(\lambda fx.f(f(f(f(fx))))))((\lambda x.(\lambda fx.(\lambda xuv.xuv))(\lambda z.z(\lambda xy.y)(\lambda xy.x))n)(\lambda fx.fx)((\lambda mfix.m(nfx))(\lambda fx.f(f(f(f(fx)))))))(xx)(\lambda x.(\lambda fx.(\lambda xuv.xuv))((\lambda z.z(\lambda xy.y)(\lambda xy.x))n)(\lambda fx.fx)((\lambda mfix.m(nfx))(\lambda fx.f(f(f(fx)))))))(xx)((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))(\lambda fx.f(f(f(f(fx))))))$

32|

33|

34|

35|

36|

37|

38|

39|

40|

41|

42|

Create λ -terms by 3 rules (set of λ -terms: A)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 178,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
fac
fac five,  $\beta x 13$ 

```

0|0 $\lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$

1|0 $\lambda xy.x$

2|0 $\lambda xy.y$

3|0 $\lambda z.z(\lambda xy.y)(\lambda xy.x)$

4|0 $\lambda xy.xyx$

5|0 $\lambda xy.xxy$

6|0 $\lambda xuv.xuv$

7|0 $\lambda fx.x$

8|0 $\lambda mfix.f(mfx)$

9|0 $\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)$

10|0 $\lambda mfix.m(mfix)$

11|0 $\lambda m.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m$

12|0 $\lambda mfix.m(nfix)$

13|0 $\lambda z.z(\lambda xxy.y)(\lambda xy.x)$

14|0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x)$

15|0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x)m(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n(\lambda xxy.y)(\lambda xy.x)$

16|0 $(\lambda y.(\lambda xy.yxx)(\lambda xy.yxx))(\lambda fm.(\lambda xuv.xuv)((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x))nm)(\lambda mfix.m(nfix))((\lambda mfix.m(fix))(\lambda fx.x))(f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m)mn)n))(\lambda fx.x))$

17|0 $(\lambda y.(\lambda xy.yxx)(\lambda xy.yxx))(\lambda fm.(\lambda xuv.xuv)((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x))nm)f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m)mn)n)m)$

18|0 $\lambda fix.xfs$

19|0 $\lambda x.x(\lambda xy.x)$

20|0 $\lambda x.x(\lambda xy.y)$

21|0 $\lambda fix.fx$

22|0 $\lambda fix.f(fx)$

23|0 $\lambda fix.f(f(fx))$

24|0 $\lambda fix.f(f(f(f(fx))))$

25|0 f

26|0 n

27|0 $(\lambda z.z(\lambda xxy.y)(\lambda xy.x))n$

28|0 $f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)$

29|0 $(\lambda mfix.m(nfix))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))$

30|0 $(\lambda y.(\lambda xy.yxx)(\lambda xy.yxx))(\lambda fm.(\lambda xuv.xuv)((\lambda z.z(\lambda xxy.y)(\lambda xy.x))n)(\lambda fix.fx)((\lambda mnfix.m(nfix))(\lambda fx.x)(f((\lambda mnfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)))$

31|0 $(\lambda mfix.m(nfix))(\lambda fx.f(f(f(f(f(fx))))((\lambda x.(\lambda fm.(\lambda xuv.xuv)((\lambda z.z(\lambda xxy.y)(\lambda xy.x))n)(\lambda fix.fx)((\lambda mnfix.m(nfix))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))))(xx))(\lambda x.(\lambda fm.(\lambda xuv.xuv)((\lambda z.z(\lambda xxy.y)(\lambda xy.x))n)(\lambda fix.fx)((\lambda mnfix.m(nfix))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))))(xx))((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))(\lambda fx.f(f(f(fx))))))$

32|0

33|0

34|0

35|0

36|0

37|0

38|0

39|0

40|0

41|0

42|0

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$
- $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 182,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
fac
fac five,  $\beta$ x14

```

0|0 $\lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$
1|0 $\lambda xy.x$
2|0 $\lambda xy.y$
3|0 $\lambda z.z(\lambda xy.y)(\lambda xy.x)$
4|0 $\lambda xy.xyx$
5|0 $\lambda xy.xxy$
6|0 $\lambda xuv.xuv$
7|0 $\lambda fx.x$
8|0 $\lambda mfix.f(mfx)$
9|0 $\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)$
10|0 $\lambda mfix.m(mfix)$
11|0 $\lambda m.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m$
12|0 $\lambda mfix.m(nfix)$
13|0 $\lambda z.z(\lambda xy.y)(\lambda xy.x)$
14|0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x)$
15|0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x)m(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n(\lambda xy.y)(\lambda xy.x)$
16|0 $(\lambda y.(\lambda xy.y)(\lambda xy.y))(\lambda mfix.(\lambda fx.(xuv)))((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))m(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n(\lambda xy.y)(\lambda xy.x))$
17|0 $(\lambda y.(\lambda xy.y)(\lambda xy.y))(\lambda mfix.(\lambda fx.(xuv)))((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))m(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n(\lambda xy.y)(\lambda xy.x))m(\lambda f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m)mn)n)m$
18|0 $\lambda fix.fx$
19|0 $\lambda x.x(\lambda xy.x)$
20|0 $\lambda x.x(\lambda xy.y)$
21|0 $\lambda fix.fx$
22|0 $\lambda fix.fff$
23|0 $\lambda fix.fff(fx)$
24|0 $\lambda fix.fff(f(f(fx))))$
25|0 f
26|0 n
27|0 $(\lambda z.z(\lambda xy.y)(\lambda xy.x))n$
28|0 $f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)$
29|0 $(\lambda mfix.m(nfix))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))$
30|0 $(\lambda y.(\lambda xy.y)(\lambda xy.y))(\lambda mfix.(\lambda fx.(xuv)))((\lambda z.z(\lambda xy.y)(\lambda xy.x))n)(\lambda fix.fx)((\lambda mfix.m(nfix))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)))$
31|0 $(\lambda mfix.(\lambda fix.fff(f(f(fx))))(nfix))((\lambda z.z(\lambda mfix.(\lambda fx.(xuv)))((\lambda z.z(\lambda xy.y)(\lambda xy.x))n)(\lambda fix.fx)((\lambda mfix.m(nfix))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))))(xix))((\lambda z.z(\lambda mfix.(\lambda fx.(xuv)))((\lambda z.z(\lambda xy.y)(\lambda xy.x))n)(\lambda fix.fx)((\lambda mfix.m(nfix))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))))(xix))((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))(\lambda fix.fff(f(f(fx))))))$
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|

Create λ -terms by 3 rules (set of λ -terms: $\textcolor{blue}{A}$)

- var.: a, b, c, x, y, z, \dots (set of variables: $\textcolor{blue}{V}$)
- app.: M, N in $\Lambda \Rightarrow (\lambda M.N)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 181,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
mul n (f (pred n))
f (pred n)
fac
fac five,  $\beta$ x15
  
```

0|0 $\lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$

1|0 $\lambda xy.x$

2|0 $\lambda xy.y$

3|0 $\lambda z.z(\lambda xy.y)(\lambda xy.x)$

4|0 $\lambda xy.xyx$

5|0 $\lambda xy.xxy$

6|0 $\lambda xuv.xuv$

7|0 $\lambda fx.x$

8|0 $\lambda mfix.f(mfx)$

9|0 $\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)$

10|0 $\lambda mfix.m(fix)$

11|0 $\lambda m.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m$

12|0 $\lambda mfix.m(nfix)$

13|0 $\lambda z.z(\lambda xy.y)(\lambda xy.x)$

14|0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x)$

15|0 $\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x)(m(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n(\lambda xy.y)(\lambda xy.x))$

16|0 $(\lambda y.(\lambda xy.y)(\lambda xy.y))(\lambda fm.(\lambda xuv.xuv)((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))nm)(\lambda mfix.m(fix))((\lambda mfix.m(fix))(\lambda fx.x))(f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m)n))(\lambda fx.x))$

17|0 $(\lambda y.(\lambda xy.y)(\lambda xy.y))(\lambda fm.(\lambda xuv.xuv)((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xy.y)(\lambda xy.x))nm)(f((\lambda mn.n(\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m)mn)n)m)$

18|0 $\lambda fix.xfs$

19|0 $\lambda x.x(\lambda xy.x)$

20|0 $\lambda x.x(\lambda xy.y)$

21|0 $\lambda fix.fx$

22|0 $\lambda fix.f(fx)$

23|0 $\lambda fix.f(f(fx))$

24|0 $\lambda fix.f(f(f(f(fx))))$

25|0 f

26|0 n

27|0 $(\lambda z.z(\lambda xy.y)(\lambda xy.x))n$

28|0 $f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)$

29|0 $(\lambda mfix.m(nfix))n((\lambda f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))$

30|0 $(\lambda y.(\lambda xy.y)(\lambda xy.y))(\lambda fm.(\lambda xuv.xuv)((\lambda z.z(\lambda xy.y)(\lambda xy.x))n)(\lambda fix.fx)((\lambda mfix.m(nfix))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)))$

31|0 $\lambda fix.(fix.(f(f(f(f(f(fx))))))((\lambda x.(\lambda fix.(\lambda xuv.xuv)((\lambda z.z(\lambda xy.y)(\lambda xy.x))n)(\lambda fix.fx)((\lambda mfix.m(nfix))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))))(xx)))((\lambda x.(\lambda fix.(\lambda xuv.xuv)((\lambda z.z(\lambda xy.y)(\lambda xy.x))n)(\lambda fix.fx)((\lambda mfix.m(nfix))n(f((\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n))))(xx)))((\lambda fix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))(\lambda fix.f(f(f(fx))))))$

32|0

33|0

34|0

35|0

36|0

37|0

38|0

39|0

40|0

41|0

42|0

Create λ -terms by 3 rules (set of λ -terms: A)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
 • abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

A crash course in λ -calculus

A crash course in λ -calculus

```

Y-combinator
true
false
not
and
or
ife
zero
succ
pred
add
sub
mul
iszero
leg
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
f (pred n)
fac
fac five,  $\beta x 18$ 
0 | 0  $\lambda y. (\lambda x. y (xx)) ( \lambda x. y (xx))$ 
1 | 0  $\lambda xy. x$ 
2 | 0  $\lambda xy. y$ 
3 | 0  $\lambda z. z (\lambda xy. y) (\lambda xy. x)$ 
4 | 0  $\lambda xy. yxy$ 
5 | 0  $\lambda xy. xxy$ 
6 | 0  $\lambda xuv. xuv$ 
7 | 0  $\lambda fx. x$ 
8 | 0  $\lambda mfix. f (mfix)$ 
9 | 0  $\lambda mfix. m (\lambda gh. h (gf)) (\lambda u. x) (\lambda u. u)$ 
10 | 0  $\lambda mfix. m (nfix)$ 
11 | 0  $\lambda nn. n (\lambda mfix. m (\lambda gh. h (gf)) (\lambda u. x) (\lambda u. u)) m$ 
12 | 0  $\lambda mfix. m (nfix)$ 
13 | 0  $\lambda z. z (\lambda xy. y) (\lambda xy. x)$ 
14 | 0  $\lambda nn. n (\lambda mfix. m (\lambda gh. h (gf)) (\lambda u. x) (\lambda u. u)) m$ 
15 | 0  $\lambda nn. n (\lambda mfix. m (\lambda gh. h (gf)) (\lambda u. x) (\lambda u. u)) m$ 
16 | 0  $(\lambda y. (\lambda x. y (xx))) (\lambda x. y (xx)) (\lambda mfix. m (\lambda xy. x) (\lambda uv. u v))$ 
17 | 0  $(\lambda y. (\lambda x. y (xx))) (\lambda x. y (xx)) (\lambda mfix. m (\lambda xy. x) (\lambda uv. u v))$ 
 $f ((\lambda nn. n (\lambda mfix. m (\lambda gh. h (gf)) (\lambda u. x) (\lambda u. u)) m$ 
18 | 0  $\lambda xss. xfs$ 
19 | 0  $\lambda xx. x (\lambda xy. x)$ 
20 | 0  $\lambda xx. x (\lambda xy. y)$ 
21 | 0  $\lambda x. fx$ 
22 | 0  $\lambda x. f (fx)$ 
23 | 0  $\lambda x. f (f (fx))$ 
24 | 0  $\lambda x. f (f (f (f (fx))))$ 
25 | 0  $f$ 
26 | 0  $n$ 
27 | 0  $(\lambda z. z (\lambda xy. y) (\lambda xy. x)) n$ 
28 | 0  $f ((\lambda mfix. m (\lambda gh. h (gf)) (\lambda u. x) (\lambda u. u)) n)$ 
29 | 0  $(\lambda mfix. m (nfix)) n f ((\lambda mfix. m (\lambda gh. h (gf)) (\lambda u. x) (\lambda u. u)) n)$ 
30 | 0  $(\lambda y. (\lambda x. y (xx))) (\lambda x. y (xx)) (\lambda mfix. m (\lambda xy. x) (\lambda uv. u v))$ 
 $gh. h (gf) ((\lambda u. x) ((\lambda u. u) n))$ 
31 | 0  $\lambda xss. (x (\lambda uv. xuv)) ((\lambda z. z (\lambda xy. y) (\lambda xy. x)) n)$ 

```

A crash course in λ -calculus

Create λ -terms by 3 rules	(set of λ -terms: A)
• var.: a, b, c, x, y, z, \dots	(set of variables: V)
• app.: M, N in Λ	$\Rightarrow (MN)$ in Λ
• abs.: X in V, M in Λ	$\Rightarrow (\lambda X.M)$ in Λ
$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$	
β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$	
Notation (guided by Currying of multivariate functions)	
• drop outermost parenthesis:	$MN = (MN)$
• app. left associative:	$MNK = ((MN)K)$
• abs. right associative, only one λ :	$\lambda xy.M = (\lambda x.(\lambda y.M))$
• app. takes precedence over abs.:	$\lambda x.MN = ((\lambda x.M)N)$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 194,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n f(pred n)
fac
fac five,  $\beta x 20$ 

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$
- $\Lambda = V \cup (\Lambda\Lambda) \cup (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: Λ)
 • var.: a, b, c, x, y, z, \dots (set of variables: V)
 • app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
 • abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X. M) \in \Lambda$

$\Lambda = V \cup \{\lambda\} \cup (\Lambda \times \Lambda)$

β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = (MN)$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
 • app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 191,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n f(pred n)
fac
fac five,  $\beta x 40$ 

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$
- $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 199,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n f(pred n)
fac
fac five,  $\beta x 50$ 

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$

$\Lambda = V \cup (\Lambda\Lambda) \cup (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 222,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
mul n (f (pred n))
fac
fac five,  $\beta x 150$ 

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 260,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n f(pred n)
fac
fac five,  $\beta x 250$ 

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 284,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n f(pred n)
fac
fac five,  $\beta x 350$ 

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$
- $\Lambda = V \cup (\Lambda\Lambda) \cup (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

A crash course in λ -calculus

```

Y-combinator
true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
f (pred n)
fac
fac five,  $\beta x$ 550
 0 | 0 (x, (x, y (xx))) (x, y (xx))
 1 | 0 (x, y, x)
 2 | 0 (x, y, y)
 3 | 0 (x, z (xxy, y)) (xxy, x)
 4 | 0 (x, y, xxy)
 5 | 0 (x, y, xxy)
 6 | 0 (xuv, xuv)
 7 | 0 (fix, x)
 8 | 0 (mfix, f (mfx))
 9 | 0 (mfix, m (gh, h (gf))) (u, x) (u, u)
10 | 0 (mfix, m (nfix))
11 | 0 (mn, n (mfix, m (gh, h (gf)))) (u, x) (u, u)) m
12 | 0 (mfix, m (nfix))
13 | 0 (x, z (xxy, y)) (xxy, x)
14 | 0 (mn, n (mfix, m (gh, h (gf)))) (u, x) (u, u)) m
15 | 0 (mn, n (mfix, m (gh, h (gf)))) (u, x) (u, u)) m
  )) (n (mfix, m (gh, h (gf))) (u, x) (u, u)) m
16 | 0 (sy, (x, y (xx))) (x, y (xx)))
17 | 0 (mfix, m (nfix)) ((mfix, f (mfx)) (fix, x))
  ((y, (x, y (xx))) (x, y (xx))) (fm, (xuv, x))
  f ((mn, (mfix, m (gh, h (gf)))) (u, x) (u, u))
  ) (fix, x)
19 | 0 (x, x (xxy, x))
20 | 0 (x, x (xxy, y))
21 | 0 (fix, fix)
22 | 0 (fix, f (fx))
23 | 0 (fix, f (f (fx)))
24 | 0 (fix, f (f (f (f (fx))))) )
25 | 0 I
26 | 0 n
27 | 0 (x, z (xxy, y)) (xxy, x)) n
28 | 0 I ((mfix, m (gh, h (gf))) (u, x) (u, u)) n
29 | 0 (mfix, m (nfix)) f ((mfix, m (gh, h (gf))) (
  (y, (x, y (xx))) (x, y (xx))) (fix, (xuv, x)
  gh, h (gf)) (u, x) (u, u)) n))
30 | 0 (fix, f ((gh, h (g ((x, (fix, (xuv, xuy) (
  (x, x (xxy, y)) (xxy, x)) n)))) (x, x (xxy, y)) (xxy, x)) n)))
31 | 0

```

fac five, $\beta \times 550$

Create λ -terms by 3 rules	(set of λ -terms: A)
• var.: a, b, c, x, y, z, \dots	(set of variables: V)
• app.: M, N in Λ	$\Rightarrow (MN)$ in Λ
• abs.: X in V, M in Λ	$\Rightarrow (\lambda X.M)$ in Λ
$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$	
β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$	
Notation (guided by Currying of multivariate functions)	
• drop outermost parenthesis:	$MN = (MN)$
• app. left associative:	$MNK = ((MN)K)$
• abs. right associative, only one λ :	$\lambda xy.M = (\lambda x.(\lambda y.M))$
• app. takes precedence over abs.:	$\lambda x.MN = ((\lambda x.M)N)$

A crash course in λ -calculus

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in A \Rightarrow (MN) \in A$
 • abs.: $X \in V, M \in A \Rightarrow (\lambda X. M) \in A$
 $\Lambda = V \cup A \cup (\lambda V. A)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = (MN)$
 • app. left associative:
 • abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
 • app. takes precedence over abs.: $\lambda x. MN = ((\lambda x. M)N)$

fac five, $\beta \times 2550$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 316,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
fac
fac five,  $\beta \times 3550$ 

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X. M) \in \Lambda$

$\Lambda = V \cup (\Lambda\Lambda) \cup (\lambda V.\Lambda)$

β -reduction: $((\lambda X. M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
- app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 245,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n f(pred n)
fac
fac five,  $\beta$ x4550

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$

$\Lambda = V \cup (\Lambda\Lambda) \cup (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 315,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
fac
fac five,  $\beta x 6550$ 

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

fac five, $\beta \times 7550$

A crash course in λ -calculus

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 297,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
fac
fac five,  $\beta x 9550$ 

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 351,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n f(pred n)
fac
fac five,  $\beta x 11550$ 

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$

$\Lambda = V \cup (\Lambda\Lambda) \cup (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 289,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
mul n (f (pred n))
fac
fac five,  $\beta$ x12550
  
```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$
- $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

A crash course in λ -calculus

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 302,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n f(pred n)
fac
fac five,  $\beta$ x15550

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 305,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
mul n (f (pred n))
fac
fac five,  $\beta$ x16550
  
```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ
- $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

A crash course in λ -calculus

A crash course in λ -calculus

Create λ -terms by 3 rules	(set of λ -terms: A)
• var.: a, b, c, x, y, z, \dots	(set of variables: V)
• app.: M, N in Λ	$\Rightarrow (MN)$ in Λ
• abs.: X in V, M in Λ	$\Rightarrow (\lambda X.M)$ in Λ
$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$	
β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$	
Notation (guided by Currying of multivariate functions)	
• drop outermost parenthesis:	$MN = (MN)$
• app. left associative:	$MNK = ((MN)K)$
• abs. right associative, only one λ :	$\lambda xy.M = (\lambda x.(\lambda y.M))$
• app. takes precedence over abs.:	$\lambda x.MN = ((\lambda x.M)N)$

Page 19 of 50

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 342,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
fac
fac five,  $\beta x 20550$ 

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 375,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
mul n (f (pred n))
fac
fac five,  $\beta x 21550$ 

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

A crash course in λ -calculus

A crash course in λ -calculus

Y-combinator	Mode 0 Cursor: 0,0,35 Items: 37,0 Nodes Count: 367,0,0,0
true	$\lambda x. y(x(x))(\lambda x.y(xx))$
false	$\lambda x.yx$
not	$\lambda x.y\bar{x}$
and	$\lambda x.z(x\bar{y}y)(\lambda xy.\bar{x})$
or	$\lambda x.y\bar{xy}$
ife	$\lambda x.y\bar{xy}\bar{xy}$
zero	$\lambda xuv.xuv$
succ	$\lambda fx.f(mfx)$
pred	$\lambda mfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)$
add	$\lambda mnfix.m(nfix)(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)m$
sub	$\lambda mnfix.m(nfix)(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)m$
mul	$\lambda mnfix.m(nfix)(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u)m$
iszero	$\lambda z.z(\lambda xyy.y)(\lambda xy.x)$
eq	$\lambda mn.n(\lambda fix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x)$
div	$\lambda mn.n(\lambda mnfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x)(\lambda (nfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x))$
mod	$\lambda (ny.(nx.y(xx))(nx.y(xx)))(\lambda mnfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(\lambda xxy.y)(\lambda xy.x)nm(\lambda mnfix.m(nfix)(\lambda fx.x)(\lambda (mnfix.m(nfix)(\lambda fx.x)(\lambda (f(mnfix.m(nfix)(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))m(mnfix.m(nfix)(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))))(\lambda fx.x)))$
pair	$\lambda fxfx$
fst	$\lambda x.x(\lambda xy.y)$
sec	$\lambda fix.fix$
one	$\lambda fix.f(fx)$
two	$\lambda fix.f(f(fx))$
three	$\lambda fix.f(f(f(fx)))$
five	$\lambda fix.f(f(f(f(fx))))$
var.	λ
var.	λ
iszero n	$(\lambda z.z(\lambda xyy.y)(\lambda xy.x))n$
f (pred n)	$((\lambda mnfix.m(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n)$
mul n (f (pred n))	$(\lambda mnfix.m(nfix)(\lambda gh.h(gf))(\lambda u.x)(\lambda u.u))n$
fac	$(\lambda y.y(\lambda x.(x(\lambda y.y)(\lambda xy.x)))$
fac five, $\beta x24550$	$31 0 (\lambda x.(x(\lambda y.y)(\lambda xy.x)))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 395,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
mul n (f (pred n))
fac
fac five,  $\beta x 25550$ 

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$
- $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 360,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
fac
fac five,  $\beta x 26550$ 

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$
- $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 361,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
mul n f (pred n)
fac
fac five,  $\beta$ x27550
  
```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$
- $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 340,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
fac
fac five,  $\beta$ x27650

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X. M) \in \Lambda$

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
- app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 370,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
mul n (f (pred n))
fac
fac five,  $\beta$ x27750
  
```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
- abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X.M) \in \Lambda$
- $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \xrightarrow{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

1st live, px27630

A crash course in λ -calculus

A crash course in λ -calculus

A crash course in λ -calculus

A crash course in λ -calculus

Mode: 0 | Cursor: 0,0,35 | Item: 31,0 | Nodes Count: 357,0,0,0

Y-combinator

```

true
false
not
and
or
ite
zero
succ
pred
add
sub
mul
iszero
leq
eq
div
mod
pair
fst
sec
one
two
three
five
var.
var.
iszero n
mul n (f (pred n))
fac
fac five,  $\beta$ x28070

```

Create λ -terms by 3 rules (set of λ -terms: Λ)

- var.: a, b, c, x, y, z, \dots (set of variables: V)
- app.: M, N in $\Lambda \Rightarrow (MN)$ in Λ
- abs.: X in V , M in $\Lambda \Rightarrow (\lambda X.M)$ in Λ

$\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X.M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)

- drop outermost parenthesis: $MN = (MN)$
- app. left associative: $MNK = ((MN)K)$
- abs. right associative, only one λ : $\lambda xy.M = (\lambda x.(\lambda y.M))$
- app. takes precedence over abs.: $\lambda x.MN = (\lambda x.(MN))$

A crash course in λ -calculus

inactive, px20075

A crash course in λ -calculus

Create λ -terms by 3 rules (set of λ -terms: **A**)
 • var.: a, b, c, x, y, z, \dots (set of variables: **V**)
 • app.: $M, N \in \Lambda \Rightarrow (MN) \in \Lambda$
 • abs.: $X \in V, M \in \Lambda \Rightarrow (\lambda X. M) \in \Lambda$
 $\Lambda = V \mid (\Lambda\Lambda) \mid (\lambda V.\Lambda)$

β -reduction: $((\lambda X. M)N) \rightarrow_{\beta} M[X := N]$

Notation (guided by Currying of multivariate functions)
 • drop outermost parenthesis: $MN = (MN)$
 • app. left associative: $MNK = ((MN)K)$
 • abs. right associative, only one λ : $\lambda xy. M = (\lambda x. (\lambda y. M))$
 • app. takes precedence over abs.: $\lambda x. MN = (\lambda x. (MN))$

```

(\(x y) x)
(\(x y) x)(\m(\n(m(x.m(\n(g h.h(g f))))(\n(u).x)(\n(u).u))\n(\n(x y.y)(\n(x y.x))
(\n(x y.x)))
\m(\n(m(f x.m(\n(g h.h(g f))))(\n(u).x)(\n(u).u))\m(\n(x y.y)(\n(x y.x))\n(m))
(\n(m(x.m(\n(g h.h(g f))))(\n(u).x)(\n(u).u))\m(\n(x y.y)(\n(x y.x))\n(m))
(\n(m))

```

A crash course in λ -calculus

A crash course in λ -calculus



Alonzo Church
(1903 - 1995)
Lambda Calculus: 1930s



Alan Turing
(1912 - 1954)
Turing Machine: 1936



Alonzo Church
(1903 - 1995)
Lambda Calculus: 1930s



Alan Turing
(1912 - 1954)
Turing Machine: 1936

- Church was Turing's doctoral advisor



Alonzo Church
(1903 - 1995)
Lambda Calculus: 1930s



Alan Turing
(1912 - 1954)
Turing Machine: 1936

- Church was Turing's doctoral advisor
- Church: the λ -calculus is Turing complete



Alonzo Church
(1903 - 1995)
Lambda Calculus: 1930s



Alan Turing
(1912 - 1954)
Turing Machine: 1936

- Church was Turing's doctoral advisor
- Church: the λ -calculus is Turing complete
- Turing: the λ -calculus can be emulated on a Turing machine



Alonzo Church
(1903 - 1995)
Lambda Calculus: 1930s



Alan Turing
(1912 - 1954)
Turing Machine: 1936

- Church was Turing's doctoral advisor
- Church: the λ -calculus is Turing complete
- Turing: the λ -calculus can be emulated on a Turing machine
- **Church-Turing Thesis:** in our universe all "effectively computable functions" can be computed by the λ -calculus/Turing machines

A crash course in type theory

Mode 1 | Cursor: 0,0,0 | Item: 0,0 | Nodes Count: 0,1,0,0,0

var. 0 | 1 a
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49

Create types by 2 rules (set of types: \mathbb{T})
• var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})

The screenshot shows a software interface for type theory development. On the left, there is a code editor window with a list of numbers from 0 to 49. The first line, 'var. 0 | 1 a', is highlighted with a yellow background. The number '1' is also highlighted in yellow. The code editor has a status bar at the top with the text: 'Mode 1 | Cursor: 0,0,0 | Item: 0,0 | Nodes Count: 0,1,0,0,0'. On the right, there is a sidebar titled 'Create types by 2 rules' which includes a list of type variables: 'var.: α, β, γ, δ, ...'. Below this, it says '(set of variables: V)'. The entire interface is enclosed in a light gray border.

A crash course in type theory

Mode 1 | Cursor: 0,0,1 | Items: 1,0 | Nodes Count: 0,2,0,0,0

var.
var.

0 1	a
1 1	b
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	
37	
38	
39	
40	
41	
42	
43	
44	
45	
46	
47	
48	
49	

Create types by 2 rules (set of types: \mathbb{T})
• var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})

A crash course in type theory

```
Mode 1 | Cursor: 0,0,2 | Item: 2,0 | Nodes Count: 0,3,0,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
3|
4|
5|
6|
7|
8|
9|
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create types by 2 rules (set of types: \mathbb{T})
• var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})

A crash course in type theory

```
Mode 1 | Cursor: 0,0,3 | Item: 3,0 | Nodes Count: 0,4,0,0,0
var. 0 | 1 | a
var. 1 | 1 | b
var. 2 | 1 | c
arr. 1, 2 3 | 1 | (b-c)
4 |
5 |
6 |
7 |
8 |
9 |
10 |
11 |
12 |
13 |
14 |
15 |
16 |
17 |
18 |
19 |
20 |
21 |
22 |
23 |
24 |
25 |
26 |
27 |
28 |
29 |
30 |
31 |
32 |
33 |
34 |
35 |
36 |
37 |
38 |
39 |
40 |
41 |
42 |
43 |
44 |
45 |
46 |
47 |
48 |
49 |
```

Create types by 2 rules (set of types: \mathbb{T})
• var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
• arr.: $S, T \in \mathbb{T} \Rightarrow (S \rightarrow T) \in \mathbb{T}$
 $\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$

A crash course in type theory

```
Mode 1 | Cursor: 0,0,4 | Item: 4,0 | Nodes Count: 0,5,0,0
var. 0 | a
var. 1 | b
var. 2 | c
arr. 1, 2 3 | (b-c)
arr. 3, 3 4 | ((b-c)-(b-c))
5 |
6 |
7 |
8 |
9 |
10 |
11 |
12 |
13 |
14 |
15 |
16 |
17 |
18 |
19 |
20 |
21 |
22 |
23 |
24 |
25 |
26 |
27 |
28 |
29 |
30 |
31 |
32 |
33 |
34 |
35 |
36 |
37 |
38 |
39 |
40 |
41 |
42 |
43 |
44 |
45 |
46 |
47 |
48 |
49 |
```

Create types by 2 rules (set of types: \mathbb{T})
• var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
• arr.: $S, T \in \mathbb{T} \Rightarrow (S \rightarrow T) \in \mathbb{T}$
 $\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$

A crash course in type theory

```
Mode 1 | Cursor: 0,0,5 | Item: 5,0 | Nodes Count: 0,6,0,0
var. 0 | a
var. 1 | b
var. 2 | c
arr. 1, 2 3 | (b-c)
arr. 3, 3 4 | ((b-c)-(b-c))
arr. 0, 3 5 | (a-(b-c))
6 |
7 |
8 |
9 |
10 |
11 |
12 |
13 |
14 |
15 |
16 |
17 |
18 |
19 |
20 |
21 |
22 |
23 |
24 |
25 |
26 |
27 |
28 |
29 |
30 |
31 |
32 |
33 |
34 |
35 |
36 |
37 |
38 |
39 |
40 |
41 |
42 |
43 |
44 |
45 |
46 |
47 |
48 |
49 |
```

Create types by 2 rules (set of types: \mathbb{T})
• var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
• arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}
 $\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$

A crash course in type theory

```
Mode 1 | Cursor: 0,0,6 | Item: 6,0 | Nodes Count: 0,7,0,0
var. 0 | a
var. 1 | b
var. 2 | c
arr. 1, 2 3 | (b-c)
arr. 3, 3 4 | ((b-c)-(b-c))
arr. 0, 3 5 | (a-(b-c))
arr. 5, 4 6 | ((a-(b-c))-((b-c)-(b-c)))
```

7 |
8 |
9 |
10 |
11 |
12 |
13 |
14 |
15 |
16 |
17 |
18 |
19 |
20 |
21 |
22 |
23 |
24 |
25 |
26 |
27 |
28 |
29 |
30 |
31 |
32 |
33 |
34 |
35 |
36 |
37 |
38 |
39 |
40 |
41 |
42 |
43 |
44 |
45 |
46 |
47 |
48 |
49 |

Create types by 2 rules (set of types: \mathbb{T})
• var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
• arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}
 $\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$

A crash course in type theory

```
Mode 1 | Cursor: 0,0,6 | Item: 6,0 | Nodes Count: 0,7,0,0
var.          0|1 a
var.          1|1 b
var.          2|1 c
arr. 1, 2    3|1 (b-c)
arr. 3, 3    4|1 ((b-c)-(b-c))
arr. 0, 3    5|1 (a-(b-c))
arr. 5, 4    6|1 ((a-(b-c))-((b-c)-(b-c)))
```

7|
8|
9|
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

A crash course in type theory

```
Mode 1 | Cursor: 0,0,6 | Item: 6,0 | Nodes Count: 0,7,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1, 2 3|1 b-c
arr. 3, 3 4|1 (b-c)-b-c
arr. 0, 3 5|1 a-b-c
arr. 5, 4 6|1 (a-b-c)-(b-c)-b-c
7|
8|
9|
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

A crash course in type theory

```
Mode 2 | Cursor: 0,0,7 | Item: 7,0 | Nodes Count: 0,7,1,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1, 2 3|1 b-c
arr. 3, 3 4|1 (b-c)-b-c
arr. 0, 3 5|1 a-b-c
arr. 5, 4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
8|
9|
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

A crash course in type theory

```
Mode 2 | Cursor: 0,0,8 | Item: 8,0 | Nodes Count: 0,7,2,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
9|
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

A crash course in type theory

```
Mode 2 | Cursor: 0,0,9 | Items: 9,0 | Nodes Count: 0,7,3,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

A crash course in type theory

```
Mode 2 | Cursor: 0,0,9 | Item: 9,0 | Nodes Count: 0,7,3,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement

A crash course in type theory

```
Mode 2 | Cursor: 0,0,9 | Item: 9,0 | Nodes Count: 0,7,3,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T

A crash course in type theory

```
Mode 2 | Cursor: 0,0,9 | Items: 9,0 | Nodes Count: 0,7,3,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement.

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V

A crash course in type theory

```
Mode 2 | Cursor: 0,0,9 | Item: 9,0 | Nodes Count: 0,7,3,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement.

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations

A crash course in type theory

```
Mode 2 | Cursor: 0,0,9 | Items: 9,0 | Nodes Count: 0,7,3,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1, 2 3|1 b-c
arr. 3, 3 4|1 (b-c)-b-c
arr. 0, 3 5|1 a-b-c
arr. 5, 4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
10|
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement.

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

A crash course in type theory

```
Mode 3 | Cursor: 0,0,10 | Item: 10,0 | Nodes Count: 0,7,3,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1, 2 3|1 b-c
arr. 3, 3 4|1 (b-c)-b-c
arr. 0, 3 5|1 a-b-c
arr. 5, 4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
11|
12|
13|
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in \mathbb{T} $\Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement.

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

A crash course in type theory

```
Mode 3 | Cursor: 0,0,11 | Item: 11,0 | Nodes Count: 0,7,3,0,0  
var. 0|1 a  
var. 1|1 b  
var. 2|1 c  
arr. 1, 2 3|1 b-c  
arr. 3, 3 4|1 (b-c)-b-c  
arr. 0, 3 5|1 a-b-c  
arr. 5, 4 6|1 (a-b-c)-(b-c)-b-c  
var. 7|2 x  
var. 8|2 y  
var. 9|2 z  
empty judgement 10|3 ()>[NIL]:[NIL]  
context 11|3 y:b-c>[NIL]:[NIL]  
12|  
13|  
14|  
15|  
16|  
17|  
18|  
19|  
20|  
21|  
22|  
23|  
24|  
25|  
26|  
27|  
28|  
29|  
30|  
31|  
32|  
33|  
34|  
35|  
36|  
37|  
38|  
39|  
40|  
41|  
42|  
43|  
44|  
45|  
46|  
47|  
48|  
49|
```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in \mathbb{T} $\Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement.

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

A crash course in type theory

```
Mode 3 | Cursor: 0,0,12 | Item: 12,0 | Nodes Count: 0,7,3,0,0  
var.  
var.  
var.  
arr. 1, 2  
arr. 3, 3  
arr. 0, 3  
arr. 5, 4  
var.  
var.  
var.  
empty judgement  
context  
context  
0|1 a  
1|1 b  
2|1 c  
3|1 b-c  
4|1 (b-c)-b-c  
5|1 a-b-c  
6|1 (a-b-c)-(b-c)-b-c  
7|2 x  
8|2 y  
9|2 z  
10|3 ()>[NIL] : [NIL]  
11|3 y:b-c>[NIL] : [NIL]  
12|3 y:b-c . x;b>[NIL] : [NIL]  
13|  
14|  
15|  
16|  
17|  
18|  
19|  
20|  
21|  
22|  
23|  
24|  
25|  
26|  
27|  
28|  
29|  
30|  
31|  
32|  
33|  
34|  
35|  
36|  
37|  
38|  
39|  
40|  
41|  
42|  
43|  
44|  
45|  
46|  
47|  
48|  
49|
```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in \mathbb{T} $\Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement.

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

A crash course in type theory

```
Mode 3 | Cursor: 0,0,13 | Item: 13,0 | Nodes Count: 0,7,3,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1, 2 3|1 b-c
arr. 3, 3 4|1 (b-c)-b-c
arr. 0, 3 5|1 a-b-c
arr. 5, 4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL] : [NIL]
judgement context 11|3 y:b-c>[NIL] : [NIL]
context var. 12|3 y:b-c . x;b>[NIL] : [NIL]
var. 13|3 y:b-c . x;b>y:b-c
14|
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in \mathbb{T} $\Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement.

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if
 $X : T$ is declared in Γ

A crash course in type theory

```
Mode 3 | Cursor: 0,0,14 | Item: 14,0 | Nodes Count: 0,7,3,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1, 2 3|1 b-c
arr. 3, 3 4|1 (b-c)-b-c
arr. 0, 3 5|1 a-b-c
arr. 5, 4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c>[NIL]:[NIL]
context 12|3 y:b-c x:b>[NIL]:[NIL]
var. 13|3 y:b-c x:b>y:b-c
var. 14|3 y:b-c x:b>x:b
15|
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in \mathbb{T} $\Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement.

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if
 $X : T$ is declared in Γ

A crash course in type theory

```
Mode 3 | Cursor: 0,0,15 | Item: 15,0 | Nodes Count: 0,7,4,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1, 2 3|1 b-c
arr. 3, 3 4|1 (b-c)-b-c
arr. 0, 3 5|1 a-b-c
arr. 5, 4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c:[NIL]:[NIL]
context 12|3 y:b-c x:b:[NIL]:[NIL]
var. 13|3 y:b-c x:b:y:b-c
var. 14|3 y:b-c x:b:x:b
app. 13, 14 15|3 y:b-c.x:b>yx:c
16|
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|
```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement.

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if
 $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if
 $\Gamma \vdash M : S \rightarrow T$ and
 $\Gamma \vdash N : S$

A crash course in type theory

```

Mode 3 | Cursor: 0,0,16 | Item: 16,0 | Nodes Count: 0,8,5,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1, 2 3|1 b-c
arr. 3, 3 4|1 (b-c)-b-c
arr. 0, 3 5|1 a-b-c
arr. 5, 4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c>[NIL]:[NIL]
context 12|3 y:b-c x:b>[NIL]:[NIL]
var. 13|3 y:b-c x:b>y:b-c
var. 14|3 y:b-c x:b>x:b
app. 13, 14 15|3 y:b-c x:b>yx:c
abs. 15, x 16|3 y:b-c>x:b.yx:b-c
17|
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in \mathbb{T} $\Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement.

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if
 - $\Gamma \vdash M : S \rightarrow T$ and
 - $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if
 - $\Gamma, X : S \vdash M : T$

A crash course in type theory

```

Mode 3 | Cursor: 0,0,17 | Item: 17,0 | Nodes Count: 0,9,6,0,0
var. 1| a
var. 2| b
var. 3| c
arr. 1,2| b-c
arr. 3,3| (b-c)-b-c
arr. 0,3| a-b-c
arr. 5,4| (a-b-c)-(b-c)-b-c
var. 7| x
var. 8| y
var. 9| z
empty judgement
context
context
var. 10| ()>[NIL]:[NIL]
var. 11| y:b-c:[NIL]:[NIL]
var. 12| y:b-c:x:b:[NIL]:[NIL]
var. 13| y:b-c:x:b:y:b-c
var. 14| y:b-c:x:b:x:b
var. 15| y:b-c:x:b>yx:c
var. 16| y:b-c>x:b>yx:b-c
app. 13,14| ()>\y:b-c\>x:b.yx:(b-c)-b-c
abs. 15,x| 
abs. 16,y| 
17| ()>\y:b-c\>x:b.yx:(b-c)-b-c
18|
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement.

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in \mathbb{V}
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if
 - $\Gamma \vdash M : S \rightarrow T$ and
 - $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if
 - $\Gamma, X : S \vdash M : T$

A crash course in type theory

```

Mode 4 | Cursor: 0,0,18 | Item: 18,0 | Nodes Count: 0,9,6,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c:[NIL]:[NIL]
context 12|3 y:b-c x:b:[NIL]:[NIL]
var. 13|3 y:b-c x:b:y:b-c
var. 14|3 y:b-c x:b:x:b
15|3 y:b-c x:b:yx:c
16|3 y:b-c>x:b:yx:b-c
17|3 ()>y:b-c\|x:b.yx:(b-c)-b-c
empty derivation 18|4
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: V)
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = V \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if
 $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if
 $\Gamma \vdash M : S \rightarrow T$ and
 $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if
 $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 4 | Cursor: 0,0,18 | Item: 18,0 | Nodes Count: 0,9,6,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c:[NIL]:[NIL]
context 12|3 y:b-c,x:b:[NIL]:[NIL]
var. 13|3 y:b-c,x:b:y:b-c
var. 14|3 y:b-c,x:b:y:b
app. 13,14 15|3 y:b-c,x:b:yx:c
abs. 15,x 16|3 y:b-c,x:b,yx:b-c
abs. 16,y 17|3 ()>y:b-c\|x:b,yx:(b-c)-b-c
context 18|4 0|y:b-c
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|
49|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in \mathbb{V}
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 4 | Cursor: 0,0,18 | Item: 18,0 | Nodes Count: 0,9,6,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c:[NIL]:[NIL]
context 12|3 y:b-c:x:b:[NIL]:[NIL]
var. 13|3 y:b-c:x:b:y:b-c
var. 14|3 y:b-c:x:b:x:b
app. 13,14 15|3 y:b-c:x:b:yx:c
abs. 15,x 16|3 y:b-c:x:b:yx:b-c
abs. 16,y 17|3 ()>y:b-c\|x:b.yx:(b-c)-b-c
context 18|4 0|y:b-c
context 18|4 1|-x:b
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|
48|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in \mathbb{V}
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 4 | Cursor: 0,0,18 | Item: 18,0 | Nodes Count: 0,9,6,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c:[NIL]:[NIL]
context 12|3 y:b-c:x:b:[NIL]:[NIL]
var. 13|3 y:b-c:x:b:y:b-c
var. 14|3 y:b-c:x:b:x:b
15|3 y:b-c:x:b:yx:c
16|3 y:b-c:x:b:yx:b-c
17|3 ()>y:b-c\|x:b.yx:(b-c)-b-c
app. 13,14 18|4 0|y:b-c
abs. 15,x 1|-x:b
abs. 16,y 2| y:b-c
context 19|
context 20|
var. 21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|
47|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: V)
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = V \mid (T \rightarrow T)$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 4 | Cursor: 0,0,18 | Item: 18,0 | Nodes Count: 0,9,6,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c:[NIL]:[NIL]
context 12|3 y:b-c x:b:[NIL]:[NIL]
var. 13|3 y:b-c x:b:y:b-c
var. 14|3 y:b-c x:b:x:b
15|3 y:b-c x:b>yx:c
16|3 y:b-c>x:b>yx:b-c
17|3 ()>y:b-c\>x:b>yx:(b-c)-b-c
app. 13,14 18|4 0|y:b-c
abs. 15,x 1|-x:b
abs. 16,y 2| y:b-c
context 3| x:b
context
var. 19|
var. 20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|
46|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: V)
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = V \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 4 | Cursor: 0,0,18 | Item: 18,0 | Nodes Count: 0,9,7,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c:[NIL]:[NIL]
context 12|3 y:b-c x:b:[NIL]:[NIL]
var. 13|3 y:b-c x:b:y:b-c
var. 14|3 y:b-c x:b:x:b
15|3 y:b-c x:b:yx:c
16|3 y:b-c x:b:yx:b-c
app. 13,14 17|3 ()>y:b-c\|x:b.yx:(b-c)-b-c
abs. 15,x 18|4 0|y:b-c
abs. 16,y 1| x:b
context 2| y:b-c
context 3| x:b
var. 4| yx:c
app. 2,3
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|
45|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement.

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in \mathbb{V}
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 4 | Cursor: 0,0,18 | Item: 18,0 | Nodes Count: 0,10,8,0,0
var. 1| a
var. 1| b
var. 2| c
arr. 1,2| b-c
arr. 3,3| (b-c)-b-c
arr. 0,3| a-b-c
arr. 5,4| (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement
context
context
var. 10|3 ()>[NIL]:[NIL]
var. 11|3 y:b-c:[NIL]:[NIL]
context
var. 12|3 y:b-c;x:b:[NIL]:[NIL]
var. 13|3 y:b-c;x:b:y:b-c
app. 13,14|4 y:b-c;x:b:yx:c
abs. 15,x|5 y:b-c>x:b:yx:b-c
abs. 16,y|6 y:b-c>x:b:yx:(b-c)-b-c
context
context
var. 17|3 ()>y:b-c\|x:b:yx:(b-c)-b-c
app. 2,3|8 y:b-c
abs. 4,x|9 y:b-c
18|4 0| y:b-c
1| x:b
2| y:b-c
3| x:b
4| yx:c
5| xx:b:yx:b-c
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|
44|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \beta) \rightarrow \gamma)$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 4 | Cursor: 0,0,18 | Item: 18,0 | Nodes Count: 0,11,9,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c:[NIL]:[NIL]
context 12|3 y:b-c:x:b:[NIL]:[NIL]
var. 13|3 y:b-c:x:b:y:b-c
var. 14|3 y:b-c:x:b:y:b
15|3 y:b-c:x:b>yx:c
16|3 y:b-c>x:b>yx:b-c
17|3 ()>y:b-c\>x:b>yx:(b-c)-b-c
app. 13,14 18|4 0|y:b-c
abs. 15,x 1|x:b
abs. 16,y 2|y:b-c
context 3|x:b
context 4|y:b-c
app. 2,3 5|y:b-c\>x:b>yx:(b-c)-b-c
abs. 4,x 6|y:b-c\>x:b>yx:(b-c)-b-c
abs. 5,y
19|
20|
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta \rightarrow \beta = \alpha \rightarrow (\beta \rightarrow \beta)$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow \beta) \rightarrow \gamma$

Statement, declaration, context, judgement:

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in \mathbb{V}
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 1 | Cursor: 0,0,25 | Item: 19,0 | Nodes Count: 0,12,9,0,0
var. 1| a
var. 2| b
var. 3| c
arr. 1,2| b-c
arr. 3,3| (b-c)-b-c
arr. 0,3| a-b-c
arr. 5,4| (a-b-c)-(b-c)-b-c
var. 7| x
var. 8| y
var. 9| z
empty judgement
context
context
var. 10| ()>[NIL]:[NIL]
var. 11| y:b-c:[NIL]:[NIL]
var. 12| y:b-c:x:b:[NIL]:[NIL]
var. 13| y:b-c:x:b:y:b-c
var. 14| y:b-c:x:b:x:b
var. 15| y:b-c:x:b>yx:c
var. 16| y:b-c>x:b>yx:b-c
var. 17| ()>y:b-c\>x:b>yx:(b-c)-b-c
app. 13,14| 0| y:b-c
abs. 15,x| 1| x:b
abs. 16,y| 2| y:b-c
context
context
var. 18| 3| x:b
var. 19| 4| yx:c
app. 2,3| 5| x:b>yx:b-c
abs. 4,x| 6| yx:b-c\>x:b>yx:(b-c)-b-c
abs. 5,y| 19| a-c
arr. 0,2| 20|
var. 21|
var. 22|
var. 23|
var. 24|
var. 25|
var. 26|
var. 27|
var. 28|
var. 29|
var. 30|
var. 31|
var. 32|
var. 33|
var. 34|
var. 35|
var. 36|
var. 37|
var. 38|
var. 39|
var. 40|
var. 41|
var. 42|
var. 43|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: V)
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = V \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \beta) \rightarrow \gamma)$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 1 | Cursor: 0,0,26 | Item: 20,0 | Nodes Count: 0,13,9,0,0
var. 1| a
var. 2| b
var. 3| c
arr. 1, 2| b-c
arr. 3, 3| (b-c)-b-c
arr. 0, 3| a-b-c
arr. 5, 4| (a-b-c)-(b-c)-b-c
var. 7| x
var. 8| y
var. 9| z
empty judgement 10| ()>[NIL]:[NIL]
context 11| y:b-c:[NIL]:[NIL]
context 12| y:b-c:x:b:[NIL]:[NIL]
context 13| y:b-c:x:b:y:b-c
context 14| y:b-c:x:b:x:b
context 15| y:b-c:x:b>yx:c
context 16| y:b-c>x:b>yx:b-c
context 17| ()>y:b-c\>x:b>yx:(b-c)-b-c
context 18| y:b-c
context 19| a-c
context 20| (a-c)-a
21|
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \beta) \rightarrow \gamma)$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 4 | Cursor: 0,0,27 | Item: 21,0 | Nodes Count: 0,13,9,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c:[NIL]:[NIL]
context 12|3 y:b-c:x:b:[NIL]:[NIL]
context 13|3 y:b-c:x:b:y:b-c
context 14|3 y:b-c:x:b:y:b
context 15|3 y:b-c:x:b>yx:c
context 16|3 y:b-c>x:b>yx:b-c
context 17|3 ()>y:b-c\>x:b\>yx:(b-c)-b-c
context 18|4 0|y:b-c
context 1|1 x:b
context 2|1 y:b-c
context 3|1 x:b
context 4|1 yx:c
context 5|1 sx:b>yx:b-c
context 6|1 y:y:b-c\>x:b\>yx:(b-c)-b-c
app. 13,14 19|1 a-c
abs. 15,x 20|1 (a-c)-a
abs. 16,y
abs. 17,y
abs. 18,y
abs. 19,y
arr. 19,0
empty derivation 21|4
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|
43|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta \rightarrow \beta = \alpha \rightarrow (\beta \rightarrow \beta)$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow \beta) \rightarrow \gamma$

Statement, declaration, context, judgement

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in \mathbb{V}
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 4 | Cursor: 0,0,27 | Item: 21,0 | Nodes Count: 0,13,9,0,0
var. 1| a
var. 2| b
var. 3| c
arr. 1,2| b-c
arr. 3,3| (b-c)-b-c
arr. 0,3| a-b-c
arr. 5,4| (a-b-c)-(b-c)-b-c
var. 7| x
var. 8| y
var. 9| z
empty judgement
context
context
var. 10| ()>[NIL]:[NIL]
var. 11| y:b-c:[NIL]:[NIL]
var. 12| y:b-c:x:b:[NIL]:[NIL]
var. 13| y:b-c:x:b:y:b-c
var. 14| y:b-c:x:b:y:b
var. 15| y:b-c:x:b>yx:c
var. 16| y:b-c>x:b>yx:b-c
var. 17| ()>y:b-c\>x:b>yx:(b-c)-b-c
app. 13,14| 0|y:b-c
abs. 15,x| 1|x:b
abs. 16,y| 2|y:b-c
context
context
var. 18| 3|x:b
var. 19| 4|yx:c
var. 20| 5|sx:b>yx:b-c
var. 21| 6|sy:b-c\>x:b>yx:(b-c)-b-c
app. 2,3| a-c
abs. 4,x| (a-c)-a
abs. 5,y| 0|x:(a-c)-a
arr. 0,2| 22|
arr. 19,0| 23|
context| 24|
| 25|
| 26|
| 27|
| 28|
| 29|
| 30|
| 31|
| 32|
| 33|
| 34|
| 35|
| 36|
| 37|
| 38|
| 39|
| 40|
| 41|
| 42|
| 43|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 4 | Cursor: 0,0,27 | Item: 21,0 | Nodes Count: 0,13,9,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c:[NIL]:[NIL]
context 12|3 y:b-c:x:b:[NIL]:[NIL]
context 13|3 y:b-c:x:b:y:b-c
context 14|3 y:b-c:x:b:x:b
context 15|3 y:b-c:x:b>yx:c
context 16|3 y:b-c>x:b>yx:b-c
context 17|3 ()>y:b-c\>x:b\>yx:(b-c)-b-c
context 18|4 0|y:b-c
context 1|1 x:b
context 2|1 y:b-c
context 3|1 x:b
context 4|1 x:b
context 5|1 y:b-c
context 6|1 y:b-c\>x:b\>yx:(b-c)-b-c
context 19|1 a-c
context 20|1 (a-c)-a
context 21|4 0|x:(a-c)-a
context 1|1 y:a-c
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|
42|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta \rightarrow \beta = \alpha \rightarrow (\beta \rightarrow \beta)$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow \beta) \rightarrow \gamma$

Statement, declaration, context, judgement

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 4 | Cursor: 0,0,27 | Item: 21,0 | Nodes Count: 0,13,9,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c:[NIL]:[NIL]
context 12|3 y:b-c x:b:[NIL]:[NIL]
context 13|3 y:b-c x:b:y:b-c
context 14|3 y:b-c x:b:y:b
context 15|3 y:b-c x:b>yx:c
context 16|3 y:b-c>x:b>yx:b-c
context 17|3 ()>y:b-c\>x:b\>yx:(b-c)-b-c
context 18|4 0|y:b-c
context 1|y:b
context 2|y:b-c
context 3|x:b
context 4|y:c
context 5|sx:b>yx:b-c
context 6|sy:b-c\>x:b\>yx:(b-c)-b-c
context 19|1 a-c
context 20|1 (a-c)-a
context 21|4 0|x:(a-c)-a
context 1|-y:a-c
context 2|--z:b
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|
41|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: V)
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = V \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement.

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 4 | Cursor: 0,0,27 | Item: 21,0 | Nodes Count: 0,13,9,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c:[NIL]:[NIL]
context 12|3 y:b-c:x:b:[NIL]:[NIL]
context 13|3 y:b-c:x:b:y:b-c
context 14|3 y:b-c:x:b:y:b
context 15|3 y:b-c:x:b>yx:c
context 16|3 y:b-c>x:b>yx:b-c
context 17|3 ()>y:b-c\>x:b>yx:(b-c)-b-c
context 18|4 0|y:b-c
context 1|y:b
context 2|y:b-c
context 3|x:b
context 4|y:c
context 5|yx:c
context 6|yx:b>yx:b-c
context 7|yx:b-c\>x:b>yx:(b-c)-b-c
context 19|1 a-c
context 20|1 (a-c)-a
context 21|4 0|x:(a-c)-a
context 1|y:a-c
context 2|-z:b
context 3|x:(a-c)-a
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|
39|
40|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: V)
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = V \mid (T \rightarrow T)$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement.

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 4 | Cursor: 0,0,27 | Item: 21,0 | Nodes Count: 0,13,9,0,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c:[NIL]:[NIL]
context 12|3 y:b-c,x:b:[NIL]:[NIL]
var. 13|3 y:b-c,x:b:y:b-c
var. 14|3 y:b-c,x:b:y:b
15|3 y:b-c,x:b>yx:c
16|3 y:b-c>x:b>yx:b-c
17|3 ()>y:b-c\>x:b>yx:(b-c)-b-c
app. 13,14 18|4 0|y:b-c
abs. 15, x 1|>x:b
abs. 16, y 2|y:b-c
context 3|x:b
context 4|x:b
var. 5|yx:c
var. 6|yx:b-c\>x:b>yx:(b-c)-b-c
app. 2,3 19|1 a-c
abs. 4, x 20|1 (a-c)-a
abs. 5, y 21|4 0|x:(a-c)-a
arr. 0,2 1|-y:a-c
arr. 19,0 2|-z:b
context 3|x:(a-c)-a
context 4|y:a-c
var. 22|
var. 23|
var. 24|
var. 25|
var. 26|
var. 27|
var. 28|
var. 29|
var. 30|
var. 31|
var. 32|
var. 33|
var. 34|
var. 35|
var. 36|
var. 37|
var. 38|
var. 39|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: V)
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = V \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 4 | Cursor: 0,0,27 | Item: 21,0 | Nodes Count: 0,13,10,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c:[NIL]:[NIL]
context 12|3 y:b-c:x:b:[NIL]:[NIL]
context 13|3 y:b-c:x:b:y:b-c
context 14|3 y:b-c:x:b:y:b
app. 13,14 15|3 y:b-c:x:b>yx:c
abs. 15, x 16|3 y:b-c>x:b>yx:b-c
abs. 16, y 17|3 ()>y:b-c\>x:b>yx:(b-c)-b-c
context 18|4 0|y:b-c
context 1|y:b
context 2|y:b-c
context 3|x:b
context 4|x:b
context 5|yx:c
context 6|yx:b-c\>x:b>yx:(b-c)-b-c
app. 2,3 19|1 a-c
abs. 4, x 20|1 (a-c)-a
abs. 5, y 21|4 0|x:(a-c)-a
arr. 0,2 1|-y:a-c
arr. 19,0 2|-z:b
context 3|x:(a-c)-a
context 4|y:a-c
app. 3,4 5|xy:a
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|
38|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in \mathbb{V}
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 4 | Cursor: 0,0,27 | Item: 21,0 | Nodes Count: 0,13,11,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c:[NIL]:[NIL]
context 12|3 y:b-c:x:b:[NIL]:[NIL]
context 13|3 y:b-c:x:b:y:b-c
context 14|3 y:b-c:x:b:y:b
context 15|3 y:b-c:x:b>yx:c
context 16|3 y:b-c>x:b>yx:b-c
context 17|3 ()>y:b-c\>x:b>yx:(b-c)-b-c
context 18|4 0|y:b-c
context 1|y:b
context 2|y:b-c
context 3|x:b
context 4|yx:c
context 5|sx:b>yx:b-c
context 6|sy:b-c\>x:b>yx:(b-c)-b-c
app. 2,3 19|1 a-c
abs. 4,x 20|1 (a-c)-a
arr. 5,y 21|4 0|x:(a-c)-a
arr. 0,2 1|-y:a-c
arr. 19,0 2|-z:b
context 3|x:(a-c)-a
context 4|y:a-c
context 5|xy:a
context 6|y(xy):c
app. 3,4 22|
app. 4,5 23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|
37|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: \mathbb{V})
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = \mathbb{V} \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in \mathbb{V}
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 4 | Cursor: 0,0,27 | Item: 21,0 | Nodes Count: 0,14,12,0
var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c:[NIL]:[NIL]
context 12|3 y:b-c:x:b:[NIL]:[NIL]
var. 13|3 y:b-c:x:b:y:b-c
var. 14|3 y:b-c:x:b:y:b
15|3 y:b-c:x:b>yx:c
16|3 y:b-c>x:b>yx:b-c
17|3 ()>y:b-c\>x:b>yx:(b-c)-b-c
app. 13,14 18|4 0|y:b-c
abs. 15, x 1| x:b
abs. 16, y 2| y:b-c
context 3| x:b
context 4| yx:c
var. 5| x:b-c\>x:b>yx:(b-c)-b-c
var. 6| y:b-c\>x:b>yx:(b-c)-b-c
app. 2,3 19|1 a-c
abs. 4, x 20|1 (a-c)-a
abs. 5, y 21|4 0|x:(a-c)-a
arr. 0,2 1| y:a-c
arr. 19,0 2| -z:b
context 3| x:(a-c)-a
context 4| y:a-c
var. 5| xy:a
var. 6| y(xy):c
abs. 6, z 7| \z:b.y(xy):b-c
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|
35|
36|

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: V)
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = V \mid (\mathbb{T} \rightarrow \mathbb{T})$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

```

Mode 4 | Cursor: 0,0,27 | Item: 21,0 | Nodes Count: 0,15,13,0,0
var. 1| a
var. 2| b
var. 3| c
arr. 1,2| b-c
arr. 3,3| (b-c)-b-c
arr. 0,3| a-b-c
arr. 5,4| (a-b-c)-(b-c)-b-c
var. 7| x
var. 8| y
var. 9| z
empty judgement
context
context
var. 10| ()>[NIL]:[NIL]
var. 11| y:b-c:[NIL]:[NIL]
var. 12| y:b-c:x:b:[NIL]:[NIL]
var. 13| y:b-c:x:b:y:b-c
var. 14| y:b-c:x:b:y:b
var. 15| y:b-c:x:b>yx:c
var. 16| y:b-c>x:b>yx:b-c
var. 17| ()>y:b-c\>x:b>yx:(b-c)-b-c
app. 13,14| y:b-c
abs. 15,x| y:b-c
abs. 16,y| y:b-c
context
context
var. 18| y:b-c
var. 19| y:b-c
app. 2,3| y:b-c
abs. 4,x| y:b-c
abs. 5,y| y:b-c
arr. 0,2| y:b-c
arr. 19,0| y:b-c
context
context
context
var. 20| (a-c)-a
var. 21| (a-c)-a
var. 22| a-c
var. 23| a-c
var. 24| a-c
var. 25| a-c
var. 26| a-c
var. 27| a-c
var. 28| a-c
var. 29| a-c
var. 30| a-c
var. 31| a-c
var. 32| a-c
var. 33| a-c
var. 34| a-c
var. 35| a-c

```

Create types by 2 rules (set of types: \mathbb{T})

- var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: V)
- arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}

$$\mathbb{T} = V \mid (T \rightarrow T)$$

Notation (again guided by Currying of multivariate functions)

- drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
- arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement

- a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
read: M is of type T
- a declaration is a statement $X : T$ where X in V
- a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
- a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
read: in the context Γ , M is of type T

Derivation rules (for valid judgements)

- var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
- app.: $\Gamma \vdash MN : T$ if $\Gamma \vdash M : S \rightarrow T$ and $\Gamma \vdash N : S$
- abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if $\Gamma, X : S \vdash M : T$

Flag format for derivations

- open nested flags for every declaration in context
- flag contains declaration
- flagpole indicates scope of declaration

A crash course in type theory

Mode 4 | Cursor: 0,0,27 | Item: 21,0 | Nodes Count: 0,16,14,0

```

var. 0|1 a
var. 1|1 b
var. 2|1 c
arr. 1,2 3|1 b-c
arr. 3,3 4|1 (b-c)-b-c
arr. 0,3 5|1 a-b-c
arr. 5,4 6|1 (a-b-c)-(b-c)-b-c
var. 7|2 x
var. 8|2 y
var. 9|2 z
empty judgement 10|3 ()>[NIL]:[NIL]
context 11|3 y:b-c:[NIL]:[NIL]
context 12|3 y:b-c:x:b:[NIL]:[NIL]
var. 13|3 y:b-c:x:b:y:b-c
var. 14|3 y:b-c:x:b:y:b
15|3 y:b-c:x:b>yx:c
16|3 y:b-c>x:b>yx:b-c
17|3 ()>y:b-c\>x:b>yx:(b-c)-b-c
app. 13,14 18|4 0|y:b-c
abs. 15, x 1|>x:b
abs. 16, y 2|>y:b-c
context 3|>x:b
context 4|>yx:c
var. 5|>xy:a
var. 6|>y(xy):c
app. 2,3 7|>z:b>y(xy):b-c
abs. 4, x 8|>y:a-c,>x:b>y(xy):(a-c)-b-c
abs. 5, y 9|>x:(a-c)-a,>y:a-c,>z:b>y(xy):((a-c)-a)-(a-c)-b-c
arr. 0,2
arr. 19,0
context
context
context
var.
var.
app. 3,4
app. 4,5
abs. 6, z
abs. 7, y
abs. 8, x
19|1 a-c
20|1 (a-c)-a
21|4 0|x:(a-c)-a
1|-y:a-c
2|-z:b
3|x:(a-c)-a
4|y:a-c
5|xy:a
6|y(xy):c
7|z:b>y(xy):b-c
8|y:a-c,>x:b>y(xy):(a-c)-b-c
9|x:(a-c)-a,>y:a-c,>z:b>y(xy):((a-c)-a)-(a-c)-b-c
22|
23|
24|
25|
26|
27|
28|
29|
30|
31|
32|
33|
34|

```

Create types by 2 rules (set of types: \mathbb{T})
 • var.: $\alpha, \beta, \gamma, \delta, \dots$ (set of variables: V)
 • arr.: S, T in $\mathbb{T} \Rightarrow (S \rightarrow T)$ in \mathbb{T}
 $\mathbb{T} = V \mid (\mathbb{T} \rightarrow \mathbb{T})$

Notation (again guided by Currying of multivariate functions)
 • drop outermost parenthesis: $\alpha \rightarrow \beta = \alpha \rightarrow \beta$
 • arr. right associative: $\alpha \rightarrow \beta \rightarrow \gamma = (\alpha \rightarrow (\beta \rightarrow \gamma))$

Statement, declaration, context, judgement:
 • a statement is of the form: $M : T$ where M in Λ and T in \mathbb{T}
 read: M is of type T
 • a declaration is a statement $X : T$ where X in V
 • a context is a list $X_1 : T_1, \dots, X_n : T_n$ of declarations
 • a judgement is of the form $\Gamma \vdash M : T$ where Γ is a context
 read: in the context Γ , M is of type T

Derivation rules (for valid judgements)
 • var.: $\Gamma \vdash X : T$ if $X : T$ is declared in Γ
 • app.: $\Gamma \vdash MN : T$ if
 $\Gamma \vdash M : S \rightarrow T$ and
 $\Gamma \vdash N : S$
 • abs.: $\Gamma \vdash \lambda X : S. M : S \rightarrow T$ if
 $\Gamma, X : S \vdash M : T$

Flag format for derivations
 • open nested flags for every declaration in context
 • flag contains declaration
 • flagpole indicates scope of declaration

(a) $y : \mathcal{L} \rightarrow \mathcal{B}$

(b) $z : \mathcal{L}$

(1) $y : \mathcal{L} \rightarrow \mathcal{B}$

(2) $z : \mathcal{L}$

(3) $yz : \mathcal{B}$

(4) $\lambda z : \mathcal{L}. yz : \mathcal{L} \rightarrow \mathcal{B}$

(5) $\lambda y : \mathcal{L} \rightarrow \mathcal{B}. \lambda z : \mathcal{L}. yz : (\mathcal{L} \rightarrow \mathcal{B}) \rightarrow \mathcal{L} \rightarrow \mathcal{B}$

(var) on (a)

(var) on (b)

(app) on (1),(2)

(abst) on (3)

(abst) on (4)

(a) $x : (\mathcal{L} \rightarrow \mathcal{Y}) \rightarrow \mathcal{L}$

(b) $y : \mathcal{L} \rightarrow \mathcal{Y}$

(c) $z : \mathcal{B}$

(v) $xy : \mathcal{L}$

(w) $y(xy) : \mathcal{Y}$

(x) $\lambda z : \mathcal{B}. y(xy) : \mathcal{B} \rightarrow \mathcal{Y}$

(y) $\lambda y : \mathcal{L} \rightarrow \mathcal{Y}. \lambda z : \mathcal{B}. y(xy) : (\mathcal{L} \rightarrow \mathcal{Y}) \rightarrow \mathcal{B} \rightarrow \mathcal{Y}$

(z) $\lambda x : (\mathcal{L} \rightarrow \mathcal{Y}) \rightarrow \mathcal{L}. \lambda y : \mathcal{L} \rightarrow \mathcal{Y}. \lambda z : \mathcal{B}. y(xy) : ((\mathcal{L} \rightarrow \mathcal{Y}) \rightarrow \mathcal{L}) \rightarrow (\mathcal{L} \rightarrow \mathcal{Y}) \rightarrow \mathcal{B} \rightarrow \mathcal{Y}$

(app) on (a),(b)

(app) on (b),(v)

(abst) on (w)

(abst) on (x)

(abst) on (y)

0 | $y : b - c$
1 | $-x : b$
2 | $y : b - c$
3 | $x : b$
4 | $yx : c$

5 | $\backslash x : b. yx : b - c$
6 | $\backslash y : b - c. \backslash x : b. yx : (b - c) - b - c$

0 | $x : (a - c) - a$
1 | $-y : a - c$
2 | $--z : b$
3 | $x : (a - c) - a$
4 | $y : a - c$
5 | $xy : a$
6 | $y(xy) : c$
7 | $\backslash z : b. y(xy) : b - c$
8 | $\backslash y : a - c. \backslash z : b. y(xy) : (a - c) - b - c$
9 | $\backslash x : (a - c) - a. \backslash y : a - c. \backslash z : b. y(xy) : ((a - c) - a) - (a - c) - b - c$

- (a) $y : \mathcal{L} \rightarrow \mathcal{B}$
- (b) $z : \mathcal{L}$
- (1) $y : \mathcal{L} \rightarrow \mathcal{B}$
- (2) $z : \mathcal{L}$
- (3) $yz : \mathcal{B}$
- (4) $\lambda z : \mathcal{L}. yz : \mathcal{L} \rightarrow \mathcal{B}$
- (5) $\lambda y : \mathcal{L} \rightarrow \mathcal{B}. \lambda z : \mathcal{L}. yz : (\mathcal{L} \rightarrow \mathcal{B}) \rightarrow \mathcal{L} \rightarrow \mathcal{B}$

- (a) $y : \mathcal{L} \rightarrow \mathcal{B}$
- (b) $z : \mathcal{L}$
- (1) $y : \mathcal{L} \rightarrow \mathcal{B}$
- (2) $z : \mathcal{L}$
- (3) $yz : \mathcal{B}$
- (4) $\lambda z : \mathcal{L}. yz : \mathcal{L} \rightarrow \mathcal{B}$
- (5) $\lambda y : \mathcal{L} \rightarrow \mathcal{B}. \lambda z : \mathcal{L}. yz : (\mathcal{L} \rightarrow \mathcal{B}) \rightarrow \mathcal{L} \rightarrow \mathcal{B}$

The Curry-Howard Isomorphism

- (a) $y : \mathcal{L} \rightarrow \mathcal{B}$
- (b) $z : \mathcal{L}$
- (1) $y : \mathcal{L} \rightarrow \mathcal{B}$
- (2) $z : \mathcal{L}$
- (3) $yz : \mathcal{B}$
- (4) $\lambda z : \mathcal{L}. yz : \mathcal{L} \rightarrow \mathcal{B}$
- (5) $\lambda y : \mathcal{L} \rightarrow \mathcal{B}. \lambda z : \mathcal{L}. yz : (\mathcal{L} \rightarrow \mathcal{B}) \rightarrow \mathcal{L} \rightarrow \mathcal{B}$

The Curry-Howard Isomorphism

- aka PAT: Propositions As Types
Proofs As Terms

- (a) $y : \mathcal{L} \rightarrow \mathcal{B}$
- (b) $\boxed{z : \mathcal{L}}$
- (1) $y : \mathcal{L} \rightarrow \mathcal{B}$
- (2) $z : \mathcal{L}$
- (3) $yz : \mathcal{B}$
- (4) $\lambda z : \mathcal{L}. yz : \mathcal{L} \rightarrow \mathcal{B}$
- (5) $\lambda y : \mathcal{L} \rightarrow \mathcal{B}. \lambda z : \mathcal{L}. yz : (\mathcal{L} \rightarrow \mathcal{B}) \rightarrow \mathcal{L} \rightarrow \mathcal{B}$

The Curry-Howard Isomorphism

- aka PAT: Propositions As Types
Proofs As Terms
- interpret context as logical assumptions and \rightarrow as \Rightarrow

- (a) $y : \mathcal{L} \rightarrow \mathcal{B}$
- (b) $\boxed{z : \mathcal{L}}$
- (1) $y : \mathcal{L} \rightarrow \mathcal{B}$
- (2) $z : \mathcal{L}$
- (3) $yz : \mathcal{B}$
- (4) $\lambda z : \mathcal{L}. yz : \mathcal{L} \rightarrow \mathcal{B}$
- (5) $\lambda y : \mathcal{L} \rightarrow \mathcal{B}. \lambda z : \mathcal{L}. yz : (\mathcal{L} \rightarrow \mathcal{B}) \rightarrow \mathcal{L} \rightarrow \mathcal{B}$

The Curry-Howard Isomorphism

- aka PAT: Propositions As Types
Proofs As Terms
- interpret context as logical assumptions and \rightarrow as \Rightarrow
 $(\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta$ becomes $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$

- (a) $y : \mathcal{L} \rightarrow \mathcal{B}$
- (b) $\boxed{z : \mathcal{L}}$
- (1) $y : \mathcal{L} \rightarrow \mathcal{B}$
- (2) $z : \mathcal{L}$
- (3) $yz : \mathcal{B}$
- (4) $\lambda z : \mathcal{L}. yz : \mathcal{L} \rightarrow \mathcal{B}$
- (5) $\lambda y : \mathcal{L} \rightarrow \mathcal{B}. \lambda z : \mathcal{L}. yz : (\mathcal{L} \rightarrow \mathcal{B}) \rightarrow \mathcal{L} \rightarrow \mathcal{B}$

The Curry-Howard Isomorphism

- aka PAT: Propositions As Types
Proofs As Terms
- interpret context as logical assumptions and \rightarrow as \Rightarrow
 $(\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta$ becomes $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$
 $\lambda y : \alpha \rightarrow \beta. \lambda z : \alpha. yz$ becomes a proof of $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$ (in the empty context)

- (a) $y : \mathcal{L} \rightarrow \mathcal{B}$
- (b) $z : \mathcal{L}$
- (1) $y : \mathcal{L} \rightarrow \mathcal{B}$
- (2) $z : \mathcal{L}$
- (3) $yz : \mathcal{B}$
- (4) $\lambda z : \mathcal{L}. yz : \mathcal{L} \rightarrow \mathcal{B}$
- (5) $\lambda y : \mathcal{L} \rightarrow \mathcal{B}. \lambda z : \mathcal{L}. yz : (\mathcal{L} \rightarrow \mathcal{B}) \rightarrow \mathcal{L} \rightarrow \mathcal{B}$

$$\begin{array}{c}
 \text{(var)} \quad \boxed{\Gamma \vdash X : T \text{ if } X : T \in \Gamma} \\
 \text{(app)} \quad \frac{\Gamma \vdash A : S \rightarrow T \quad \Gamma \vdash B : S}{\Gamma \vdash AB : T} \leftrightarrow \frac{A \Rightarrow B \quad A}{B} \text{ (}\Rightarrow\text{-elim)} \\
 \text{(abst)} \quad \frac{\Gamma, X : S \vdash A : T}{\Gamma \vdash \lambda X : S. A : S \rightarrow T} \leftrightarrow \boxed{\begin{array}{c} \text{Assume } A \\ \vdots \\ B \\ A \Rightarrow B \end{array}} \text{ (}\Rightarrow\text{-intro)}
 \end{array}$$

The Curry-Howard Isomorphism

- aka PAT: Propositions As Types
Proofs As Terms
- interpret context as logical assumptions and \rightarrow as \Rightarrow
 $(\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta$ becomes $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$
 $\lambda y : \alpha \rightarrow \beta. \lambda z : \alpha. yz$ becomes a proof of $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$ (in the empty context)

(a) $y : \mathcal{L} \rightarrow \mathcal{B}$ (b) $z : \mathcal{L}$ (1) $y : \mathcal{L} \rightarrow \mathcal{B}$ (2) $z : \mathcal{L}$ (3) $yz : \mathcal{B}$ (4) $\lambda z : \mathcal{L}. yz : \mathcal{L} \rightarrow \mathcal{B}$ (5) $\lambda y : \mathcal{L} \rightarrow \mathcal{B}. \lambda z : \mathcal{L}. yz : (\mathcal{L} \rightarrow \mathcal{B}) \rightarrow \mathcal{L} \rightarrow \mathcal{B}$ (var) $\Gamma \vdash X : T$ if $X : T \in \Gamma$ (app) $\frac{\Gamma \vdash A : S \rightarrow T \quad \Gamma \vdash B : S}{\Gamma \vdash AB : T} \leftrightarrow \frac{A \Rightarrow B \quad A}{B} (\Rightarrow\text{-elim})$ (abst) $\frac{\Gamma, X : S \vdash A : T}{\Gamma \vdash \lambda X : S. A : S \rightarrow T} \leftrightarrow \boxed{\begin{array}{c} \text{Assume } A \\ \vdots \\ B \\ A \Rightarrow B \end{array}} (\Rightarrow\text{-intro})$

The Curry-Howard Isomorphism

- aka PAT: Propositions As Types
Proofs As Terms
- interpret context as logical assumptions and \rightarrow as \Rightarrow
 $(\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta$ becomes $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$
 $\lambda y : \alpha \rightarrow \beta. \lambda z : \alpha. yz$ becomes a proof of $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$ (in the empty context)
- a proposition is true if and only if the corresponding type is inhabited
(type inhabited: term of given type exists)

- (a) $y : \mathcal{L} \rightarrow \mathcal{B}$
- (b) $\boxed{z : \mathcal{L}}$
- (1) $y : \mathcal{L} \rightarrow \mathcal{B}$
- (2) $z : \mathcal{L}$
- (3) $yz : \mathcal{B}$
- (4) $\lambda z : \mathcal{L}. yz : \mathcal{L} \rightarrow \mathcal{B}$
- (5) $\lambda y : \mathcal{L} \rightarrow \mathcal{B}. \lambda z : \mathcal{L}. yz : (\mathcal{L} \rightarrow \mathcal{B}) \rightarrow \mathcal{L} \rightarrow \mathcal{B}$

$$\begin{array}{c}
 \text{(var)} \quad \boxed{\Gamma \vdash X : T \text{ if } X : T \in \Gamma} \\
 \text{(app)} \quad \frac{\Gamma \vdash A : S \rightarrow T \quad \Gamma \vdash B : S}{\Gamma \vdash AB : T} \leftrightarrow \frac{A \Rightarrow B \quad A}{B} \text{ (}\Rightarrow\text{-elim)} \\
 \text{(abst)} \quad \frac{\Gamma, X : S \vdash A : T}{\Gamma \vdash \lambda X : S. A : S \rightarrow T} \leftrightarrow \boxed{\begin{array}{c} \text{Assume } A \\ \vdots \\ B \\ A \Rightarrow B \end{array}} \text{ (}\Rightarrow\text{-intro)}
 \end{array}$$

The Curry-Howard Isomorphism

- aka PAT: Propositions As Types
Proofs As Terms
- interpret context as logical assumptions and \rightarrow as \Rightarrow
 $(\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta$ becomes $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$
 $\lambda y : \alpha \rightarrow \beta. \lambda z : \alpha. yz$ becomes a proof of $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$ (in the empty context)
- a proposition is true if and only if the corresponding type is inhabited
(type inhabited: term of given type exists)
- so far: only propositions and \Rightarrow modelled by type theory

- (a) $y : \mathcal{L} \rightarrow \mathcal{B}$
- (b) $z : \mathcal{L}$
- (1) $y : \mathcal{L} \rightarrow \mathcal{B}$
- (2) $z : \mathcal{L}$
- (3) $yz : \mathcal{B}$
- (4) $\lambda z : \mathcal{L}. yz : \mathcal{L} \rightarrow \mathcal{B}$
- (5) $\lambda y : \mathcal{L} \rightarrow \mathcal{B}. \lambda z : \mathcal{L}. yz : (\mathcal{L} \rightarrow \mathcal{B}) \rightarrow \mathcal{L} \rightarrow \mathcal{B}$

$$\begin{array}{c}
 \text{(var)} \quad \boxed{\Gamma \vdash X : T \text{ if } X : T \in \Gamma} \\
 \text{(app)} \quad \frac{\Gamma \vdash A : S \rightarrow T \quad \Gamma \vdash B : S}{\Gamma \vdash AB : T} \leftrightarrow \frac{A \Rightarrow B \quad A}{B} \text{ (}\Rightarrow\text{-elim)} \\
 \text{(abst)} \quad \frac{\Gamma, X : S \vdash A : T}{\Gamma \vdash \lambda X : S. A : S \rightarrow T} \leftrightarrow \boxed{\begin{array}{c} \text{Assume } A \\ \vdots \\ B \\ A \Rightarrow B \end{array}} \text{ (}\Rightarrow\text{-intro)}
 \end{array}$$

The Curry-Howard Isomorphism

- aka PAT: Propositions As Types
Proofs As Terms
- interpret context as logical assumptions and \rightarrow as \Rightarrow
 $(\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta$ becomes $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$
 $\lambda y : \alpha \rightarrow \beta. \lambda z : \alpha. yz$ becomes a proof of $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$ (in the empty context)
- a proposition is true if and only if the corresponding type is inhabited
(type inhabited: term of given type exists)
- so far: only propositions and \Rightarrow modelled by type theory
richer type theory: all of logic (and set theory) can be modelled!

(a)	$y : \mathcal{L} \rightarrow \mathcal{B}$
(b)	$z : \mathcal{L}$
(1)	$y : \mathcal{L} \rightarrow \mathcal{B}$
(2)	$z : \mathcal{L}$
(3)	$yz : \mathcal{B}$
(4)	$\lambda z : \mathcal{L}. yz : \mathcal{L} \rightarrow \mathcal{B}$
(5)	$\lambda y : \mathcal{L} \rightarrow \mathcal{B}. \lambda z : \mathcal{L}. yz : (\mathcal{L} \rightarrow \mathcal{B}) \rightarrow \mathcal{L} \rightarrow \mathcal{B}$

$$\begin{array}{lcl}
 \text{(var)} & \Gamma \vdash X : T \text{ if } X : T \in \Gamma & \\
 \text{(app)} & \frac{\Gamma \vdash A : S \rightarrow T \quad \Gamma \vdash B : S}{\Gamma \vdash AB : T} & \leftrightarrow \frac{A \Rightarrow B \quad A}{B} \text{ (}\Rightarrow\text{-elim)} \\
 \text{(abst)} & \frac{\Gamma, X : S \vdash A : T}{\Gamma \vdash \lambda X : S. A : S \rightarrow T} & \leftrightarrow \boxed{\begin{array}{c} \text{Assume } A \\ \vdots \\ B \\ A \Rightarrow B \end{array}} \text{ (}\Rightarrow\text{-intro)}
 \end{array}$$

The Curry-Howard Isomorphism

- aka PAT: Propositions As Types
Proofs As Terms
- interpret context as logical assumptions and \rightarrow as \Rightarrow
 $(\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta$ becomes $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$
 $\lambda y : \alpha \rightarrow \beta. \lambda z : \alpha. yz$ becomes a proof of $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$ (in the empty context)
- a proposition is true if and only if the corresponding type is inhabited
(type inhabited: term of given type exists)
- so far: only propositions and \Rightarrow modelled by type theory
richer type theory: all of logic (and set theory) can be modelled!
- philosophical implication: computing and arguing are the same thing!

(a)	$y : \mathcal{L} \rightarrow \mathcal{B}$
(b)	$z : \mathcal{L}$
(1)	$y : \mathcal{L} \rightarrow \mathcal{B}$
(2)	$z : \mathcal{L}$
(3)	$yz : \mathcal{B}$
(4)	$\lambda z : \mathcal{L}. yz : \mathcal{L} \rightarrow \mathcal{B}$
(5)	$\lambda y : \mathcal{L} \rightarrow \mathcal{B}. \lambda z : \mathcal{L}. yz : (\mathcal{L} \rightarrow \mathcal{B}) \rightarrow \mathcal{L} \rightarrow \mathcal{B}$

$$\begin{array}{ll}
 \text{(var)} & \Gamma \vdash X : T \text{ if } X : T \in \Gamma \\
 \text{(app)} & \frac{\Gamma \vdash A : S \rightarrow T \quad \Gamma \vdash B : S}{\Gamma \vdash AB : T} \leftrightarrow \frac{A \Rightarrow B \quad A}{B} \text{ (}\Rightarrow\text{-elim)} \\
 \text{(abst)} & \frac{\Gamma, X : S \vdash A : T}{\Gamma \vdash \lambda X : S. A : S \rightarrow T} \leftrightarrow \boxed{\begin{array}{c} \text{Assume } A \\ \vdots \\ B \\ A \Rightarrow B \end{array}} \text{ (}\Rightarrow\text{-intro)}
 \end{array}$$

The Curry-Howard Isomorphism

- aka PAT: Propositions As Types
Proofs As Terms
- interpret context as logical assumptions and \rightarrow as \Rightarrow
 $(\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta$ becomes $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$
 $\lambda y : \alpha \rightarrow \beta. \lambda z : \alpha. yz$ becomes a proof of $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$ (in the empty context)
- a proposition is true if and only if the corresponding type is inhabited
(type inhabited: term of given type exists)
- so far: only propositions and \Rightarrow modelled by type theory
richer type theory: all of logic (and set theory) can be modelled!
- philosophical implication: computing and arguing are the same thing!
(cf. also: Church-Turing Thesis)

The Curry-Howard Isomorphism

- aka PAT: Propositions As Types
Proofs As Terms
- interpret context as logical assumptions and \rightarrow as \Rightarrow
 $(\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta$ becomes $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$
 $\lambda y : \alpha \rightarrow \beta. \lambda z : \alpha. yz$ becomes a proof of $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$ (in the empty context)
- a proposition is true if and only if the corresponding type is inhabited
(type inhabited: term of given type exists)
- so far: only propositions and \Rightarrow modelled by type theory
richer type theory: all of logic (and set theory) can be modelled!
- philosophical implication: computing and arguing are the same thing!
(cf. also: Church-Turing Thesis)
- advantage over set theory: proofs are objects of type theory but meta-objects of set theory

The Curry-Howard Isomorphism

- aka PAT: Propositions As Types
Proofs As Terms
- interpret context as logical assumptions and \rightarrow as \Rightarrow
 $(\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta$ becomes $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$
 $\lambda y : \alpha \rightarrow \beta. \lambda z : \alpha. yz$ becomes a proof of $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$ (in the empty context)
- a proposition is true if and only if the corresponding type is inhabited
(type inhabited: term of given type exists)
- so far: only propositions and \Rightarrow modelled by type theory
richer type theory: all of logic (and set theory) can be modelled!
- philosophical implication: computing and arguing are the same thing!
(cf. also: Church-Turing Thesis)
- advantage over set theory: proofs are objects of type theory but meta-objects of set theory
 λ -terms: data structure for all logical arguments!

The Curry-Howard Isomorphism

- aka PAT: Propositions As Types
Proofs As Terms
- interpret context as logical assumptions and \rightarrow as \Rightarrow
 $(\alpha \rightarrow \beta) \rightarrow \alpha \rightarrow \beta$ becomes $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$
 $\lambda y : \alpha \rightarrow \beta. \lambda z : \alpha. yz$ becomes a proof of $(\alpha \Rightarrow \beta) \Rightarrow \alpha \Rightarrow \beta$ (in the empty context)
- a proposition is true if and only if the corresponding type is inhabited
(type inhabited: term of given type exists)
- so far: only propositions and \Rightarrow modelled by type theory
richer type theory: all of logic (and set theory) can be modelled!
- philosophical implication: computing and arguing are the same thing!
(cf. also: Church-Turing Thesis)
- advantage over set theory: proofs are objects of type theory but meta-objects of set theory
 λ -terms: data structure for all logical arguments!
basis of Lean: inductive types

A crash course in type theory



Haskell Curry
(1900 - 1982)



William Alvin Howard
(1926)

A richer type theory: the Calculus of Constructions

(silent)	$\boxed{(\text{ord})}$	$\boxed{() \vdash * : \square}$	Two rules in one: $s = *$ or $s = \square$
(silent)	$\boxed{(\text{var})}$	$\frac{\Gamma \vdash A : s}{\Gamma, X : A \vdash X : A}$	if $X \notin \text{dom}(\Gamma)$
(silent)	$\boxed{(\text{weak})}$	$\frac{\Gamma \vdash A : B \quad \Gamma \vdash C : s}{\Gamma, X : C \vdash A : B}$	if $X \notin \text{dom}(\Gamma)$
(silent)	$\boxed{(\text{form})}$	$\frac{\Gamma \vdash A : s_1 \quad \Gamma, X : A \vdash B : s_2}{\Gamma \vdash \Pi X : A. B : s_2}$	four rules in one $\leftarrow = A \rightarrow B \text{ if } X \notin \text{FV}(B)$
(app)	$\Gamma \vdash A : \Pi X : S. T \quad \Gamma \vdash B : S$	$\Gamma \vdash AB : T[X := B]$	(silent)
(abst)	$\Gamma, X : S \vdash A : T \quad \Gamma \vdash \Pi X : S. T : s$	$\Gamma \vdash \lambda X : S. A : \Pi X : S. T$	
(conv)	$\Gamma \vdash A : B \quad \Gamma \vdash B' : s$	$\Gamma \vdash A : B'$	if $B =_n B'$ (silent)

A richer type theory: the Calculus of Constructions

- (1) Sets $S : *.$
- (2) Propositions $P : *.$
- (3) Predicates $P : S \rightarrow *.$
- (4) Implication $A \Rightarrow B = A \rightarrow B (= \prod x : A. B \text{ if } x \notin FV(B)).$
- (5) Universal q. $\forall x \in S : P(x) = \prod x : S. P_x.$
- (6) Absurdity $\perp \equiv \prod \mathcal{L} : *. \mathcal{L}.$
- (7) Negation $\neg \equiv \prod A : *. A \rightarrow \perp.$
- (8) Conjunction $\wedge \equiv \prod A : *. \prod B : *. \prod C : *. (A \rightarrow B \rightarrow C) \rightarrow C, A \wedge B =_A AB.$
- (9) Disjunction $\vee \equiv \prod A : *. \prod B : *. \prod C : *. (A \rightarrow C) \rightarrow (B \rightarrow C) \rightarrow C, A \vee B =_V AB.$
- (10) Existential q. $\exists \equiv \prod S : *. \prod P : S \rightarrow *. \prod C : *. ((\prod x : S. (P_x \rightarrow C)) \rightarrow C),$
 $\exists x \in S : P(x) = \exists S P.$
- (11) Axioms Add assumption in front of context.

A richer type theory: the Calculus of Constructions

$$(A \vee B) \Rightarrow (\neg A \Rightarrow B)$$

$\vdash \lambda C : (\prod C : * . (A \rightarrow C) \rightarrow (B \rightarrow C) \rightarrow C) \rightarrow (A \rightarrow \perp) \rightarrow B$.

- (a) $A : *$
- (b) $B : *$
- (c) $x : \prod C : * . (A \rightarrow C) \rightarrow (B \rightarrow C) \rightarrow C$
- (d) $y : A \rightarrow \perp$
- (1) $xB : (A \rightarrow B) \rightarrow (B \rightarrow B) \rightarrow B$ (app) on (c), (b)
- (e) $u : A$
- (2) $yu : \perp$ (app) on (d), (e)
- (3) $|yuB : B$ (app) on (2), (b)
- (4) $\lambda u : A. yuB : A \rightarrow B$ (ab) on (3)
- (5) $xB(\lambda u : A. yuB) : (B \rightarrow B) \rightarrow B$ (app) on (1), (4)
- (f) $v : B$
- (6) $|v : B$ (var) on (f)
- (7) $\lambda v : B. v : B \rightarrow B$ (ab) on (6)
- (8) $xB(\lambda u : A. yuB)(\lambda v : B. v) : B$ (app) on (5), (7)
- (9) $\lambda y : A \rightarrow \perp. xB(\lambda u : A. yuB)(\lambda v : B. v) : (A \rightarrow \perp) \rightarrow B$ (ab) on (8)
- (10) $\lambda x : (\prod C : * . (A \rightarrow C) \rightarrow (B \rightarrow C) \rightarrow C). \lambda y : A \rightarrow \perp. xB(\lambda u : A. yuB)(\lambda v : B. v) : (\prod C : * . (A \rightarrow C) \rightarrow (B \rightarrow C) \rightarrow C) \rightarrow (A \rightarrow \perp) \rightarrow B$ (ab) on (9)

A screenshot of a Lean 4 development environment within the Visual Studio Code (VS Code) interface. The window title is "GEB".

Explorer View: Shows a file tree with the following structure:

- GRUPPE 1
 - Talklean GEB
- GRUPPE 2
 - Lean Infoview

The file "Talklean.lean" is currently selected and has a status bar message "nicht gespeichert" (not saved).

Code Editor: The main editor area displays the following code:

```
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5
```

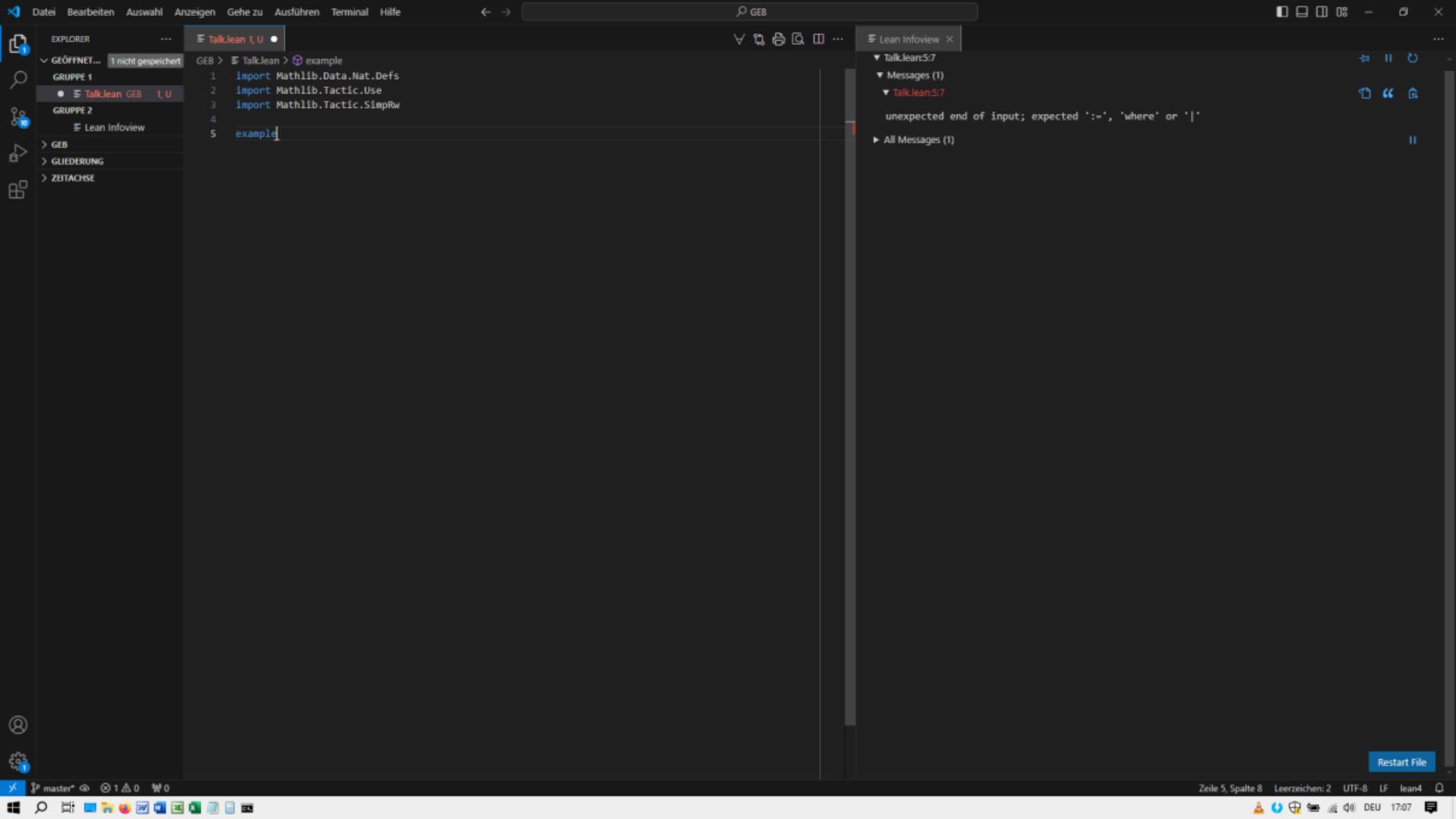
Terminal: The terminal tab shows the command "GEB" and its output.

Lean Infoview: The right-hand panel displays information about the current file:

- Talklean:5:0
- No info found.
- All Messages (0)

Bottom Status Bar: Shows the current file path "master", line "Zeile 5, Spalte 1", character count "Leerzeichen:2", encoding "UTF-8", line separator "LF", and file extension "lean4". It also includes a "Restart File" button.

Taskbar: The bottom of the screen shows the Windows taskbar with various pinned icons.



File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

Explorer ...

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1 • E Talklean GEB 1, U

GRUPPE 2 E Lean Infoview

> GEB

> GUEDERUNG

> ZETACHSE

E Talklean 1, U

GEB > E Talklean > example

```
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 example (P : Prop) [Q : Prop]
```

Lean Infoview

Talklean:5:29

Messages (1)

Talklean:5:29

unexpected end of input; expected `:-`, `where` or `|`

All Messages (1)

Restart File

master* ④ ⑤ 1 ▲ 0 ⚡ 0

Zeile 5, Spalte 30 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:08

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB

EXPLORER

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E Talklean GEB 1, U

GRUPPE 2

- E Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

Talklean 1, U

GEB > E Talklean > ...

```
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 example (P : Prop) (Q : Prop) : |
```

Lean Infoview

- ▼ Talklean:5:32
- ▼ Messages (1)
- ▼ Talklean:5:32
- unexpected end of input
- All Messages (1)

Restart File

Zeile 5, Spalte 33 Leerzeichen:2 UTF-8 LF lean4 DEU 17:08

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

Explorer ...

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1 • E Talklean GEB 1, U

GRUPPE 2 E Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

E Talklean 1, U

GEB > E Talklean > example

```
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 example (P : Prop) (Q : Prop) : (P + Q) → P + Q
```

Lean Infoview

Talklean:5:47

Expected type

P Q : Prop

Prop

Messages (1)

Talklean:5:47

unexpected end of input; expected `:=`, `where` or `|`

All Messages (1)

Restart File

Zeile 5, Spalte 48 Leerzeichen:2 UTF-8 LF lean4 DEU 17:08

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB

EXPLORER

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E Talklean GEB 1, U

GRUPPE 2

- E Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

Talklean 1, U

GEB > E Talklean > example

```
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 example (P : Prop) (Q : Prop) : (P + Q) + P + Q :=
```

Lean Infoview

- ▼ Talklean:5:50
- ▼ Messages (1)
- ▼ Talklean:5:50
- unexpected end of input
- All Messages (1)

Restart File

Zeile 5, Spalte 51 Leerzeichen:2 UTF-8 LF lean4 DEU 17:09

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

Explorer

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E Talklean GEB 1, U

GRUPPE 2

- E Lean Infoview

> GEB
> GLIEDERUNG
> ZEITACHSE

Talklean 1, U

GEB > E Talklean > example

```
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 example (P : Prop) (Q : Prop) : (P + Q) + P + Q :=
```

Lean Infoview

Talklean6:2

Expected type

P Q : Prop

↳ (P + Q) + P + Q

Messages (1)

Talklean6:2

don't know how to synthesize placeholder context:

P Q : Prop

↳ (P + Q) + P + Q

All Messages (1)

Restart File

master ④ ⑤ 1 ▲ 0 ⚡ 0

Zeile 6, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:10

Windows Taskbar icons

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

Explorer ...

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E TalkLean GEB 1, U

GRUPPE 2

- E Lean Infoview

> GEB
> GLIEDERUNG
> ZEITACHSE

TalkLean 1, U

GEB > E TalkLean > example

```
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 example (P : Prop) (Q : Prop) : (P + Q) + P + Q := 
6 | λ (y : P + Q) => 
```

Lean Infoview

TalkLean:6:19

Expected type

P Q : Prop

y : P → Q

↳ P → Q

Messages (1)

TalkLean:6:19

don't know how to synthesize placeholder context:

P Q : Prop

y : P → Q

↳ P → Q

All Messages (1)

Restart File

master* ④ ⑤ 1 ▲ 0 ⚡ 0

Zeile 6, Spalte 20 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:10

Windows Taskbar icons

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

Explorer ...

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E TalkLean GEB 1, U

GRUPPE 2

- E Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

TalkLean 1, U

GEB > E TalkLean > example

```
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 example (P : Prop) (Q : Prop) : (P + Q) + P + Q := 
6 | λ (y : P + Q) => λ (x : P) => [
```

Lean Infoview

Lean 6.32

Expected type

P Q : Prop

y : P → Q

x : P

h Q

Messages (1)

Lean 6.32

don't know how to synthesize placeholder context:

P Q : Prop

y : P + Q

x : P

h Q

All Messages (1)

Restart File

master* ④ ⑤ 1 ▲ 0 ⚡ 0

Zeile 6, Spalte 33 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:11

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB

EXPLORER

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E Talklean GEB U

GRUPPE 2

- E Lean Infoview

> GEB

> GLÜEDERUNG

> ZEITACHSE

Talklean U

GEB > Talklean > example

```
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 example (P : Prop) (Q : Prop) : (P → Q) → P → Q := 
6 | λ (y : P → Q) => λ (x : P) => y x
```

Lean Infoview

Talklean:6:35

Expected type

P Q : Prop

y : P → Q

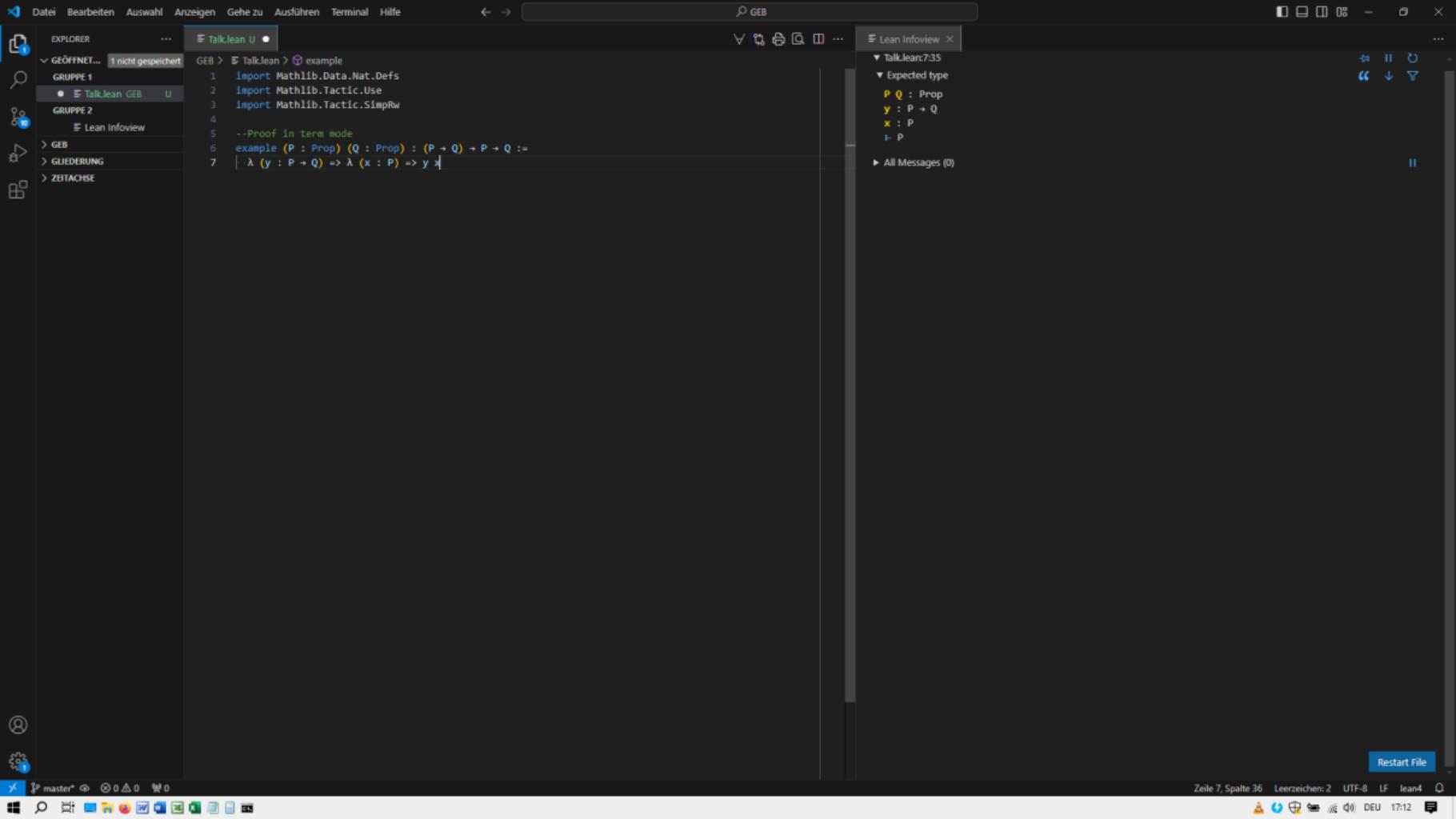
x : P

|- P

All Messages (0)

Restart File

Zeile 6, Spalte 36 Leerzeichen:2 UTF-8 LF lean4 DEU 17:11



File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > example

```
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 --Proof in term mode
6 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := 
7   λ (y : P → Q) => λ (x : P) => y x
8
9 example (P : Prop) (Q : Prop) : (P → Q) + P + Q :=
```

Lean Infoview

- ▼ Talklean:9:50
- ▼ Messages (1)
- ▼ Talklean:9:50
- unexpected end of input
- All Messages (1)

Restart File

Zeile 9, Spalte 51 Leerzeichen:2 UTF-8 LF lean4 DEU 17:12

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

Explorer ...

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E Talklean GEB 2, U

GRUPPE 2

- E Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

Talklean 2, U

```
GEB > E Talklean > ...
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 --Proof in term mode
6 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := 
7   λ (y : P → Q) => λ (x : P) => y x
8
9 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
10
```

Lean Infoview

Talklean:102

Tactic state

1 goal

P Q : Prop

|- (P → Q) + P + Q

Messages (1)

Talklean:102

unexpected end of input; expected '{'

All Messages (2)

Restart File

master 2 △ 0 ⌂ 0

Zeile 10, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:13

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB

EXPLORER

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E TalkLean GEB 1, U

GRUPPE 2

- E Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

TalkLean 1, U

GEB > E TalkLean > ...

```
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 --Proof in term mode
6 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := 
7   λ (y : P → Q) => λ (x : P) => y x
8
9 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
10 intro y
11
```

Lean Infoview

TalkLean:11:2

Tactic state

1 goal

P Q : Prop

y : P → Q

|- P → Q

All Messages (1)

Restart File

Zeile 11, Spalte 3 Leerzeichen:2 UTF-8 LF lean4 DEU 17:13

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 --Proof in term mode
6 example (P : Prop) (Q : Prop) : (P → Q) + P + Q :=
7 λ (y : P → Q) => λ (x : P) => y x
8
9 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
10 intro y
11 intro x

Lean Infoview

Talklean:12:2
Tactic state
1 goal
P Q : Prop
y : P → Q
x : P
· · ·
All Messages (1)

Restart File

Zeile 12, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:13

A screenshot of a Lean 4 development environment within the Visual Studio Code (VS Code) interface. The main code editor shows a file named "Talklean.lean" containing the following Lean 4 code:

```
GEB > Talklean > ...
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 --Proof in term mode
6 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := 
7   λ (y : P → Q) => λ (x : P) => y x
8
9 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
10 intro y
11 intro x
12 exact y x
13
```

The Explorer sidebar on the left lists open files: "GEB" (not saved), "GRUPPE 1" (containing "Talklean GEB"), "GRUPPE 2" (containing "Lean Infoview"), and "ZETACHSE". The Status Bar at the bottom right shows "Restart File".

The Lean Infoview panel on the right displays the following information:

- Talklean:13:2
- Tactic state
- No goals
- All Messages (0)

The bottom taskbar shows the current workspace path as "D:\master" and includes icons for search, file operations, and system status.

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

```
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 --Proof in term mode
6 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := 
7   λ (y : P → Q) => λ (x : P) => y x
8
9 --Proof in tactic mode
10 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
11   intro y
12   intro x
13   exact y x
14
```

Lean Infoview

- ▼ Talklean:14:2
- ▼ Tactic state
- No goals

All Messages (0)

Restart File

Zeile 14, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:14



Talklean U

```
GEB > Talklean > ...
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 --Proof in term mode
6 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := 
7   λ (y : P → Q) => λ (x : P) => y x
8
9 --Proof in tactic mode
10 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
11   intro y
12   intro x
13   exact y x
14
15 --Named version of previous example
16 lemma simple_lemma_1 (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
17   intro y
18   intro x
19   exact y x
20
```

Lean Infoview

```
Talklean:202
Tactic state
No goals
All Messages (0)
```

Restart File

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

REPL Explorer Lean Infoview

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1 • E Talklean GEB 2, U

GRUPPE 2 E Lean Infoview

> GEB

> GLÜEDERUNG

> ZETACHSE

1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 --Proof in term mode
6 example (P : Prop) (Q : Prop) : (P → Q) + P + Q :=
7 λ (y : P + Q) => λ (x : P) => y x
8
9 --Proof in tactic mode
10 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
11 intro y
12 intro x
13 exact y x
14
15 --Named version of previous example
16 lemma simple_lemma_1 (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
17 intro y
18 intro x
19 exact y x
20
21 --Moving assumption from context to statement
22 lemma simple_lemma_2 : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q := by

Lean Infoview

Talklean:23:2
Tactic state
1 goal
 $\vdash \forall (P\ Q : \text{Prop}), (P \rightarrow Q) + P + Q$
Messages (1)
Talklean:23:2
unexpected end of input; expected '{'
All Messages (2)

Restart File

Zeile 23, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:16

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Lean Infoview

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E Talklean GEB 1, U

GRUPPE 2

- E Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

REPL

Code Editor (Talklean 1, U)

```
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 --Proof in term mode
6 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
7   λ (y : P + Q) => λ (x : P) => y x
8
9 --Proof in tactic mode
10 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
11   intro y
12   intro x
13   exact y x
14
15 --Named version of previous example
16 lemma simple_lemma_1 (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
17   intro y
18   intro x
19   exact y x
20
21 --Moving assumption from context to statement
22 lemma simple_lemma_2 : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q := by
23   intro P
24
```

Lean Infoview

- Talklean:24:2
- Tactic state
- 1 goal
- P : Prop
- ↑ v (Q : Prop), (P + Q) + P + Q

All Messages (1)

Restart File

Zeile 24, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:16

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Lean Infoview

Explorer

GlöÖFNET... 1 nicht gespeichert

GRUPPE 1

- E Talklean GEB 1, U

GRUPPE 2

- E Lean Infoview

> GEB

> GLÜEDERUNG

> ZEITACHSE

Talklean 1, U

```
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 --Proof in term mode
6 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
7   λ (y : P + Q) => λ (x : P) => y x
8
9 --Proof in tactic mode
10 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
11   intro y
12   intro x
13   exact y x
14
15 --Named version of previous example
16 lemma simple_lemma_1 (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
17   intro y
18   intro x
19   exact y x
20
21 --Moving assumption from context to statement
22 lemma simple_lemma_2 : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q := by
23   intro P
24   intro Q
```

Lean Infoview

Talklean:252

Tactic state

1 goal

P Q : Prop

|- (P → Q) + P + Q

All Messages (1)

Restart File

Zeile 25, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:17

GlÖFFNET... 1 nicht gespeichert

```
GEB > Talklean > ...
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 --Proof in term mode
6 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := 
7   λ (y : P → Q) => λ (x : P) => y x
8
9 --Proof in tactic mode
10 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
11   intro y
12   intro x
13   exact y x
14
15 --Named version of previous example
16 lemma simple_lemma_1 (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
17   intro y
18   intro x
19   exact y x
20
21 --Moving assumption from context to statement
22 lemma simple_lemma_2 : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q := by
23   intro P
24   intro Q
25   intro y
26   intro x
27   exact y x
28
```

Lean Infoview

```
▼ Talklean:282
▼ Tactic state
No goals
▶ All Messages (0)
```

Restart File



File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

```
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 --Proof in term mode
6 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := 
7   λ (y : P + Q) => λ (x : P) => y x
8
9 --Proof in tactic mode
10 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
11   intro y
12   intro x
13   exact y x
14
15 --Named version of previous example
16 lemma simple_lemma_1 (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
17   intro y
18   intro x
19   exact y x
20
21 --Moving assumption from context to statement
22 lemma simple_lemma_2 : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q := by
23   intro P
24   intro Q
25   intro y
26   intro x
27   exact y x
28
29 --Check types of lemmas
30 #check simple_lemma_1
```

Lean Infoview

- ▼ Talklean:30:1
- ▼ Expected type
- ↳ $\forall (P\ Q : \text{Prop}), (P \rightarrow Q) + P + Q$
- ▼ Messages (1)
- ▼ Talklean:30:0
- simple_lemma_1 (P Q : Prop) : (P → Q) + P + Q
- All Messages (1)

Restart File

Zeile 30 Spalte 22 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:18

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean U

GRUPPE 1

- Talklean GEB U

GRUPPE 2

- Lean Infoview

> GEB

> QUEDERUNG

> ZEITACHSE

EXPLORER

GLEFFNET... [nicht gespeichert]

1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 --Proof in term mode
6 example (P : Prop) (Q : Prop) : (P → Q) + P + Q :=
7 λ (y : P + Q) => λ (x : P) => y x
8
9 --Proof in tactic mode
10 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
11 intro y
12 intro x
13 exact y x
14
15 --Named version of previous example
16 lemma simple_lemma_1 (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
17 intro y
18 intro x
19 exact y x
20
21 --Moving assumption from context to statement
22 lemma simple_lemma_2 : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q := by
23 intro P
24 intro Q
25 intro y
26 intro x
27 exact y x
28
29 --Check types of lemmas
30 #check simple_lemma_1
31 #check simple_lemma_2

Lean Infoview

Talklean:31:21

Expected type

↳ ∀ (P Q : Prop), (P → Q) + P + Q

Messages (1)

Talklean:31:0

simple_lemma_2 (P Q : Prop) : (P → Q) + P + Q

All Messages (2)

Restart File

Zeile 31, Spalte 22 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:18

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean U

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1 • E Talklean GEB U

GRUPPE 2 E Lean Infoview

> GEB

> GLÜEDERUNG

> ZEITACHSE

REPL

EXPLORER

Talklean U

1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 --Proof in term mode
6 example (P : Prop) (Q : Prop) : (P → Q) + P + Q :=
7 λ (y : P + Q) => λ (x : P) => y x
8
9 --Proof in tactic mode
10 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
11 intro y
12 intro x
13 exact y x
14
15 --Named version of previous example
16 lemma simple_lemma_1 (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
17 intro y
18 intro x
19 exact y x
20
21 --Moving assumption from context to statement
22 lemma simple_lemma_2 : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q := by
23 intro P
24 intro Q
25 intro y
26 intro x
27 exact y x
28
29 --Check types of lemmas
30 #check simple_lemma_1
31 #check simple_lemma_2
32
33 --Print proof terms of lemmas
34 #print simple_lemma_1

Lean Infoview

Talklean:34:21

Expected type

↑ ∃ (P Q : Prop), (P → Q) + P + Q

Messages (1)

Talklean:34:0

theorem simple_lemma_1 : ∀ (P Q : Prop), (P → Q) + P + Q :=
fun P Q y x => y x

All Messages (3)

Restart File

Zeile 34 Spalte 22 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:18

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean U

GRUPPE 1

GRUPPE 2

Lean Infoview

1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 --Proof in term mode
6 example (P : Prop) (Q : Prop) : (P → Q) + P + Q :=
7 λ (y : P + Q) => λ (x : P) => y x
8
9 --Proof in tactic mode
10 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
11 intro y
12 intro x
13 exact y x
14
15 --Named version of previous example
16 lemma simple_lemma_1 (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
17 intro y
18 intro x
19 exact y x
20
21 --Moving assumption from context to statement
22 lemma simple_lemma_2 : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q := by
23 intro P
24 intro Q
25 intro y
26 intro x
27 exact y x
28
29 --Check types of lemmas
30 #check simple_lemma_1
31 #check simple_lemma_2
32
33 --Print proof terms of lemmas
34 #print simple_lemma_1
35 #print simple_lemma_2

Lean Infoview

Talklean:35:21

Expected type

↑ ∃ (P Q : Prop), (P → Q) + P + Q

Messages (1)

Talklean:35:0

theorem simple_lemma_2 : ∀ (P Q : Prop), (P → Q) + P + Q :=
fun P Q y x => y x

All Messages (4)

Restart File

Zeile 35, Spalte 22 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:19

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

RESTART FILE

EXPLORER

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E TalkLean GEB 2, U

GRUPPE 2

- Lean Infoview

> GEB

> GLÜEDERUNG

> ZEITACHSE

TalkLean 2, U

```
GEB > E TalkLean > example
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 --Proof in term mode
6 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := 
7   λ (y : P + Q) => λ (x : P) => y x
8
9 --Proof in tactic mode
10 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
11   intro y
12   intro x
13   exact y x
14
15 --Named version of previous example
16 lemma simple_lemma_1 (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
17   intro y
18   intro x
19   exact y x
20
21 --Moving assumption from context to statement
22 lemma simple_lemma_2 : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q := by
23   intro P
24   intro Q
25   intro y
26   intro x
27   exact y x
28
29 --Check types of lemmas
30 #check simple_lemma_1
31 #check simple_lemma_2
32
33 --Print proof terms of lemmas
34 #print simple_lemma_1
35 #print simple_lemma_2
36
37 --Copy and paste proof terms ("fun x => y" alternative to "λ x => y")
38 example : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q :=
39   fun P Q y x => y x
```

Lean Infoview

TalkLean:39:19

Expected type

P Q : Prop

y : P → Q

x : P

↳ P

Messages (2)

TalkLean:39:6

unused variable `P`
note: this linter can be disabled with `set_option linter.unusedVariables false`

TalkLean:39:8

unused variable `Q`
note: this linter can be disabled with `set_option linter.unusedVariables false`

All Messages (6)

Restart File

master* 0 ▲ 2 ◑ 4 ⌂ 0 17:22

Zeile 39, Spalte 20 Leerzeichen: 2 UTF-8 LF DEU

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > example

```
1 import Mathlib.Data.Nat.Defs
2 import Mathlib.Tactic.Use
3 import Mathlib.Tactic.SimpRW
4
5 --Proof in term mode
6 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := 
7   λ (y : P + Q) => λ (x : P) => y x
8
9 --Proof in tactic mode
10 example (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
11   intro y
12   intro x
13   exact y x
14
15 --Named version of previous example
16 lemma simple_lemma_1 (P : Prop) (Q : Prop) : (P → Q) + P + Q := by
17   intro y
18   intro x
19   exact y x
20
21 --Moving assumption from context to statement
22 lemma simple_lemma_2 : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q := by
23   intro P
24   intro Q
25   intro y
26   intro x
27   exact y x
28
29 --Check types of lemmas
30 #check simple_lemma_1
31 #check simple_lemma_2
32
33 --Print proof terms of lemmas
34 #print simple_lemma_1
35 #print simple_lemma_2
36
37 --Copy and paste proof terms ("fun x => y" alternative to "λ x => y")
38 example : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q :=
39   fun _ _ y x => y x
```

Lean Infoview

Talklean:39:19

Expected type

$x^{+^2} \ x^{\dagger} : \text{Prop}$

$y : x^{+^2} \rightarrow x^{\dagger}$

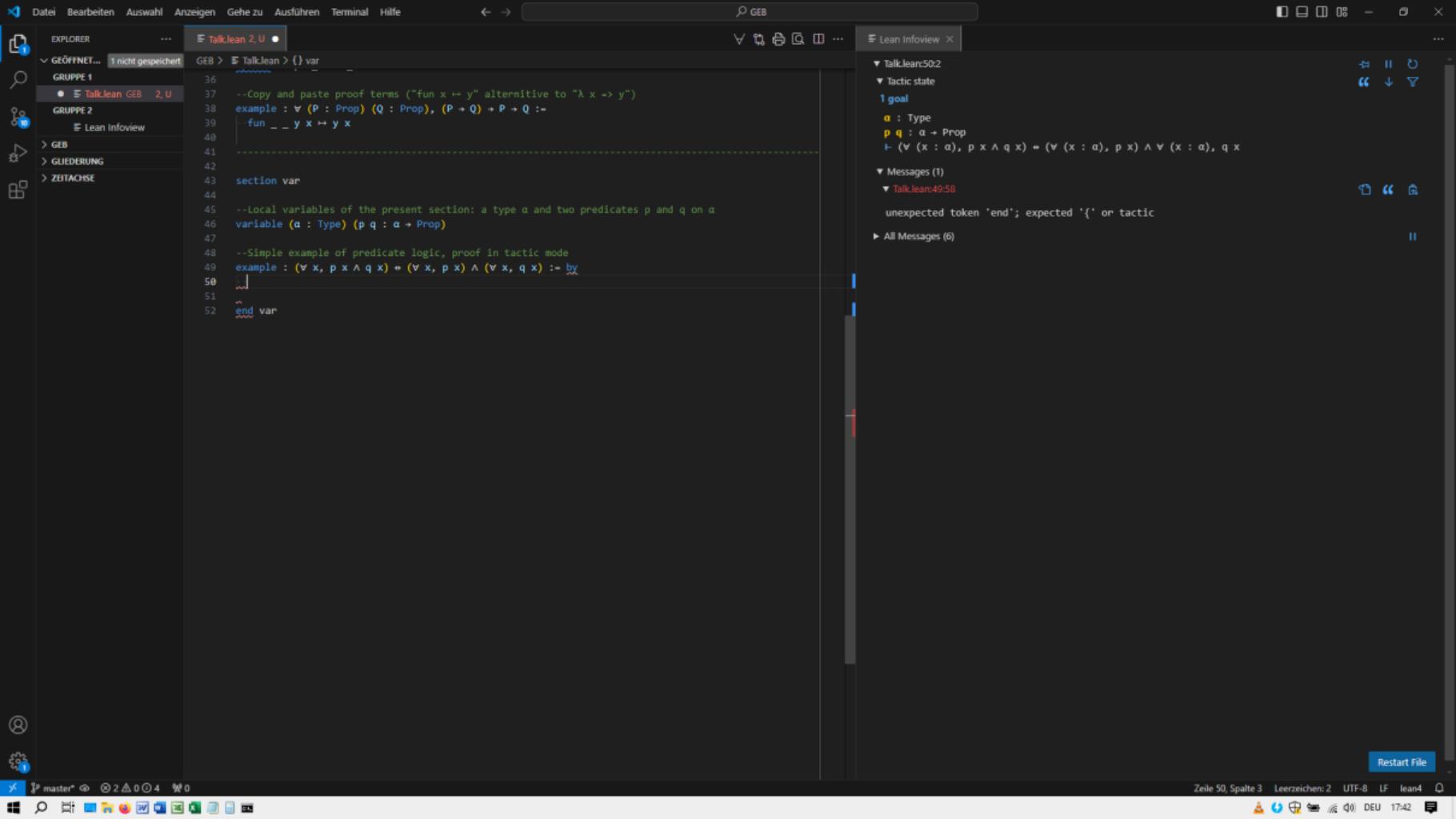
$x : x^{+^2}$

$\vdash x^{+^2}$

All Messages (4)

Restart File

Zeile 39, Spalte 20 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:22



EXPLORER ...

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1 • Talklean GEB 1, U

GRUPPE 2 Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

TALKLEAN 1, U

GEB > Talklean > () var

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q :=
39   fun _ _ y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
45 variable (a : Type) (p q : a → Prop)
46
47 --Simple example of predicate logic, proof in tactic mode
48 example : (∀ x, p x ∧ q x) + (∀ x, p x) ∧ (∀ x, q x) := by
49   constructor
50   |
51   |
52 end var
```

Lean Infoview

Talklean:51:2

Tactic state

2 goals

case mp

a : Type

p q : a → Prop

↑ (forall (x : a), p x ∧ q x) → (forall (x : a), p x) ∧ ∀ (x : a), q x

case mpr

a : Type

p q : a → Prop

↑ ((forall (x : a), p x) ∧ ∀ (x : a), q x) → ∀ (x : a), p x ∧ q x

All Messages (5)

Restart File

Zeile 51, Spalte 3 Leerzeichen:2 UTF-8 LF lean4

0 1 0 0 4

Windows Taskbar

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB

EXPLORER

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1

- TalkLean GEB 1, U

GRUPPE 2

- Lean Infoview

> GEB

> GUEDERUNG

> ZETACHSE

Talklean 1, U

GEB > Talklean > () var > example

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q :=
39   fun _ _ y x ↦ y x
40
41 -----
42
43 section var
44
45 --Local variables of the present section: a type a and two predicates p and q on a
46 variable (a : Type) (p q : a → Prop)
47
48 --Simple example of predicate logic, proof in tactic mode
49 example : (v x, p x ∧ q x) + (v x, p x) ∧ (v x, q x) := by
50   constructor
51   .. sorry
52   .. sorry
53
54 end var
```

Lean Infoview

Talklean:52:9

Tactic state

No goals

All Messages (5)

Restart File

Zeile 52, Spalte 10 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:44

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB

EXPLORER ...

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1 • E Talklean GEB 3, U

GRUPPE 2 E Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

Talklean 3, U

36 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
37 example : ∀ (P : Prop), (P → Q) + P + Q :=
38 fun _ _ y x ↦ y x
39
40
41 section var
42
43 --Local variables of the present section: a type a and two predicates p and q on a
44 variable (a : Type) (p q : a → Prop)
45
46 --Simple example of predicate logic, proof in tactic mode
47 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
48 constructor
49 |
50 sorry
51
52 end var

Lean Infoview

Talklean:51:4

Tactic state

1 goal

case mp

a : Type

p q : a → Prop

↑ (forall (x : a), p x ∧ q x) → (forall (x : a), p x) ∧ ∀ (x : a), q x

Messages (2)

Talklean:51:2

unsolved goals

case mp

a : Type

p q : a → Prop

↑ (forall (x : a), p x ∧ q x) → (forall (x : a), p x) ∧ ∀ (x : a), q x

Talklean:49:56

unsolved goals

case mpr

a : Type

p q : a → Prop

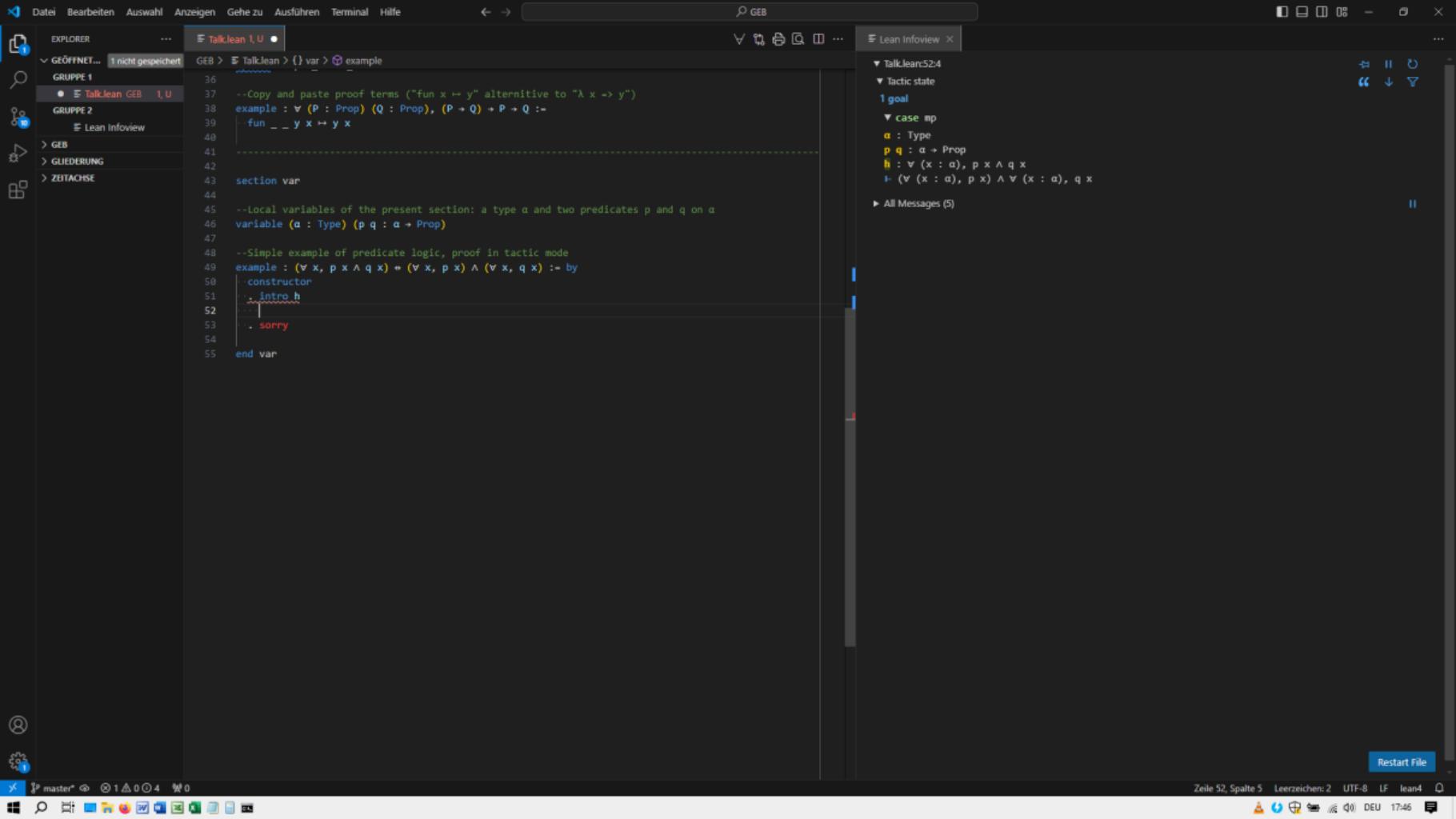
↑ ((forall (x : a), p x) ∧ ∀ (x : a), q x) → ∀ (x : a), p x ∧ q x

All Messages (7)

Restart File

master* 3 ▲ 0 0 4

Zeile 51, Spalte 5 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:46



File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

RESTART FILE

EXPLORER GLÖFFNET... 1 nicht gespeichert

GRUPPE 1 • Talklean GEB 1, U

GRUPPE 2 Lean Infoview

> GEB

> GLÜEDERUNG

> ZEITACHSE

Talklean 1, U

GEB > Talklean > () var > example

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q :=
39   fun _ _ y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
45 variable (a : Type) (p q : a → Prop)
46
47
48 --Simple example of predicate logic, proof in tactic mode
49 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
50   constructor
51   . intro h
52   . constructor
53   |
54   . sorry
55
56 end var
```

Lean Infoview

Talklean:53:4

Tactic state

2 goals

case mp.left

a : Type

p q : a → Prop

h : ∀ (x : a), p x ∧ q x

· · ·

case mp.right

a : Type

p q : a → Prop

h : ∀ (x : a), p x ∧ q x

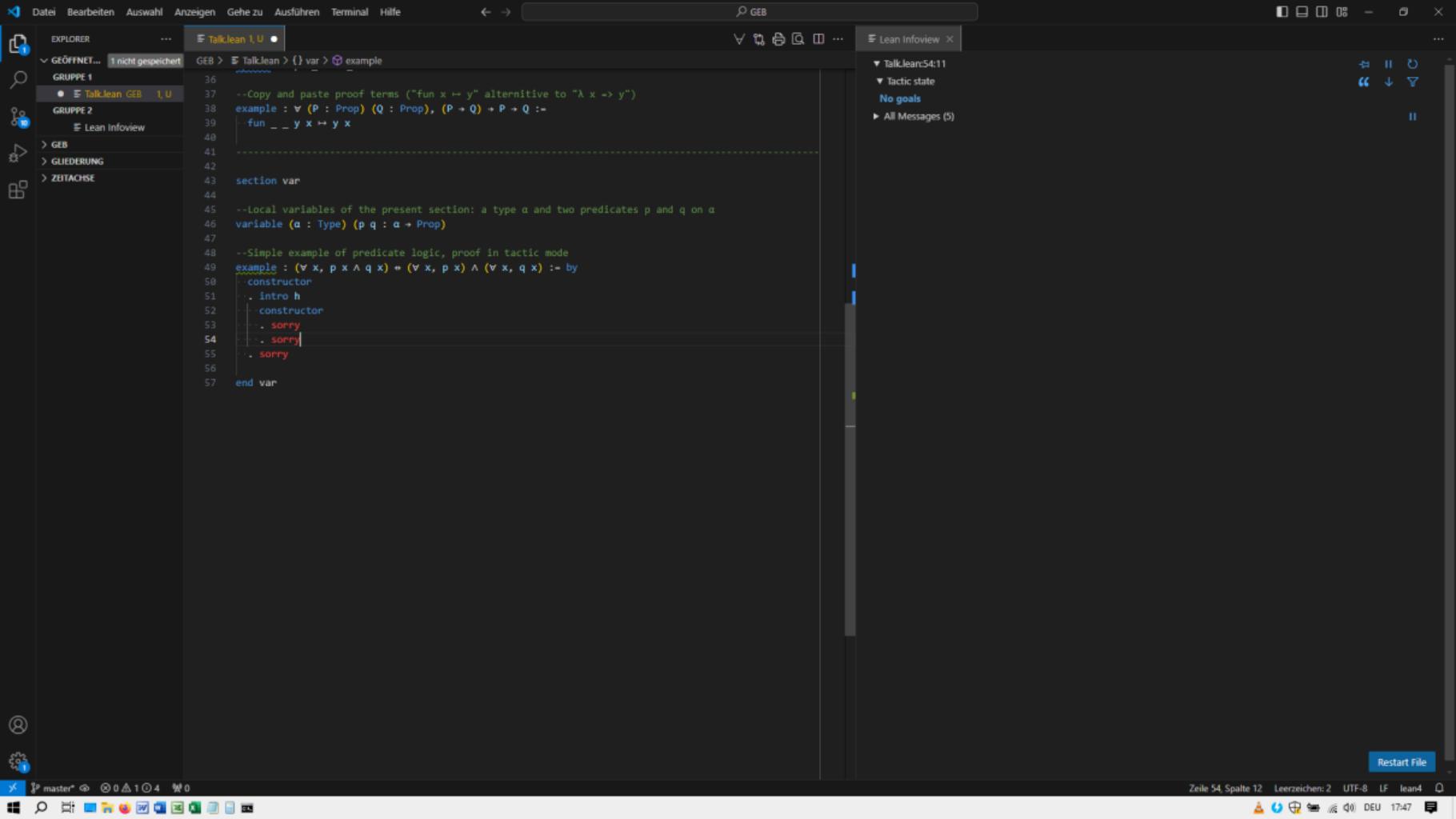
· · ·

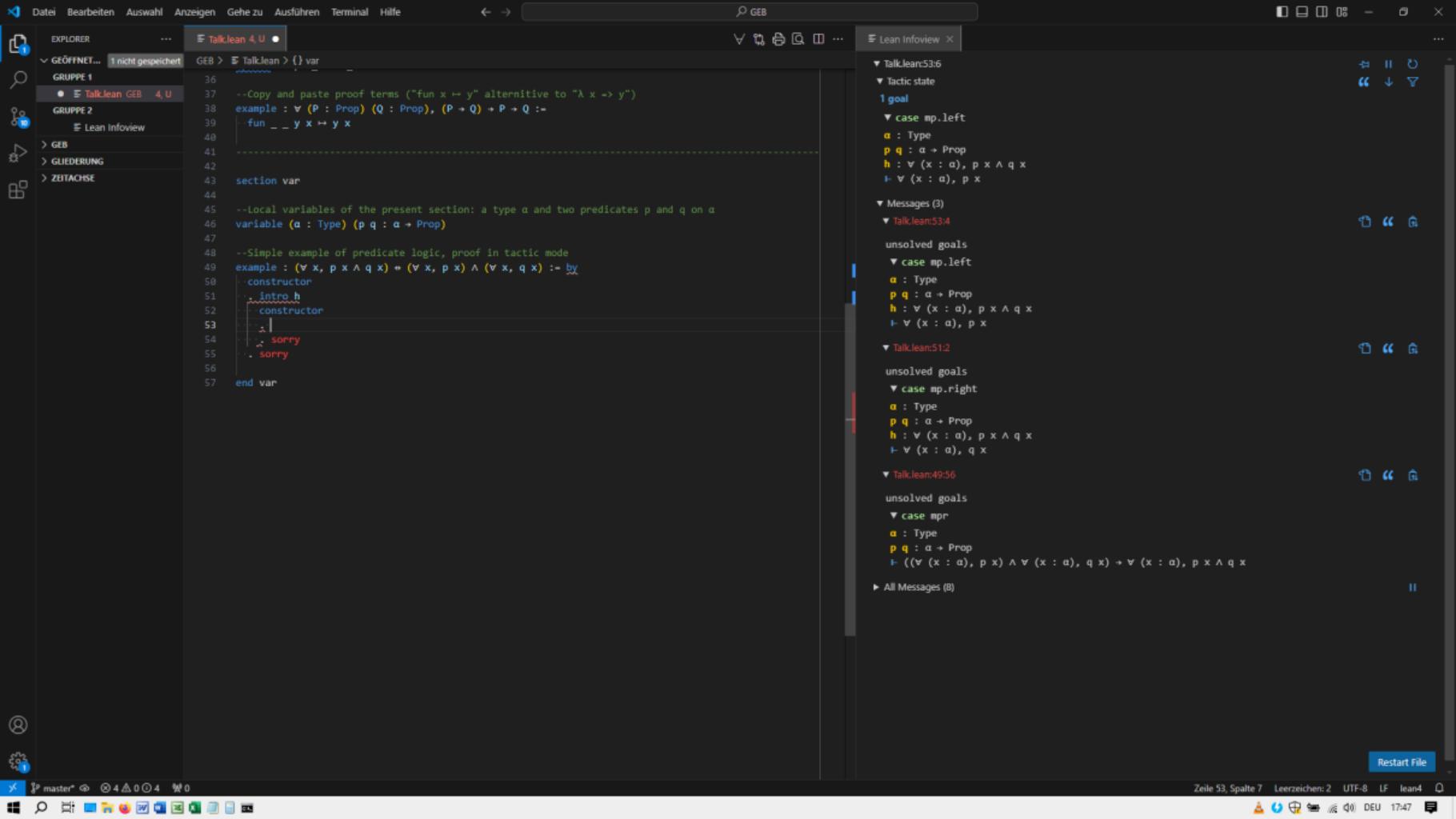
All Messages (5)

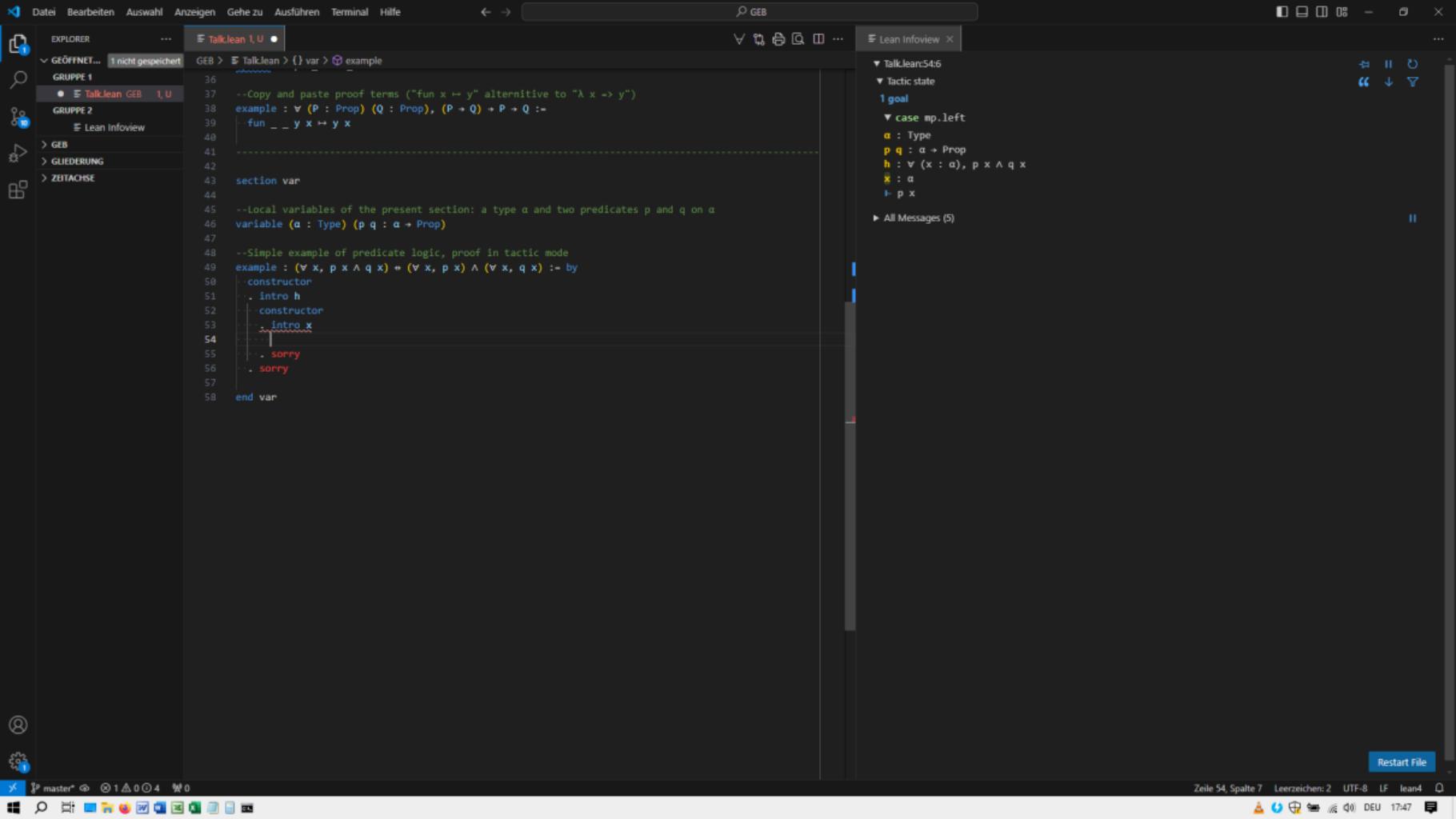
Restart File

master ① 1 ▲ 0 ④ ⑤

Zeile 53, Spalte 5 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:47







File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

RESTART FILE

Explorer

GlöÖfnet... 1 nicht gespeichert

GRUPPE 1

- TalkLean GEB 1, U

GRUPPE 2

- Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

Talklean 1, U

GEB > Talklean > () var > example

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ {P : Prop} {Q : Prop}, (P → Q) + P + Q :=
39   fun _ _ y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
45 variable (a : Type) (p q : a → Prop)
46
47
48 --Simple example of predicate logic, proof in tactic mode
49 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
50   constructor
51   . intro h
52   . constructor
53   . intro x
54   . exact (h x).1
55   . sorry
56   . sorry
57
58 end var
```

Lean Infoview

TalkLean:54:19

Tactic state

No goals

Expected type

- a : Type
- p q : a → Prop
- h : ∀ (x : a), p x ∧ q x
- x : a
- ∀ {a b : Prop}, a ∧ b → a

All Messages (5)

Zeile 54, Spalte 20 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:47

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB

Explorer

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E Talklean GEB 3, U

GRUPPE 2

- Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

Talklean 3, U

36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q :=
39 fun _ _ y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
variable (a : Type) (p q : a → Prop)
45
46 --Simple example of predicate logic, proof in tactic mode
47 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
48 constructor
49 . intro h
50 . constructor
51 . intro x
52 . exact (h x).1
53
54
55 λ |
56 sorry
57
58 end var

Lean Infoview

Talklean:55:6

Tactic state

1 goal

case mp.right

a : Type

p q : a → Prop

h : ∀ (x : a), p x ∧ q x

↑ ∀ (x : a), q x

Messages (2)

Talklean:55:4

unsolved goals

case mp.right

a : Type

p q : a → Prop

h : ∀ (x : a), p x ∧ q x

↑ ∀ (x : a), q x

Talklean:49:56

unsolved goals

case mpr

a : Type

p q : a → Prop

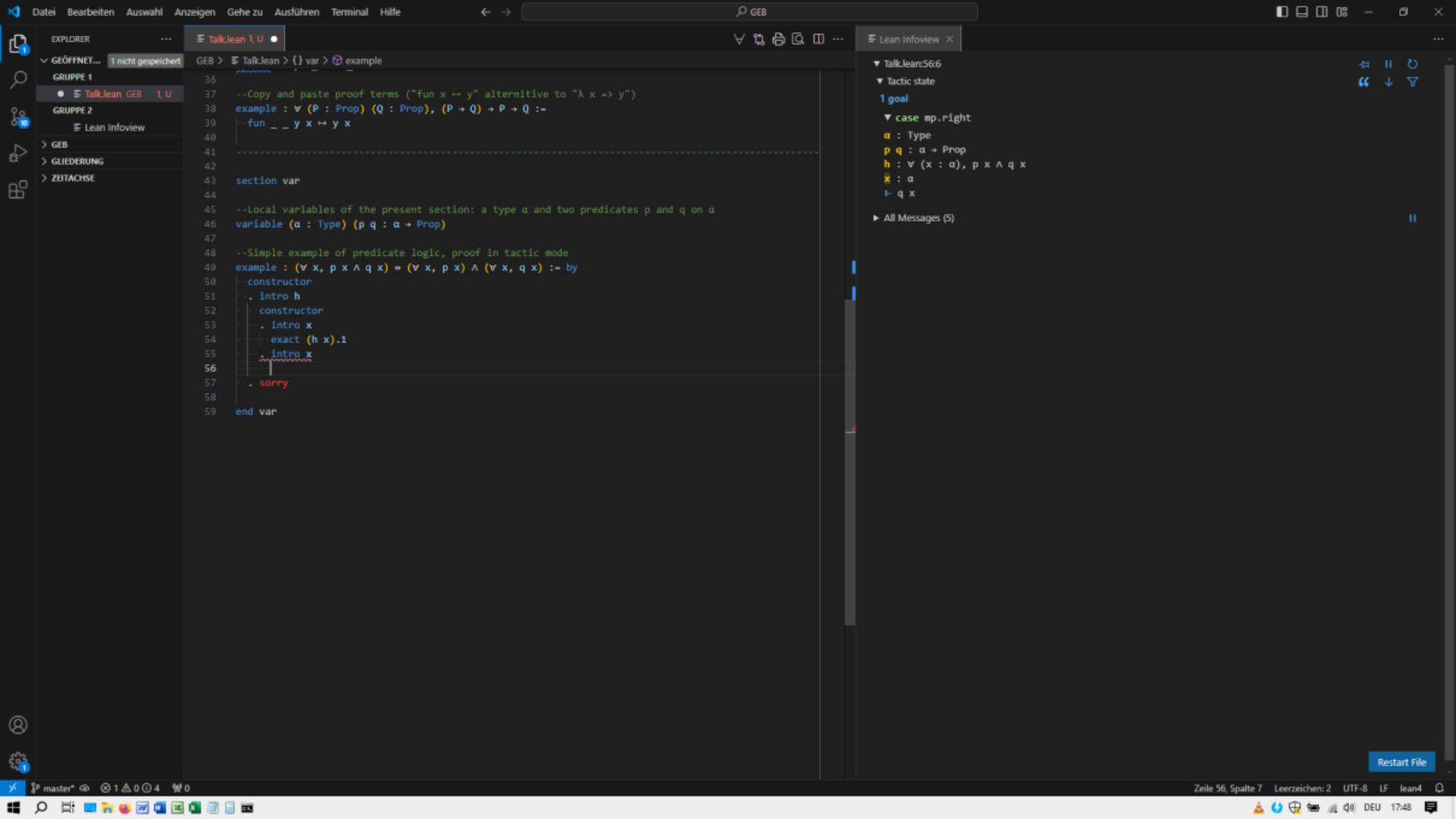
↑ ((forall (x : a), p x) ∧ forall (x : a), q x) + forall (x : a), p x ∧ q x

All Messages (7)

Restart File

master 3 ▲ 0 0 4

Zeil 55, Spalte 7 Leerzeichen: 2 UTF-8 LF lean4 DEU 17:48



File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

RESTART FILE

EXPLORER GLÖFFNET... 1 nicht gespeichert

GRUPPE 1 • TalkLean GEB 1, U

GRUPPE 2 Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

Talklean 1, U

GEB > Talklean > () var > example

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ {P : Prop} {Q : Prop}, (P → Q) + P + Q :=
39   fun _ _ y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
45 variable (a : Type) (p q : a → Prop)
46
47 --Simple example of predicate logic, proof in tactic mode
48 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
49   constructor
50   . intro h
51   . constructor
52   . . intro x
53   . . exact (h x).1
54   . . intro x
55   . . exact (h x).2
56   . . sorry
57
58 end var
```

Lean Infoview

Talklean:56:19

Tactic state

No goals

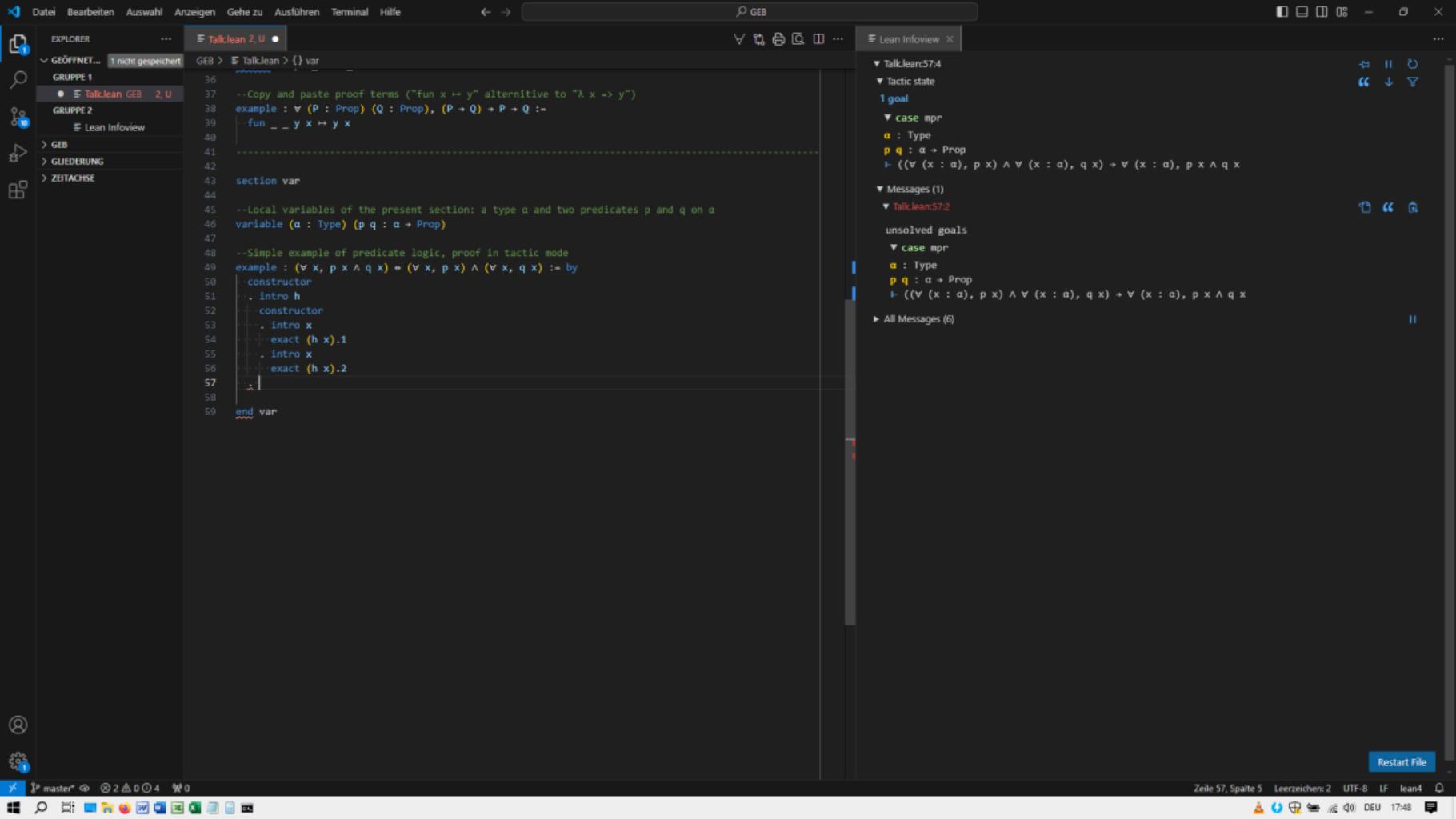
Expected type

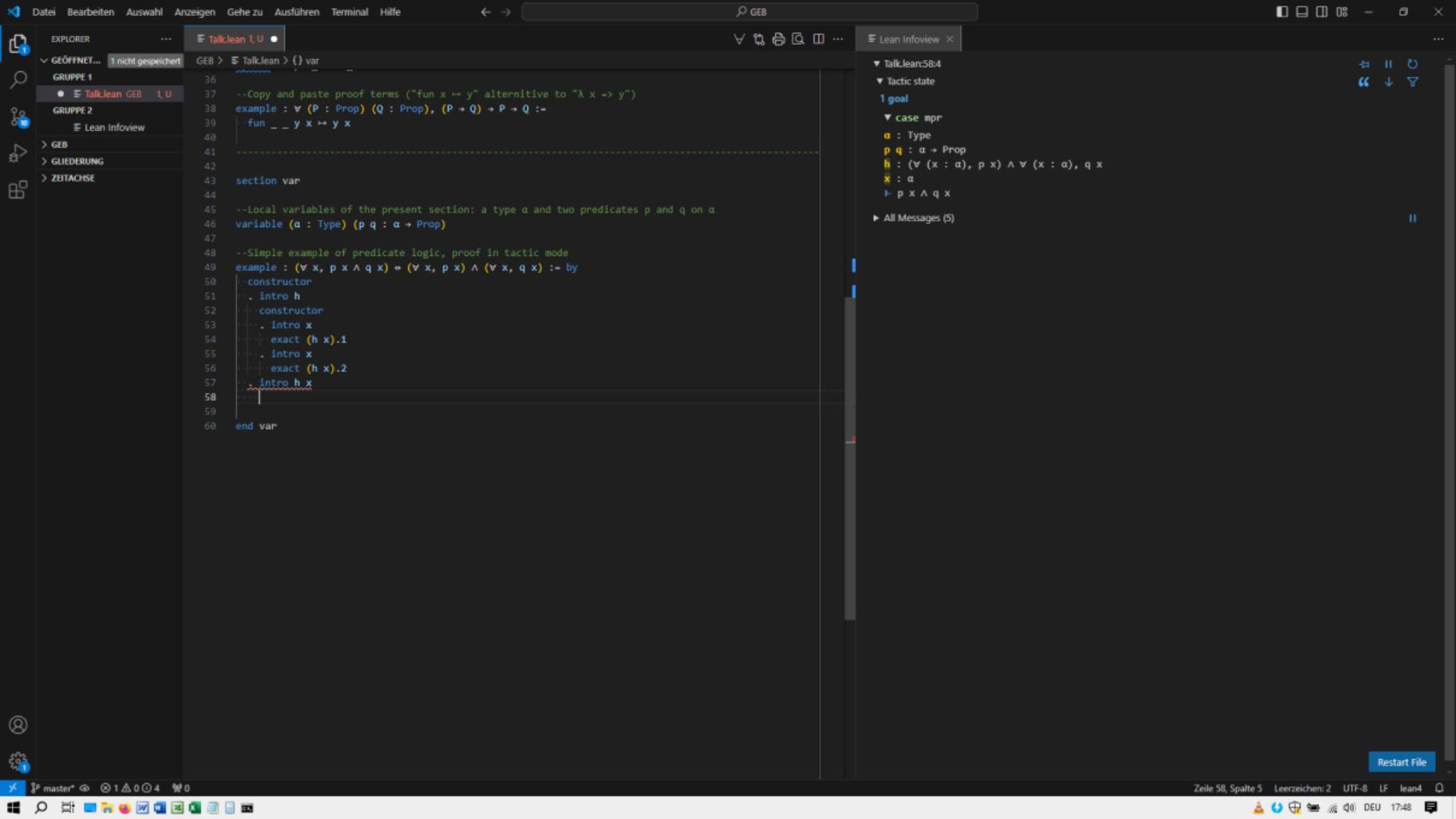
a : Type
p q : a → Prop
h : ∀ (x : a), p x ∧ q x
x : a
h : ∀ {a b : Prop}, a ∧ b → b

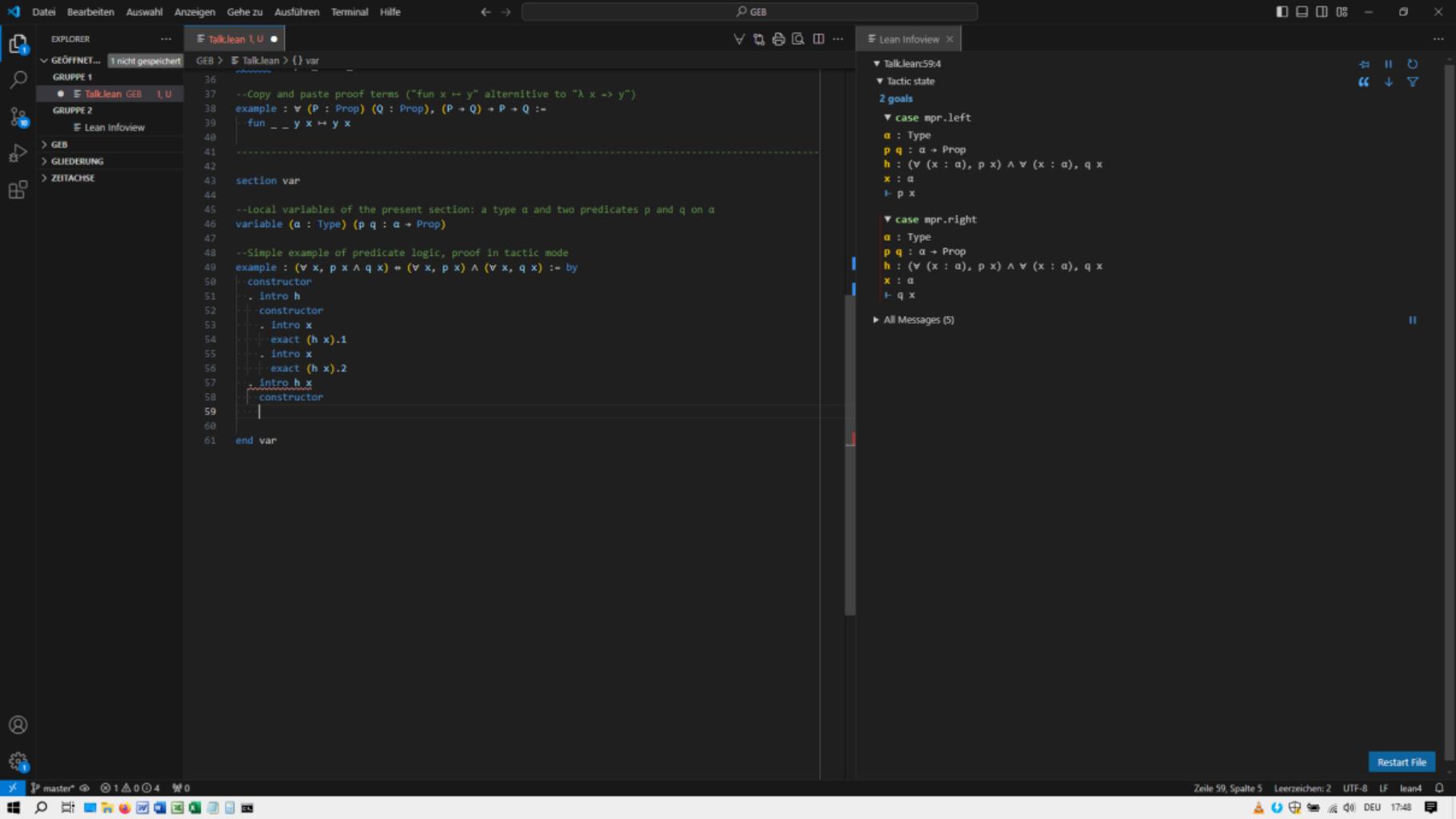
All Messages (5)

Zeile 56, Spalte 20 Leerzeichen: 2 UTF-8 LF lean4

Restart File







File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

RESTART FILE

EXPLORER ...

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1 • TalkLean GEB 1, U

GRUPPE 2

Lean Infoview

> GEB

> GUEDERUNG

> ZEITACHSE

Talklean 1, U

GEB > Talklean > () var > example

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q :=
39   fun _ _ y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
45 variable (a : Type) (p q : a → Prop)
46
47 --Simple example of predicate logic, proof in tactic mode
48 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
49   constructor
50   . intro h
51   . constructor
52   . . intro x
53   . . exact (h x).1
54   . . intro x
55   . . exact (h x).2
56   . . intro h x
57   . . constructor
58   . . sorry
59   . . sorry
60
61 end var
```

Lean Infoview

Talklean:60:11

Tactic state

No goals

All Messages (5)

Restart File

Zeile 60, Spalte 12 Leerzeichen: 2 UTF-8 LF lean4

Windows Taskbar

EXPLORER ...

GLÖFFNET... **Talklean 3, U** [nicht gespeichert]

GRUPPE 1 • **Talklean GEB** 3, U

GRUPPE 2

Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q :=
39 fun _ _ y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type α and two predicates p and q on α
45 variable (α : Type) (p q : $\alpha \rightarrow \text{Prop}$)
46
47 --Simple example of predicate logic, proof in tactic mode
48 example : ($\forall x, p \wedge q x$) + ($\forall x, p x$) \wedge ($\forall x, q x$) := by
49 constructor
50 . intro h
51 . constructor
52 . intro x
53 . exact (h x).1
54 . intro x
55 . exact (h x).2
56 . intro h x
57 . constructor
58 .
59 . sorry
60 .
61 end var

Lean Infoview

Talklean:59:6

Tactic state

1 goal

case mpr.left

α : Type

p, q : $\alpha \rightarrow \text{Prop}$

h : ($\forall (x : \alpha), p x$) \wedge ($\forall (x : \alpha), q x$)

x : α

$\vdash p x$

Messages (2)

Talklean:59:4

unsolved goals

case mpr.left

α : Type

p, q : $\alpha \rightarrow \text{Prop}$

h : ($\forall (x : \alpha), p x$) \wedge ($\forall (x : \alpha), q x$)

x : α

$\vdash p x$

Talklean:57:2

unsolved goals

case mpr.right

α : Type

p, q : $\alpha \rightarrow \text{Prop}$

h : ($\forall (x : \alpha), p x$) \wedge ($\forall (x : \alpha), q x$)

x : α

$\vdash q x$

All Messages (7)

Zeile 59, Spalte 7 Leerzeichen: 2 UTF-8 LF lean4

Restart File

DEU 17:49

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

RESTART FILE

EXPLORER ...

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1 • E TalkLean GEB 1, U

GRUPPE 2 E Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

E TalkLean 1, U

GEB > E TalkLean > () var > example

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ {P : Prop} {Q : Prop}, (P → Q) + P + Q :=
39   fun _ _ y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
45 variable (a : Type) (p q : a → Prop)
46
47 --Simple example of predicate logic, proof in tactic mode
48 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
49   constructor
50   . intro h
51   . constructor
52   . intro x
53   . exact (h x).1
54   . intro x
55   . exact (h x).2
56   . intro h x
57   . constructor
58   . exact h.1 x
59   . sorry
60
61 end var
```

Lean Infoview

TalkLean:59:17

Tactic state

No goals

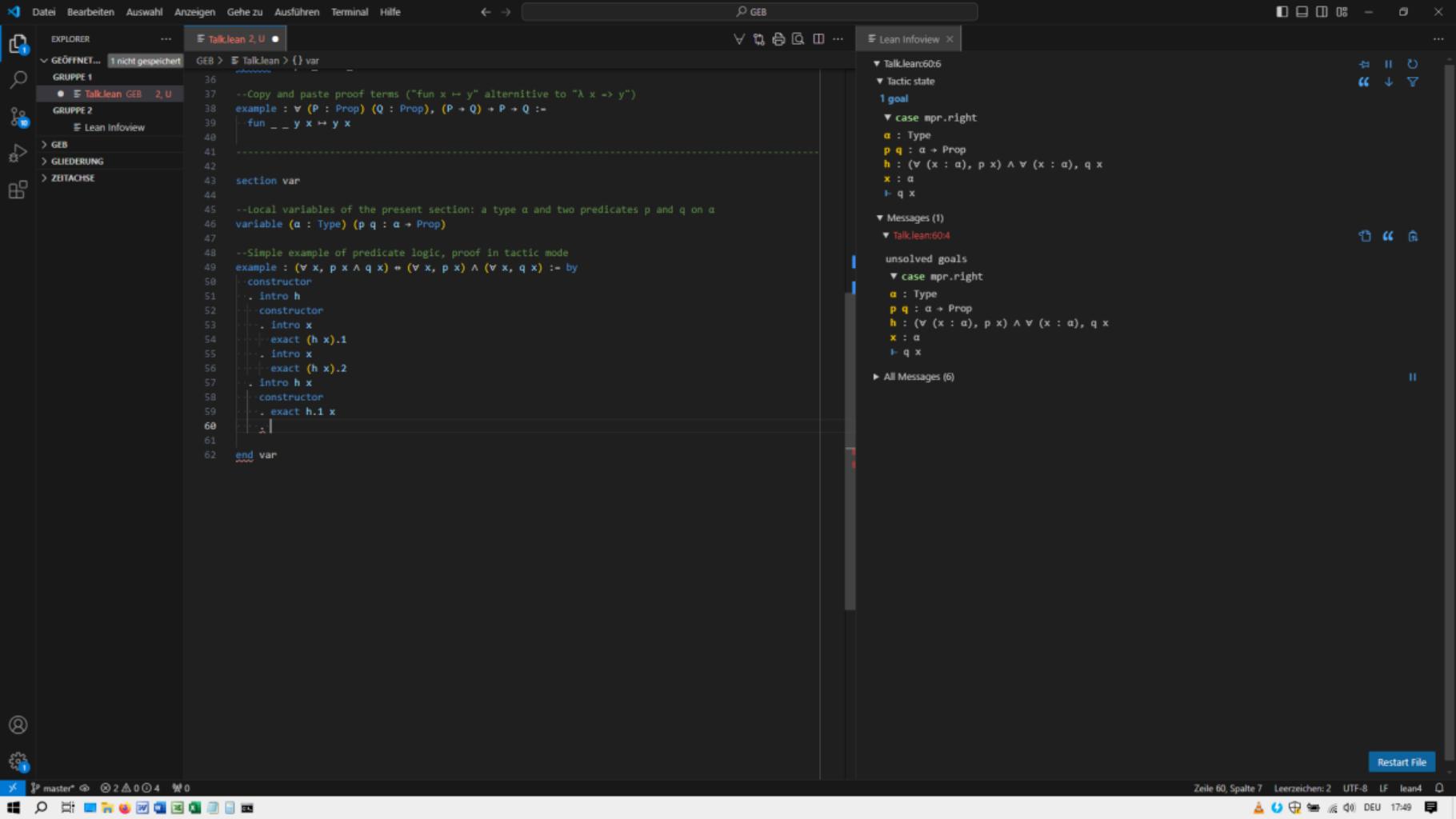
Expected type

a : Type
p q : a → Prop
h : (forall (x : a), p x) ∧ (forall (x : a), q x)
x : a
h : a

All Messages (5)

Zeile 59, Spalte 18 Leerzeichen: 2 UTF-8 LF lean4

Restart File



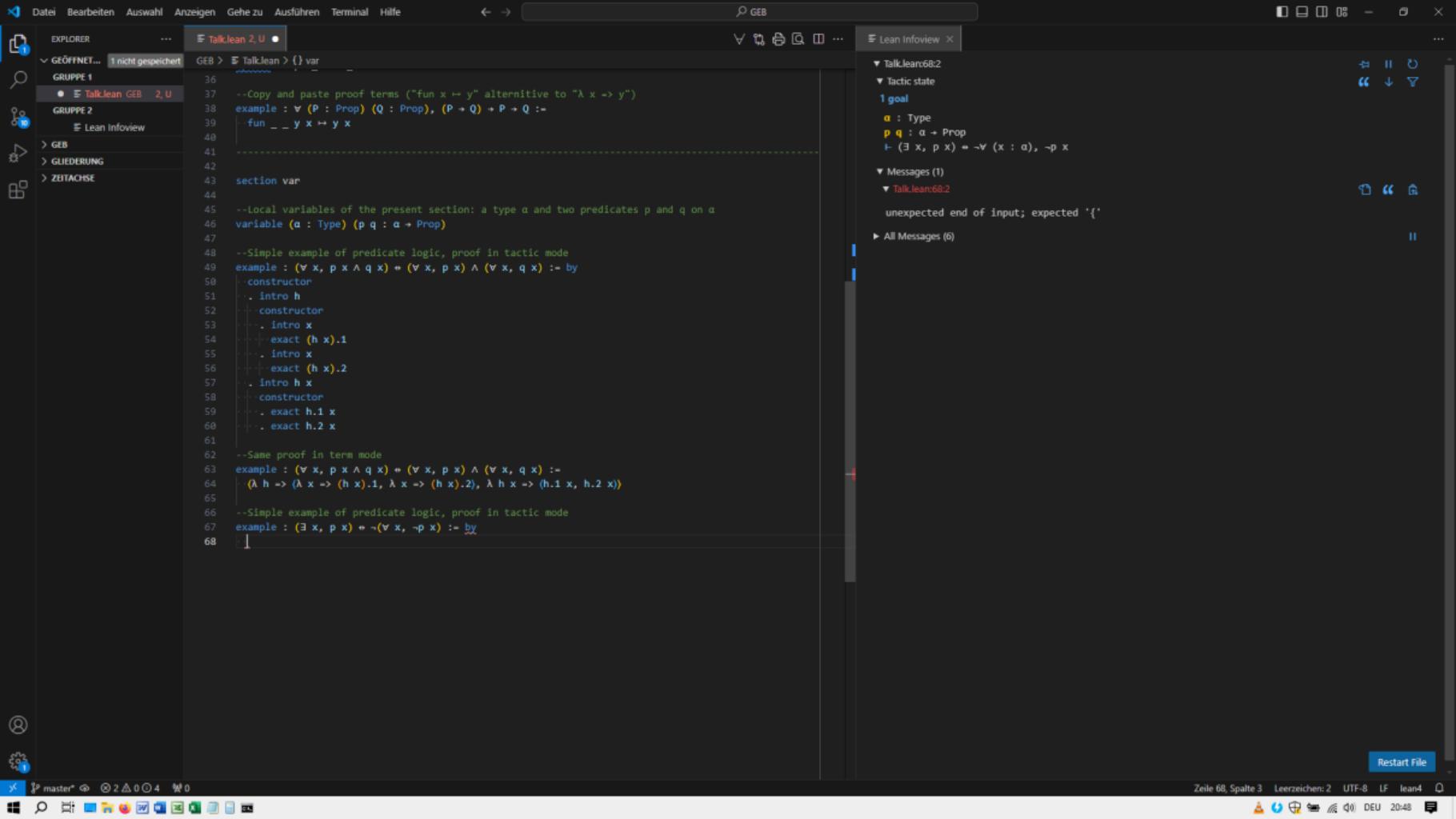


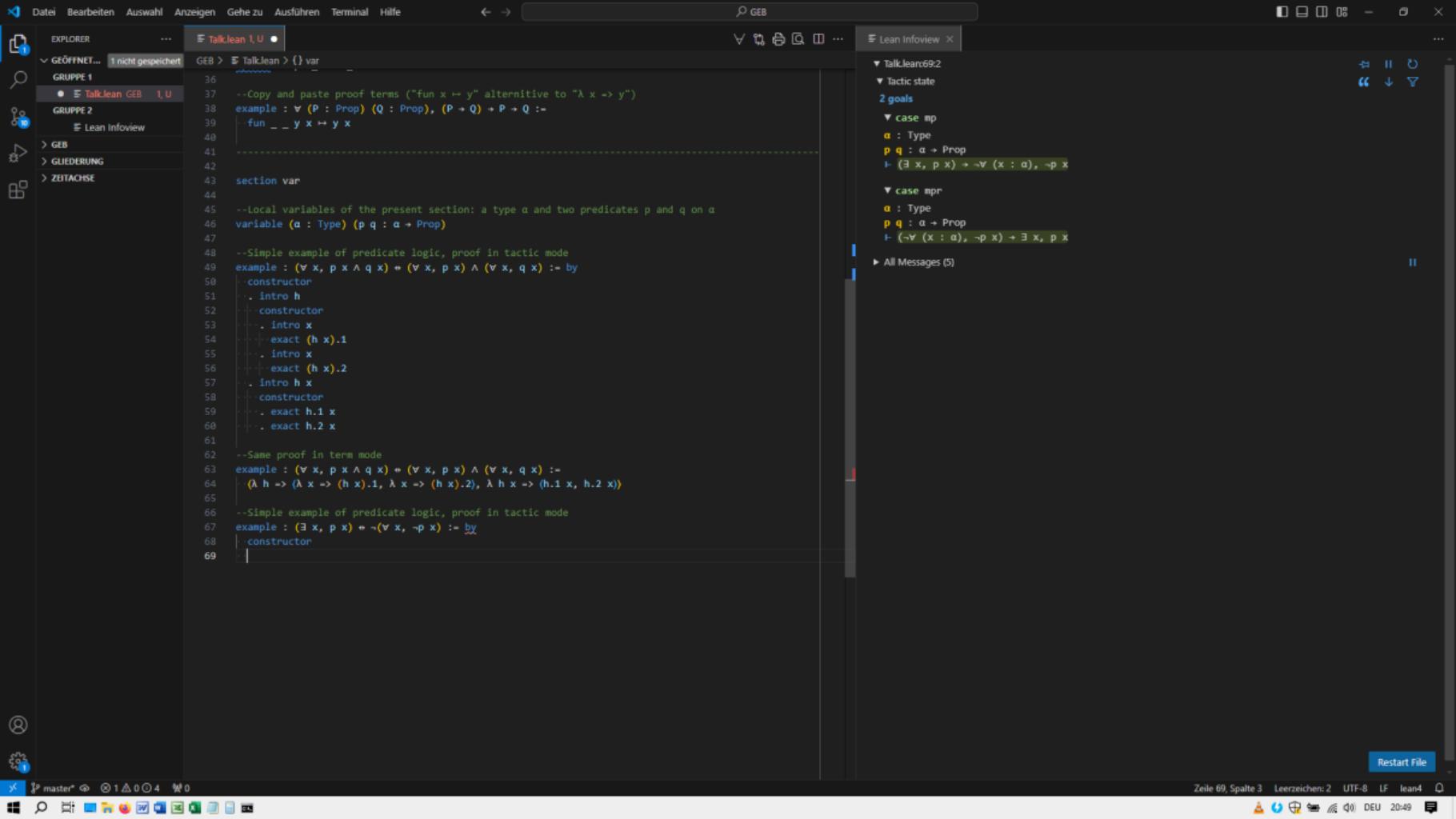
Lean Infoview
Talk.lean:61:
Tactic state
No goals
All Messages

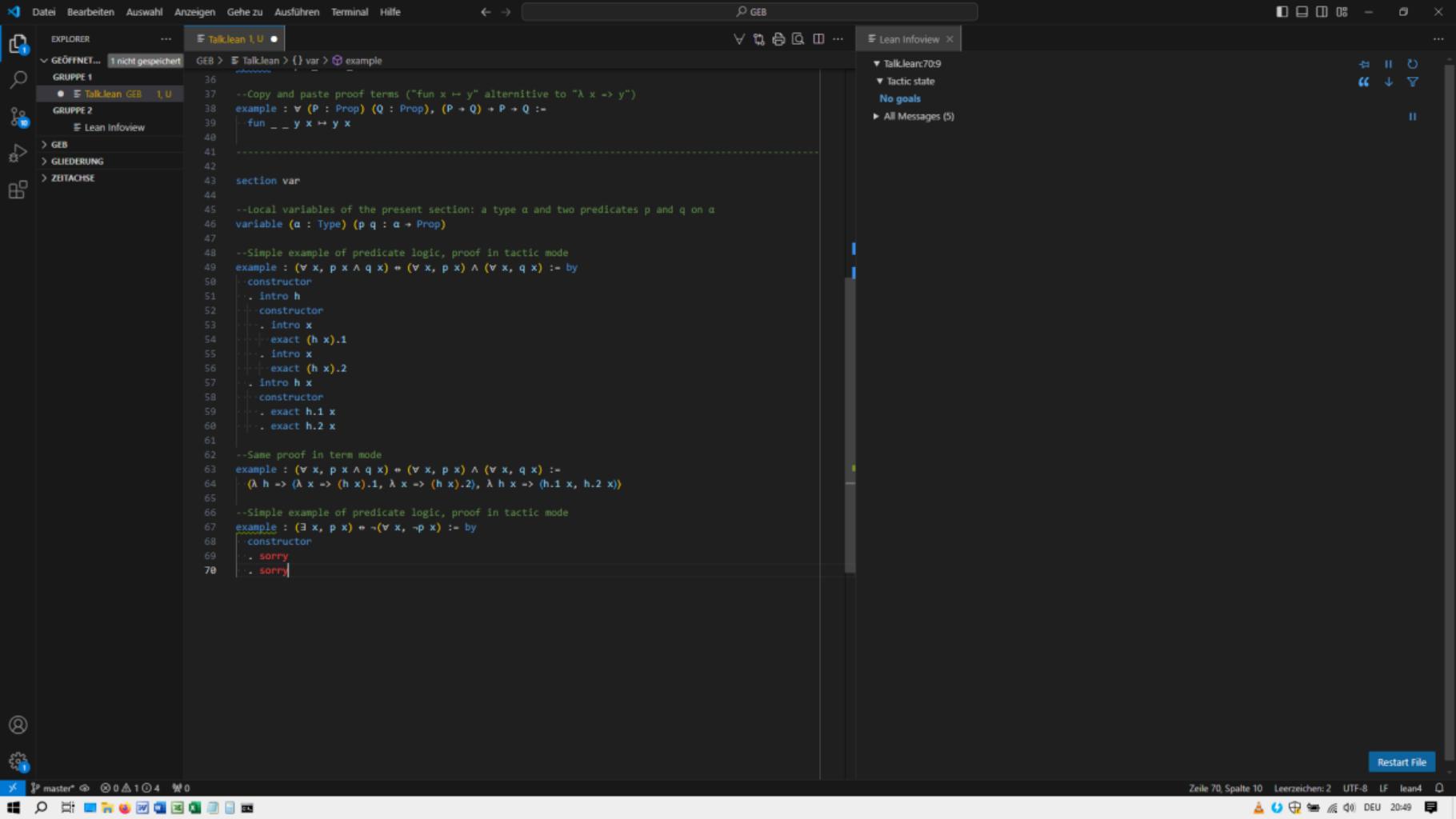


EXPLORER
GEÖFFNET... 1 nicht geöffnet
GRUPPE 1
• **TalkLean GEB**
GRUPPE 2
 Lean Infoview
GEB
GLIEDERUNG
ZEITACHSE

Lean Infoview
Talk.lean650
No info found
All Messages







File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB

EXPLORER

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E TalkLean GEB 3, U

GRUPPE 2

- Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

TalkLean 3, U

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q :=
39   fun _ _ y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
45 variable (a : Type) (p q : a → Prop)
46
47
48 --Simple example of predicate logic, proof in tactic mode
49 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
50   constructor
51   . intro h
52   . constructor
53   . intro x
54   . exact (h x).1
55   . intro x
56   . exact (h x).2
57   . intro h x
58   . constructor
59   . exact h.1 x
60   . exact h.2 x
61
62 --Same proof in term mode
63 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) :=
64   (λ h => (λ x => (h x).1, λ x => (h x).2), λ h x => (h.1 x, h.2 x))
65
66 --Simple example of predicate logic, proof in tactic mode
67 example : (exists x, p x) + ¬(forall x, ¬p x) := by
68   constructor
69   . intro x
70   . sorry
```

Lean Infoview

TalkLean:69:4

Tactic state

1 goal

case mp

a : Type

p q : a → Prop

↑ (exists x, p x) + ¬forall (x : a), ¬p x

Messages (2)

TalkLean:69:2

unsolved goals

case mp

a : Type

p q : a → Prop

↑ (exists x, p x) + ¬forall (x : a), ¬p x

TalkLean:67:39

unsolved goals

case mpr

a : Type

p q : a → Prop

↑ (¬forall (x : a), ¬p x) + exists x, p x

All Messages (7)

Restart File

master 3 ▲ 0 0 4 90

Zeile 69, Spalte 5 Leerzeichen: 2 UTF-8 LF lean4 DEU 20:50

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

RESTART FILE

EXPLORER ...

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1 • TalkLean GEB 1, U

GRUPPE 2 Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

Talklean 1, U

GEB > Talklean > () var > example

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q :=
39   fun _ _ y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
45 variable (a : Type) (p q : a → Prop)
46
47
48 --Simple example of predicate logic, proof in tactic mode
49 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
50   constructor
51   . intro h
52     constructor
53     . intro x
54     exact (h x).1
55     . intro x
56     exact (h x).2
57   . intro h x
58     constructor
59     . exact h.1 x
60     . exact h.2 x
61
62 --Same proof in term mode
63 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) :=
64   (λ h => (λ x => (h x).1, λ x => (h x).2), λ h x => (h.1 x, h.2 x))
65
66 --Simple example of predicate logic, proof in tactic mode
67 example : (exists x, p x) + (forall x, ¬p x) := by
68   constructor
69   . intro h ha
70   . sorry
```

Lean Infoview

TalkLean:704

Tactic state

1 goal

case imp

a : Type

p q : a → Prop

h : ∃ x, p x

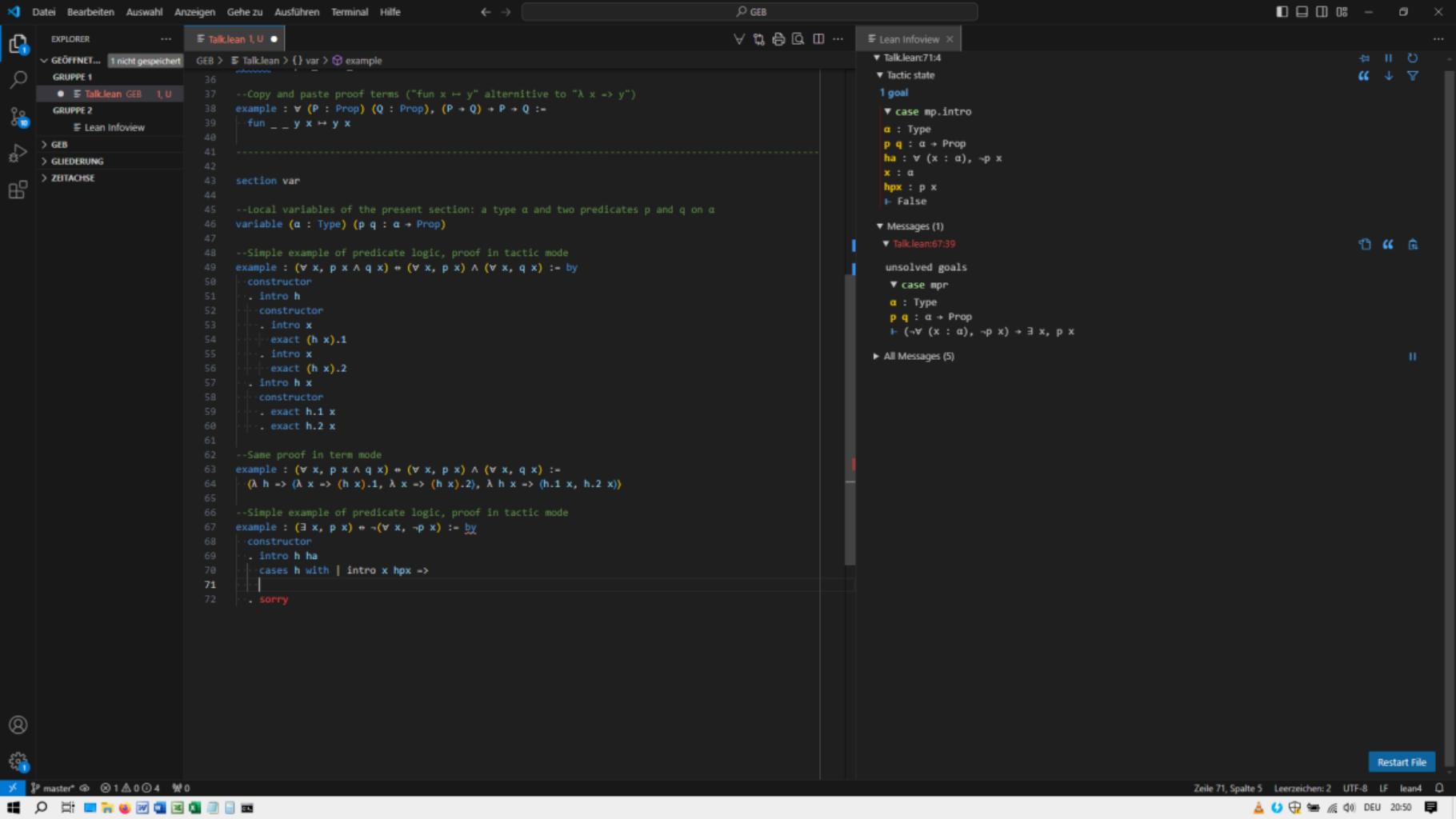
ha : ∀ (x : a), ¬p x

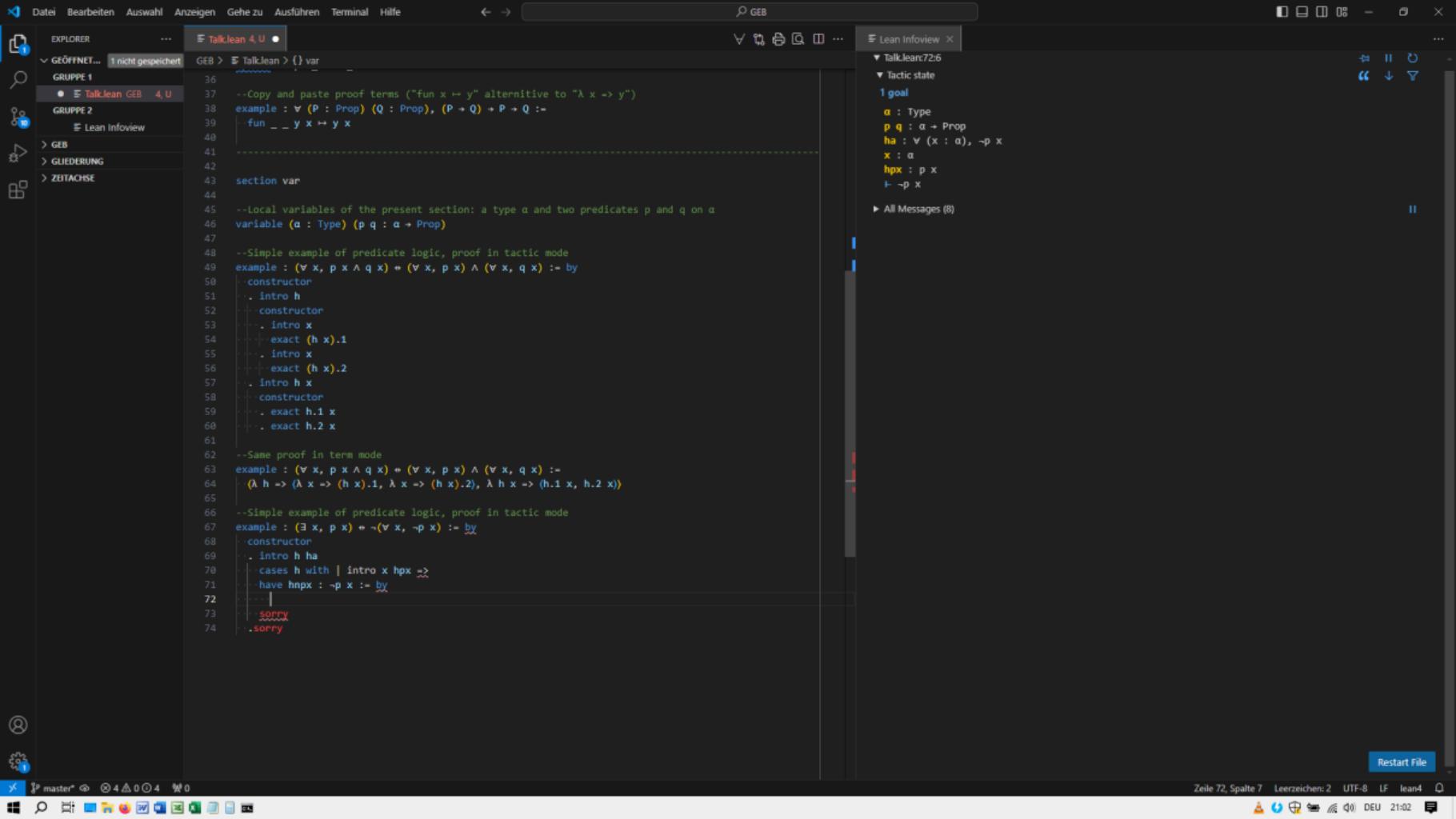
h : False

All Messages (5)

Restart File

Zeile 70, Spalte 5 Leerzeichen: 2 UTF-8 LF lean4





File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

RESTART FILE

EXPLORER ...

GlÖFFNET... **Talklean 1, U** [nicht gespeichert]

GEB > Talklean > () var > example

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ {P : Prop} {Q : Prop}, (P → Q) + P + Q :=
39   fun _ _ y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
45 variable (a : Type) (p q : a → Prop)
46
47 --Simple example of predicate logic, proof in tactic mode
48 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
49   constructor
50   . intro h
51   . constructor
52   . intro x
53   . exact (h x).1
54   . intro x
55   . exact (h x).2
56   . intro h x
57   . constructor
58   . exact h.1 x
59   . exact h.2 x
60
61 --Same proof in term mode
62 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) :=
63   (λ h => (λ x => (h x).1, λ x => (h x).2), λ h x => (h.1 x, h.2 x))
64
65 --Simple example of predicate logic, proof in tactic mode
66 example : (exists x, p x) + ¬(forall x, ¬p x) := by
67   constructor
68   . intro h ha
69   . cases h with | intro x hpx ↞
70   . have hnpfx : ¬p x := by
71   . exact ha x
72
73   |
74   . sorry
```

Lean Infoview

▼ Talklean:73:4

▼ Tactic state

1 goal

▼ case mp.intro

α : Type
p q : α → Prop
ha : ∀ (x : α), ¬p x
x : α
hpx : p x
hnpfx : ¬p x
─ False

All Messages (5)

Restart File

Zeile 73, Spalte 5 Leerzeichen: 2 UTF-8 LF lean4

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

RESTART FILE

EXPLORER

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E TalkLean GEB 1, U

GRUPPE 2

- Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

TalkLean 1, U

GEB > TalkLean > () var > example

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ {P : Prop} {Q : Prop}, (P → Q) + P + Q :=
39   fun _ _ y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
45 variable (a : Type) (p q : a → Prop)
46
47 --Simple example of predicate logic, proof in tactic mode
48 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
49   constructor
50   . intro h
51   . constructor
52   . intro x
53   . exact (h x).1
54   . intro x
55   . exact (h x).2
56   . intro h x
57   . constructor
58   . exact h.1 x
59   . exact h.2 x
60
61 --Same proof in term mode
62 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) :=
63   (λ h => (λ x => (h x).1, λ x => (h x).2), λ h x => (h.1 x, h.2 x))
64
65 --Simple example of predicate logic, proof in tactic mode
66 example : (exists x, p x) ∨ ¬(forall x, ¬p x) := by
67   constructor
68   . intro h ha
69   . intro x
70   . cases h with | intro x hpx =>
71     have hpx : ¬p x := by
72     | exact ha x
73     . exact hpx hpx
74     . sorry
```

Lean Infoview

TalkLean:73:18

Tactic state

No goals

Expected type

- a : Type
- p q : a → Prop
- ha : ∀ (x : a), ¬p x
- x : a
- hpx : p x
- hpx : ¬p x
- ¬p x

All Messages (5)

Restart File

Zeile 73, Spalte 19 Leerzeichen: 2 UTF-8 LF lean4 DEU 20:51

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

RESTART FILE

EXPLORER ...

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1 • Talklean GEB 2, U

GRUPPE 2 Lean Infoview

> GEB

> GLÜEDERUNG

> ZEITACHSE

Talklean 2, U

GEB > Talklean > () var

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q := 
39   fun _ -> y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
45 variable (a : Type) (p q : a → Prop)
46
47
48 --Simple example of predicate logic, proof in tactic mode
49 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
50   constructor
51   . intro h
52   . constructor
53   . intro x
54   . exact (h x).1
55   . intro x
56   . exact (h x).2
57   . intro h x
58   . constructor
59   . exact h.1 x
60   . exact h.2 x
61
62 --Same proof in term mode
63 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := 
64   (λ h => (λ x => (h x).1, λ x => (h x).2), λ h x => (h.1 x, h.2 x))
65
66 --Simple example of predicate logic, proof in tactic mode
67 example : (exists x, p x) + ¬(forall x, ¬p x) := by
68   constructor
69   . intro h ha
70   . cases h with | intro x hpx =>
71   . have hpx : ¬p x := by
72   . exact ha x
73   . exact hpx hpx
74
```

Lean Infoview

▼ Talklean:74:4

▼ Tactic state

1 goal

▼ case mpr

a : Type

p q : a → Prop

|- (¬∀ (x : a), ¬p x) + ∃ x, p x

▼ Messages (2)

▼ Talklean:74:2

unsolved goals

▼ case mpr

a : Type

p q : a → Prop

|- (¬∀ (x : a), ¬p x) + ∃ x, p x

▼ Talklean:74:4

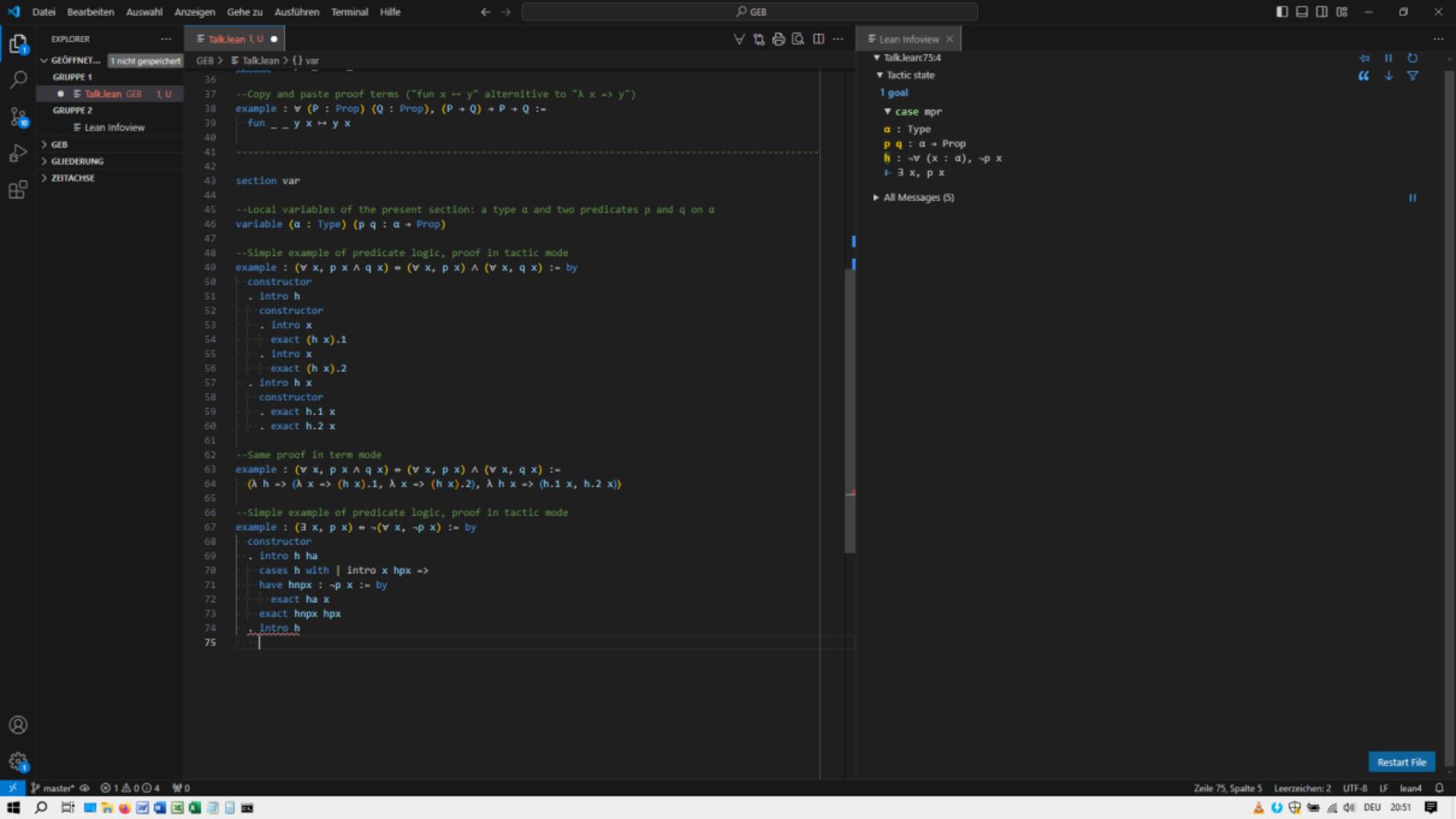
unexpected end of input; expected '{'

All Messages (6)

Restart File

master ④ ⑤ 2 ▲ 0 ① 4 ⌂ ⑥

Zeile 74, Spalte 5 Leerzeichen: 2 UTF-8 LF 20:51 DEU



File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB

EXPLORER

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1

- Talklean GEB 1, U

GRUPPE 2

- Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

Talklean 1, U

GEB > Talklean > () var

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q :=
39   fun _ _ y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
45 variable (a : Type) (p q : a → Prop)
46
47
48 --Simple example of predicate logic, proof in tactic mode
49 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
50   constructor
51   . intro h
52     constructor
53     . intro x
54       exact (h x).1
55     . intro x
56       exact (h x).2
57   . intro h x
58     constructor
59     . exact h.1 x
60     . exact h.2 x
61
62 --Same proof in term mode
63 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) :=
64   (λ h => (λ x => (h x).1, λ x => (h x).2), λ h x => (h.1 x, h.2 x))
65
66 --Simple example of predicate logic, proof in tactic mode
67 example : (exists x, p x) + ¬(forall x, ¬p x) := by
68   constructor
69   . intro h ha
70     cases h with | intro x hpx =>
71       have hpx : ¬p x := by
72         exact ha x
73       exact hpx hpx
74     intro h
75     apply Classical.byContradiction
76
```

Lean Infoview

Talklean:764

Tactic state

1 goal

case mpr.h

a : Type

p q : a → Prop

h : ¬r (x : a), ¬p x

¬(¬exists x, p x) + False

All Messages (5)

Restart File

File 1 ▲ 0 0 4 W 0

Zeile 76, Spalte 5 Leerzeichen: 2 UTF-8 LF lean4 DEU DEU 20:52

EXPLORER ...

GLÖFFNET... **Talklean 1, U** [nicht gespeichert]

GRUPPE 1

- **TalkLean GEB 1, U**

GRUPPE 2

- Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

Talklean 1, U

GEB > **Talklean > () var**

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ {P : Prop} {Q : Prop}, (P → Q) + P + Q :=
39   fun _ _ y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
45 variable (a : Type) (p q : a → Prop)
46
47 --Simple example of predicate logic, proof in tactic mode
48 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
49   constructor
50   . intro h
51   . constructor
52   . intro x
53   . exact (h x).1
54   . intro x
55   . exact (h x).2
56   . intro h x
57   . constructor
58   . exact h.1 x
59   . exact h.2 x
60
61 --Same proof in term mode
62 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) :=
63   (λ h => (λ x => (h x).1, λ x => (h x).2), λ h x => (h.1 x, h.2 x))
64
65 --Simple example of predicate logic, proof in tactic mode
66 example : (exists x, p x) ∨ ¬(forall x, ¬p x) := by
67   constructor
68   . intro h ha
69   . cases h with | intro x hpx =>
70   . have hpx : ¬p x := by
71   . exact ha x
72   . exact hpx hpx
73   . intro h
74   . apply Classical.byContradiction
75   . intro hnex
76   . intro hnex
77
```

Lean Infoview

▼ Talklean:77:4

▼ Tactic state

1 goal

▼ case mpr.h

a : Type

p q : a → Prop

h : ¬r (x : a), ¬p x

hnex : ¬¬r x, p x

↳ False

All Messages (5)

Restart File

Zeile 77, Spalte 5 Leerzeichen: 2 UTF-8 LF lean4

master 1 0 0 4 W0

Windows Taskbar icons

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

RESTART FILE

Explorer

GlÖFFNET... **Talklean 3, U** [nicht gespeichert]

GRUPPE 1

- **TalkLean** GEB 3, U

GRUPPE 2

- Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

TALKLEAN 3, U

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q :=
39   fun _ _ y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
45 variable (a : Type) (p q : a → Prop)
46
47
48 --Simple example of predicate logic, proof in tactic mode
49 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
50   constructor
51   . intro h
52     constructor
53     . intro x
54       exact (h x).1
55     . intro x
56       exact (h x).2
57   . intro h x
58     constructor
59     . exact h.1 x
60     . exact h.2 x
61
62 --Same proof in term mode
63 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) :=
64   (λ h => (λ x => (h x).1, λ x => (h x).2), λ h x => (h.1 x, h.2 x))
65
66 --Simple example of predicate logic, proof in tactic mode
67 example : (exists x, p x) + ¬(forall x, ¬p x) := by
68   constructor
69   . intro h ha
70     cases h with | intro x hpx =>
71       have hpx : ¬p x := by
72         exact ha x
73       exact hpx hpx
74     . intro h
75     apply Classical.byContradiction
76     intro hnx
77     have ha : ∀ (x : a), ¬p x := by
78     |
79       sorry
```

Lean Infoview

TalkLean:78:6

Tactic state

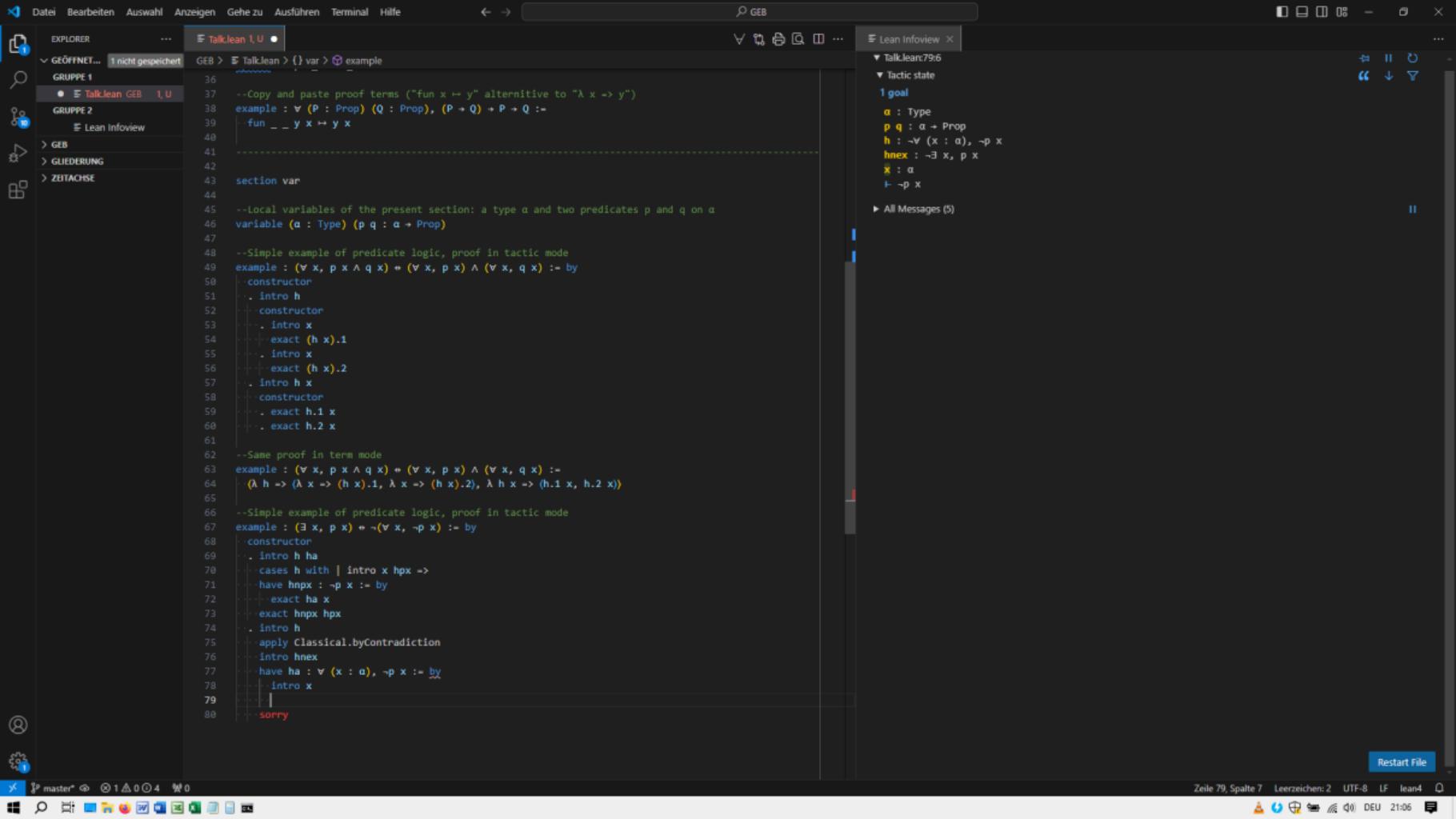
1 goal

a : Type
p q : a → Prop
h : ¬r (x : a), ¬p x
hnex : ¬r x, p x
l- r (x : a), ¬p x

All Messages (7)

Restart File

Zeile 78, Spalte 7 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:06



File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

RESTART FILE

EXPLORER ...

GLÖFFNET... [nicht gespeichert]

GRUPPE 1 • Talklean GEB 1, U

GRUPPE 2

Lean Infoview

> GEB

> GLÜEDERUNG

> ZEITACHSE

Talklean 1, U

GEB > Talklean > () var > example

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q :=
39   fun _ -> y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
45 variable (a : Type) (p q : a → Prop)
46
47 --Simple example of predicate logic, proof in tactic mode
48 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
49   constructor
50   . intro h
51   . constructor
52   . intro x
53   . exact (h x).1
54   . intro x
55   . exact (h x).2
56   . intro h x
57   . constructor
58   . exact h.1 x
59   . exact h.2 x
60
61 --Same proof in term mode
62 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) :=
63   (λ h => (λ x => (h x).1, λ x => (h x).2), λ h x => (h.1 x, h.2 x))
64
65 --Simple example of predicate logic, proof in tactic mode
66 example : (exists x, p x) + ¬(forall x, ¬p x) := by
67   constructor
68   . intro h ha
69   . intro h ha
70   . cases h with | intro x hpx =>
71   . have hpx : ¬p x := by
72   . exact ha x
73   . exact hpx hpx
74   . intro h
75   . apply Classical.byContradiction
76   . intro hnx
77   . have ha : ∀ (x : a), ¬p x := by
78   . intro x
79   . intro hpx
80   . |
81   . sorry
```

Lean Infoview

Talklean:806

Tactic state

1 goal

a : Type
p q : a → Prop
h : ∀ (x : a), ¬p x
hnex : ¬∃ x, p x
x : a
hpx : p x
← False

All Messages (5)

Restart File

Zeile 80, Spalte 7 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:07

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

EXPLORER ...

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1 • TalkLean GEB 4, U

GRUPPE 2 Lean Infoview

> GEB

> GLÜEDERUNG

> ZEITACHSE

Talklean 4, U

GEB > Talklean > () var

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q :=
39   fun _ -> y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
45 variable (a : Type) (p q : a → Prop)
46
47 --Simple example of predicate logic, proof in tactic mode
48 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
49   constructor
50   . intro h
51   . constructor
52   . intro x
53   . exact (h x).1
54   . intro x
55   . exact (h x).2
56   . intro h x
57   . constructor
58   . exact h.1 x
59   . exact h.2 x
60
61 --Same proof in term mode
62 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) :=
63   (λ h => (λ x => (h x).1, λ x => (h x).2), λ h x => (h.1 x, h.2 x))
64
65 --Simple example of predicate logic, proof in tactic mode
66 example : (exists x, p x) + ¬(forall x, ¬p x) := by
67   constructor
68   . intro h ha
69   . intro x
70   . cases h with | intro x hpx =>
71   . have hpx : ¬p x := by
72   . exact ha x
73   . exact hpx hpx
74   . intro h
75   . apply Classical.byContradiction
76   . intro hnx
77   . have ha : ∀ (x : a), ¬p x := by
78   . intro x
79   . intro hpx
80   . have hnx : ∃ x, p x := by
81   . sorry
82   . sorry
83   . sorry
```

Lean Infoview

TalkLean:81:8

Tactic state

1 goal

a : Type

p q : a → Prop

h : ¬r (x : a), ¬p x

hnex : ¬∃ x, p x

x : a

hpx : p x

l- ∃ x, p x

All Messages (8)

Restart File

Zeile 81, Spalte 9 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:07

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1 • TalkLean GEB 1, U

GRUPPE 2 Lean Infoview

> GEB

> GLÜEDERUNG

> ZEITACHSE

Talklean 1, U

GEB > Talklean > () var > example

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q :=
39   fun _ -> y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
45 variable (a : Type) (p q : a → Prop)
46
47 --Simple example of predicate logic, proof in tactic mode
48 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
49   constructor
50   . intro h
51   . constructor
52   . intro x
53   . exact (h x).1
54   . intro x
55   . exact (h x).2
56   . intro h x
57   . constructor
58   . exact h.1 x
59   . exact h.2 x
60
61 --Same proof in term mode
62 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) :=
63   (λ h => (λ x => (h x).1, λ x => (h x).2), λ h x => (h.1 x, h.2 x))
64
65 --Simple example of predicate logic, proof in tactic mode
66 example : (exists x, p x) ∨ ¬(forall x, ¬p x) := by
67   constructor
68   . intro h ha
69   . intro x
70   . cases h with | intro x hpx =>
71   . have hpx : ¬p x := by
72   . exact ha x
73   . exact hpx hpx
74   . intro h
75   . apply Classical.byContradiction
76   . intro hnx
77   . have ha : ∀ (x : a), ¬p x := by
78   . intro x
79   . intro hpx
80   . have hex : ∃ x, p x := by
81   . use x
82   . sorry
```

Lean Infoview

TalkLean:82:6

Tactic state

1 goal

a : Type
p q : a → Prop
h : ¬r (x : a), ¬p x
hnex : ¬∃ x, p x
x : a
hpx : p x
hex : ∃ x, p x
false

All Messages (5)

Restart File

master 1 ▲ 0 0 4 90

Zeile 82, Spalte 7 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:07

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

RESTART FILE

EXPLORER

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1 • TalkLean GEB 1, U

GRUPPE 2 Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

Talklean 1, U

GEB > Talklean > () var

```
36
37 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
38 example : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q :=
39   fun _ _ y x ↦ y x
40
41
42 section var
43
44 --Local variables of the present section: a type a and two predicates p and q on a
45 variable (a : Type) (p q : a → Prop)
46
47 --Simple example of predicate logic, proof in tactic mode
48 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) := by
49   constructor
50   . intro h
51   . constructor
52   . intro x
53   . exact (h x).1
54   . intro x
55   . exact (h x).2
56   . intro h x
57   . constructor
58   . exact h.1 x
59   . exact h.2 x
60
61 --Same proof in term mode
62 example : (forall x, p x ∧ q x) + (forall x, p x) ∧ (forall x, q x) :=
63   (λ h => (λ x => (h x).1, λ x => (h x).2), λ h x => (h.1 x, h.2 x))
64
65 --Simple example of predicate logic, proof in tactic mode
66 example : (exists x, p x) + ¬(forall x, ¬p x) := by
67   constructor
68   . intro h ha
69   . intro h ha
70   . cases h with | intro x hpx =>
71   . have hnxp : ¬p x := by
72   . exact ha x
73   . exact hnxp hpx
74   . intro h
75   . apply Classical.byContradiction
76   . intro hnx
77   . have ha : ∀ (x : a), ¬p x := by
78   . intro x
79   . intro hpx
80   . have hex : ∃ x, p x := by
81   . . use x
82   . exact hnx hex
83
```

Lean Infoview

TalkLean:83/4

Tactic state

1 goal

case mpr.h

a : Type

p q : a → Prop

h : ¬r (x : a), ¬p x

hnx : ¬∃ x, p x

ha : ∀ (x : a), ¬p x

|- False

All Messages (5)

Restart File

Zeile 83, Spalte 5 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:08

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

RESTART FILE

Explorer ...

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1 • Talklean GEB U

GRUPPE 2

Lean Infoview

GEB

GUEDERUNG

ZETACHSE

Talklean U

36 --Copy and paste proof terms ("fun x ↦ y" alternative to "λ x ↦ y")
37 example : ∀ (P : Prop) (Q : Prop), (P → Q) + P + Q :=
38 fun _ _ y x ↦ y x
39
40
41 section var
42
43 --Local variables of the present section: a type a and two predicates p and q on a
44 variable (a : Type) (p q : a → Prop)
45
46 --Simple example of predicate logic, proof in tactic mode
47 example : (v x, p x ∧ q x) + (v x, p x) ∧ (v x, q x) := by
48 constructor
49 . intro h
50 . constructor
51 . intro x
52 . exact (h x).1
53 . intro x
54 . exact (h x).2
55 . intro h x
56 . constructor
57 . exact h.1 x
58 . exact h.2 x
59
60
61 --Same proof in term mode
62 example : (v x, p x ∧ q x) + (v x, p x) ∧ (v x, q x) :=
63 (λ h => (λ x => (h x).1, λ x => (h x).2), λ h x => (h.1 x, h.2 x))
64
65
66 --Simple example of predicate logic, proof in tactic mode
67 example : (exists x, p x) + ¬(v x, ¬p x) := by
68 constructor
69 . intro h ha
70 . cases h with | intro x hpx =>
71 . have hnxpx : ¬p x := by
72 . exact ha x
73 . exact hnxpx hpx
74 . intro h
75 . apply Classical.byContradiction
76 . intro hnx
77 . have ha : v (x : a), ¬p x := by
78 . intro x
79 . intro hpx
80 . have hex : ∃ x, p x := by
81 . use x
82 . exact hnx hex
83 . exact ha
84

Lean Infoview

Talklean:84:0

Tactic state

No goals

All Messages (4)

Restart File

Zeile 84, Spalte 1 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:08

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

GRUPPE 1

GRUPPE 2

> GEB

> GUEDERUNG

> ZETACHSE

EXPLORER

Talklean GEB

Lean Infoview

section var

```
65 --Simple example of predicate logic, proof in tactic mode
66 example : ( $\exists x, p x$ )  $\Rightarrow$  ( $\forall x, \neg p x$ ) := by
67   constructor
68   . intro h ha
69   . cases h with | intro x hpx =>
70     have hpx :  $\neg p x$  := by
71       exact ha x
72       exact hpx hpx
73     . intro h
74     apply Classical.byContradiction
75     intro hnex
76     have ha :  $\forall (x : a), \neg p x$  := by
77       intro x
78       intro hpx
79       have hex :  $\exists x, p x$  := by
80         .use x
81         .exact hnex hex
82       exact hnex hex
83     exact h ha
84
85 --Same proof in term mode
86 lemma ex_iff : ( $\exists x, p x$ )  $\Leftrightarrow$  ( $\forall x, \neg p x$ ) :=
87   (λ h ha => Exists.elim h (λ hpx => ha x hpx),
88    λ h => Classical.byContradiction (λ hnex => h (λ x hpx => hnex (x, hpx))))
89
90 end var
91
```

Lean Infoview

No info found.

All Messages (4)

Restart File

Zeile 91, Spalte 1 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:10

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

REPL GEB > Talklean > ...

GRUPPE 1 • Talklean GEB U

GRUPPE 2 Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

EXPLORER

Talklean U ●

43 section var

84

85 --Same proof in term mode

86 lemma ex_iff : (exists x, p x) = not(exists x, not p x) :=

87 (λ h ha => exists.elim h (λ x hpx => ha x hpx),

88 λ h => classical.byContradiction (λ hnex => h (λ x hpx => hnex (x, hpx))))

89

90 end var

91

92

93

94 --A simple inductive definition with constant constructors

95 inductive Weekday where

96 | sunday

97 | monday

98 | tuesday

99 | wednesday

100 | thursday

101 | friday

102 | saturday

103 deriving Repr --Technically (represent elements of Weekday by their respective names)

104

Lean Infoview

▼ Talklean:104:0

No info found.

► All Messages (4)

Restart File

master* 0 0 0 0 4 0 0 Zeile 104, Spalte 1 Leerzeichen:2 UTF-8 LF lean4

Windows Taskbar Icons

Zeile 104, Spalte 1 Leerzeichen:2 UTF-8 LF DEU 21:15

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

Explorer ...

GLÖFFNET... [nicht gespeichert] GEB > Talklean > ...

GRUPPE 1 ● Talklean GEB U

GRUPPE 2

Lean Infoview

> GEB

> GLÜEDERUNG

> ZEITACHSE

Talklean U

```
43 section var
44
45 --Same proof in term mode
46 lemma ex_iff : (exists x, p x) = not(exists x, not p x) := 
47   let h ha := exists.elim h (λ x hpx => ha x hpx),
48   λ h => Classical.byContradiction (λ hnex => h (hnex x, hpx)))
49
50 end var
51
52
53
54 --A simple inductive definition with constant constructors
55 inductive Weekday where
56   | sunday
57   | monday
58   | tuesday
59   | wednesday
60   | thursday
61   | friday
62   | saturday
63   deriving Repr --Technically represent elements of Weekday by their respective names
64
65 open Weekday --Open namespace Weekday (Weekday.monday -> monday)
66
67 --A simple functions on Weekday
68 def next_day (d : Weekday) : Weekday := by
69   cases d with
70     | sunday => exact monday
71     | monday => exact tuesday
72     | tuesday => exact wednesday
73     | wednesday => exact thursday
74     | thursday => exact friday
75     | friday => exact saturday
76     | saturday => exact sunday
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
```

Lean Infoview

▼ Talklean:117:0

▼ Tactic state

No goals

▶ All Messages (4)

Restart File

Zeile 117, Spalte 1 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:19

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

REPL master* 0 ▲ 0 ◑ 4 ⌂ 0

EXPLORER ...

GLÖFFNET... [nicht gespeichert]

GRUPPE 1 ● Talklean GEB U

GRUPPE 2

Lean Infoview

> GEB

> GLÜEDERUNG

> ZEITACHSE

Talklean U

GEB > Talklean > ...

section var

84

--Same proof in term mode

85 lemma ex_iff : (exists x, p x) = ~forall x, ~p x :=

86 (lambda h ha => exists.elim h (lambda x hpx => ha x hpx),

87 lambda h => Classical.byContradiction (lambda hnex => h (lambda x hpx => hnex (x, hpx))))

88

89 end var

90

91

92

93

--A simple inductive definition with constant constructors

94 inductive Weekday where

95 | sunday

96 | monday

97 | tuesday

98 | wednesday

99 | thursday

100 | friday

101 | saturday

102 deriving Repr --Technically represent elements of Weekday by their respective names

103

104 open Weekday --Open namespace Weekday (Weekday.monday -> monday)

105

106

--A simple functions on Weekday

107 def next_day (d : Weekday) : Weekday := by

108 cases d with

109 | sunday => exact monday

110 | monday => exact tuesday

111 | tuesday => exact wednesday

112 | wednesday => exact thursday

113 | thursday => exact friday

114 | friday => exact saturday

115 | saturday => exact sunday

116

117

--Another simple functions on Weekday

118 def prev_day (d : Weekday) : Weekday := by

119 cases d with

120 | sunday => exact saturday

121 | monday => exact sunday

122 | tuesday => exact monday

123 | wednesday => exact tuesday

124 | thursday => exact wednesday

125 | friday => exact thursday

126 | saturday => exact friday

127

128

Lean Infoview

Talklean:128:0

Tactic state

No goals

All Messages (4)

Restart File

Zeile 128, Spalte 1 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:19

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

REPL GEB > Talklean > ...

GRUPPE 1

- E Talklean GEB U

GRUPPE 2

- E Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

EXPLORER

Talklean U ●

```
1 nicht gespeichert
43   section var
44
45   -- Same proof in term mode
46   lemma ex_iff : (exists x, p x) = ~forall x, ~p x := 
47     (lambda h ha => exists.elim h (lambda x hpx => ha x hpx),
48      lambda h => classical.byContradiction (lambda hnex => h (lambda x hpx => hnex (x, hpx))))
49
50   end var
51
52
53
54   --A simple inductive definition with constant constructors
55   inductive Weekday where
56     | sunday
57     | monday
58     | tuesday
59     | wednesday
60     | thursday
61     | friday
62     | saturday
63     deriving Repr --Technically represent elements of Weekday by their respective names)
64
65   open Weekday --Open namespace Weekday (Weekday.monday -> monday)
66
67   --A simple functions on Weekday
68   def next_day (d : Weekday) : Weekday := by
69     cases d with
70       | sunday => exact monday
71       | monday => exact tuesday
72       | tuesday => exact wednesday
73       | wednesday => exact thursday
74       | thursday => exact friday
75       | friday => exact saturday
76       | saturday => exact sunday
77
78   --Another simple functions on Weekday
79   def prev_day (d : Weekday) : Weekday := by
80     cases d with
81       | sunday => exact saturday
82       | monday => exact sunday
83       | tuesday => exact monday
84       | wednesday => exact tuesday
85       | thursday => exact wednesday
86       | friday => exact thursday
87       | saturday => exact friday
88
89   #check sunday
```

Lean Infoview

- ▼ Talklean:129:13
- ▼ Expected type
- ↳ Weekday
- ▼ Messages (1)
- ▼ Talklean:129:0
- Weekday.sunday : Weekday
- All Messages (5)

Restart File

Zeile 129, Spalte 14 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:21

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

REPL

EXPLORER

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E Talklean GEB U

GRUPPE 2

- E Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

Talklean U

```
43 section var
44
45 -- Same proof in term mode
46 lemma ex_iff : (exists x, p x) = ~forall x, ~p x := 
47   (lambda h ha => exists.elim h (lambda x hpx => ha x hpx),
48    lambda h => classical.byContradiction (lambda hnex => h (lambda x hpx => hnex (x, hpx))))
49
50 end var
51
52
53
54 -- A simple inductive definition with constant constructors
55 inductive Weekday where
56   | sunday
57   | monday
58   | tuesday
59   | wednesday
60   | thursday
61   | friday
62   | saturday
63   deriving Repr -- Technicality (represent elements of Weekday by their respective names)
64
65 open Weekday -- Open namespace Weekday (Weekday.monday -> monday)
66
67 -- A simple functions on Weekday
68 def next_day (d : Weekday) : Weekday := by
69   cases d with
70     | sunday => exact monday
71     | monday => exact tuesday
72     | tuesday => exact wednesday
73     | wednesday => exact thursday
74     | thursday => exact friday
75     | friday => exact saturday
76     | saturday => exact sunday
77
78 -- Another simple functions on Weekday
79 def prev_day (d : Weekday) : Weekday := by
80   cases d with
81     | sunday => exact saturday
82     | monday => exact sunday
83     | tuesday => exact monday
84     | wednesday => exact tuesday
85     | thursday => exact wednesday
86     | friday => exact thursday
87     | saturday => exact friday
88
89 #check sunday
90 #check next_day sunday
```

Lean Infoview

- Talklean:130:22
- Expected type
- Weekday
- Messages (1)
- Talklean:130:0
- next_day sunday : Weekday

All Messages (6)

Restart File

Zeile 130, Spalte 23 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:22

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

REPL GEB > Talklean > ...

GRUPPE 1 • Talklean GEB U

GRUPPE 2 Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

EXPLORER

Talklean U ●

```
43 section var
44
45 -- Same proof in term mode
46 lemma ex_iff : (exists x, p x) = ~forall x, ~p x := 
47   (lambda h ha => exists.elim h (lambda x hpx => ha x hpx),
48    lambda h => classical.byContradiction (lambda hnex => h (lambda x hpx => hnex (x, hpx))))
49
50 end var
51
52
53
54 -- A simple inductive definition with constant constructors
55 inductive Weekday where
56   | sunday
57   | monday
58   | tuesday
59   | wednesday
60   | thursday
61   | friday
62   | saturday
63   deriving Repr -- Technicality (represent elements of Weekday by their respective names)
64
65 open Weekday -- Open namespace Weekday (Weekday.monday -> monday)
66
67 -- A simple functions on Weekday
68 def next_day (d : Weekday) : Weekday := by
69   cases d with
70     | sunday => exact monday
71     | monday => exact tuesday
72     | tuesday => exact wednesday
73     | wednesday => exact thursday
74     | thursday => exact friday
75     | friday => exact saturday
76     | saturday => exact sunday
77
78 -- Another simple functions on Weekday
79 def prev_day (d : Weekday) : Weekday := by
80   cases d with
81     | sunday => exact saturday
82     | monday => exact sunday
83     | tuesday => exact monday
84     | wednesday => exact tuesday
85     | thursday => exact wednesday
86     | friday => exact thursday
87     | saturday => exact friday
88
89 #check sunday
90 #check next_day sunday
91 #eval prev_day sunday|
```

Lean Infoview

- ▼ Talklean:131:21
- ▼ Expected type
- ↳ Weekday
- ▼ Messages (1)
- ▼ Talklean:131:0
- Weekday.saturday

All Messages (7)

Restart File

Zeile 131, Spalte 22 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:22

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

RESTART FILE

EXPLORER

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E TalkLean GEB 2, U

GRUPPE 2

- E Lean Infoview

> GEB

> GLÜEDERUNG

> ZEITACHSE

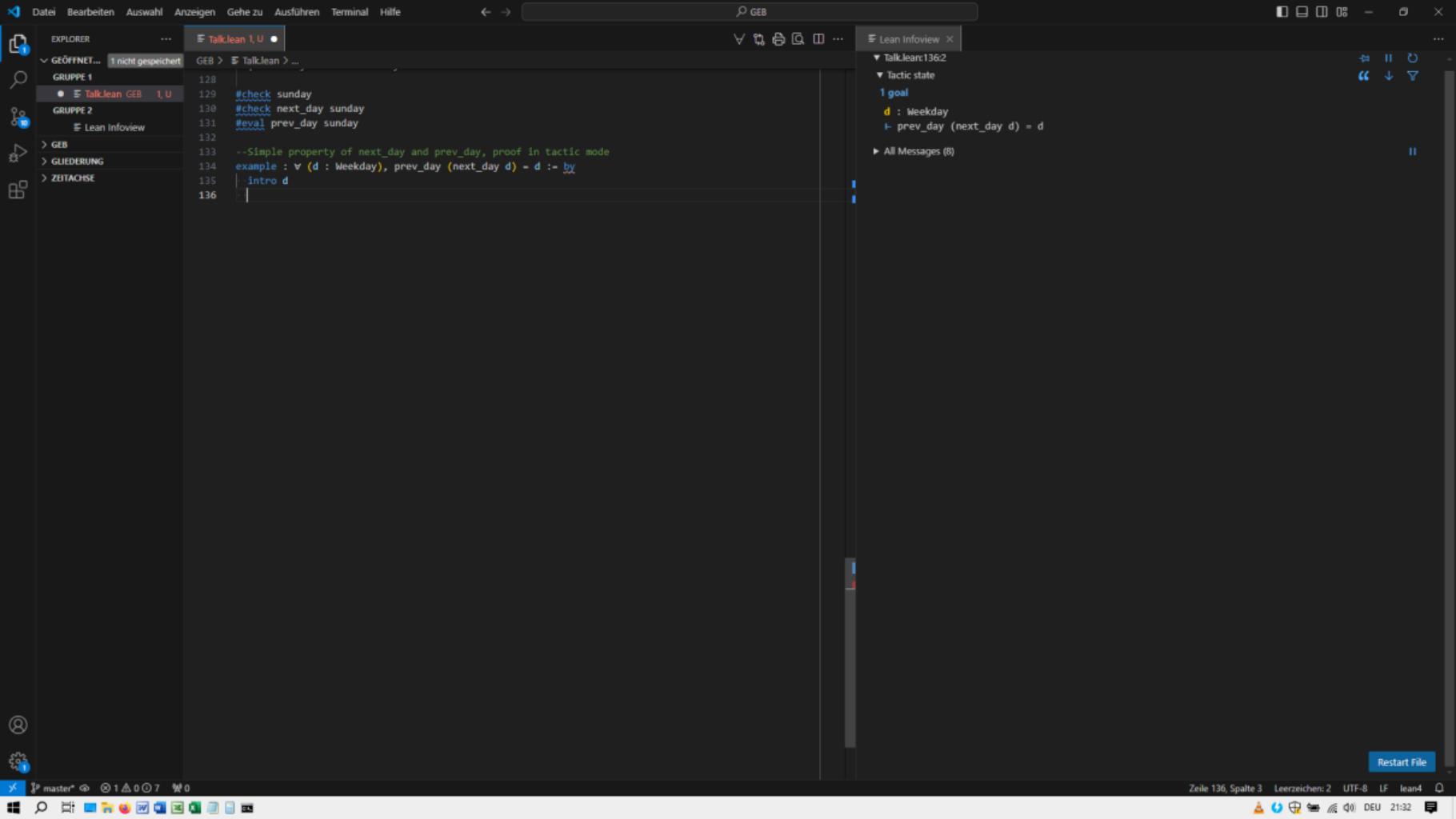
TalkLean 2, U

```
128
129 #check sunday
130 #check next_day sunday
131 #eval prev_day sunday
132
133 --Simple property of next_day and prev_day, proof in tactic mode
134 example : ∀ (d : Weekday), prev_day (next_day d) = d := by
135
```

Lean Infoview

- ▼ TalkLean:135:2
- ▼ Tactic state
- 1 goal
 - ↳ ∀ (d : Weekday), prev_day (next_day d) = d
- ▼ Messages (1)
 - ▼ TalkLean:135:2
 - unexpected end of input; expected '{'
- All Messages (9)

Zeil 135, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:32



Datei Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

Explorer GLÖFFNET... GRUPPE 1 GRUPPE 2 > GEB > GLIEDERUNG > ZEITACHSE

Talklean 1, U [nicht gespeichert]

GEB > Talklean > example

```
128
129 #check sunday
130 #check next_day sunday
131 #eval prev_day sunday
132
133 --Simple property of next_day and prev_day, proof in tactic mode
134 example : ∀ (d : Weekday), prev_day (next_day d) = d := by
135 intro d
136 cases d with
137 | sunday => sorry
138 | monday => sorry
139 | tuesday => sorry
140 | wednesday => sorry
141 | thursday => sorry
142 | friday => sorry
143 | saturday => sorry|
```

Lean Infoview

Talklean:143:21
Tactic state
No goals
All Messages (8)



Restart File

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

RESTART FILE

EXPLORER

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E TalkLean GEB 1, U

GRUPPE 2

- E Lean Infoview

> GEB

> GLÜDERUNG

> ZEITACHSE

TalkLean 1, U

GEB > Talklean > example

```
128
129 #check sunday
130 #check next_day sunday
131 #eval prev_day sunday
132
133 --Simple property of next_day and prev_day, proof in tactic mode
134 example : ∀ (d : Weekday), prev_day (next_day d) = d := by
135 intro d
136 cases d with
137 | sunday => sorry
138 | monday => sorry
139 | tuesday => sorry
140 | wednesday => sorry
141 | thursday => sorry
142 | friday => sorry
143 | saturday => sorry
```

Lean Infoview

TalkLean:137:14

Tactic state

1 goal

case sunday

prev_day (next_day sunday) = sunday

All Messages (0)

Zeile 137, Spalte 15 Leerzeichen:2 UTF-8 LF lean4 DEU 21:32

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

RESTART FILE

EXPLORER

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E TalkLean GEB 1, U

GRUPPE 2

- E Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

TalkLean 1, U

GEB > TalkLean > example

```
128
129 #check sunday
130 #check next_day sunday
131 #eval prev_day sunday
132
133 --Simple property of next_day and prev_day, proof in tactic mode
134 example : ∀ (d : Weekday), prev_day (next_day d) = d := by
135 intro d
136 cases d with
137 | sunday => rfl
138 | monday => sorry
139 | tuesday => sorry
140 | wednesday => sorry
141 | thursday => sorry
142 | friday => sorry
143 | saturday => sorry
```

Lean Infoview

TalkLean:137:17

Tactic state

No goals

All Messages (8)

Restart File

Zeile 137, Spalte 18 Leerzeichen: 2 UTF-8 LF lean4

Windows Taskbar

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

RESTART FILE

EXPLORER

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E TalkLean GEB 1, U

GRUPPE 2

- E Lean Infoview

> GEB

> GLÜDERUNG

> ZEITACHSE

TalkLean 1, U

GEB > TalkLean > example

```
128
129 #check sunday
130 #check next_day sunday
131 #eval prev_day sunday
132
133 --Simple property of next_day and prev_day, proof in tactic mode
134 example : ∀ (d : Weekday), prev_day (next_day d) = d := by
135 intro d
136 cases d with
137 | sunday => rfl
138 | monday => sorry
139 | tuesday => sorry
140 | wednesday => sorry
141 | thursday => sorry
142 | friday => sorry
143 | saturday => sorry
```

Lean Infoview

TalkLean:138:14

Tactic state

1 goal

case monday

prev_day (next_day monday) = monday

All Messages (0)

Zeile 138, Spalte 15 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:32

The screenshot shows a development environment for the Lean 4 theorem prover within the Visual Studio Code (VS Code) interface. The top navigation bar includes 'Datei', 'Bearbeiten', 'Auswahl', 'Anzeigen', 'Gehe zu', 'Ausführen', 'Terminal', and 'Hilfe'. The title bar displays 'GEB' and the file path 'Talklean 1, U'. The left sidebar contains icons for 'EXPLORER', 'GÖFFNET...', 'GRUPPE 1', 'GRUPPE 2', 'Lean Infoview', 'GEB', 'GUEDERUNG', and 'ZETACHSE'. The main editor area shows a Lean 4 code file named 'Talklean 1, U' with the status 'nicht gespeichert'. The code includes imports from 'GEB' and 'Tactic', and defines a tactic mode example involving 'next_day' and 'prev_day' for weekdays. The right-hand panel, titled 'Lean Infoview', displays the state of the proof attempt: 'Talklean:138:17', 'Tactic state', 'No goals', and 'All Messages (8)'. A status bar at the bottom shows 'Restart File' and system information like 'Zeile 138, Spalte 18', 'Leerzeichen:2', 'UTF-8', 'LF', 'lean4', and the date/time '21:33'.

```
128
129 #check sunday
130 #check next_day sunday
131 #eval prev_day sunday
132
133 --Simple property of next_day and prev_day, proof in tactic mode
134 example : ∀ (d : Weekday), prev_day (next_day d) = d := by
135 intro d
136 cases d with
137 | sunday => rfl
138 | monday => rfl|
139 | tuesday => sorry
140 | wednesday => sorry
141 | thursday => sorry
142 | friday => sorry
143 | saturday => sorry
```

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

128
129 `#check sunday`
130 `#check next_day sunday`
131 `#eval prev_day sunday`
132
133 `--Simple property of next_day and prev_day, proof in tactic mode`
example : `v (d : Weekday), prev_day (next_day d) = d := by`
135 `intro d`
136 `cases d with`
137 `| sunday => rfl`
138 `| monday => rfl`
139 `| tuesday => rfl`
140 `| wednesday => rfl`
141 `| thursday => rfl`
142 `| friday => rfl`
143 `| saturday => rfl`
144 |

Lean Infoview

Talklean:144:2
Tactic state
No goals
All Messages (7)

Restart File

Zeile 144, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:33

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

Explorer ...

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E Talklean GEB 2, U

GRUPPE 2

- E Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

Talklean 2, U

```
128
129 #check sunday
130 #check next_day sunday
131 #eval prev_day sunday
132
133 --Simple property of next_day and prev_day, proof in tactic mode
134 example : ∀ (d : Weekday), prev_day (next_day d) = d := by
135 intro d
136 cases d with
137 | sunday => rfl
138 | monday => rfl
139 | tuesday => rfl
140 | wednesday => rfl
141 | thursday => rfl
142 | friday => rfl
143 | saturday => rfl
144
145 --Named version of previous example, proof in tactic mode
146 lemma prev_next_eq_id : ∀ (d : Weekday), prev_day (next_day d) = d := by
```

Lean Infoview

- ▼ Talklean:147:2
- ▼ Tactic state
- 1 goal
 - |- ∀ (d : Weekday), prev_day (next_day d) = d
- ▼ Messages (1)
 - ▼ Talklean:147:2
 - unexpected end of input; expected '{'
- All Messages (9)

Restart File

master* ④ ⑤ 2 ▲ 0 ⑦ ⑧ ⑨

Zeile 147, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:33

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

Explorer ...

GLÖFFNET... [nicht gespeichert] GEB > Talklean > ...

GRUPPE 1

- Talklean GEB 1, U

GRUPPE 2

- Lean Infoview

> GEB

> GLÜDERUNG

> ZEITACHSE

Talklean 1, U

```
128
129 #check sunday
130 #check next_day sunday
131 #eval prev_day sunday
132
133 --Simple property of next_day and prev_day, proof in tactic mode
134 example : ∀ (d : Weekday), prev_day (next_day d) = d := by
135 intro d
136 cases d with
137 | sunday => rfl
138 | monday => rfl
139 | tuesday => rfl
140 | wednesday => rfl
141 | thursday => rfl
142 | friday => rfl
143 | saturday => rfl
144
145 --Named version of previous example, proof in tactic mode
146 lemma prev_next_eq_id : ∀ (d : Weekday), prev_day (next_day d) = d := by
147 intro d
```

Lean Infoview

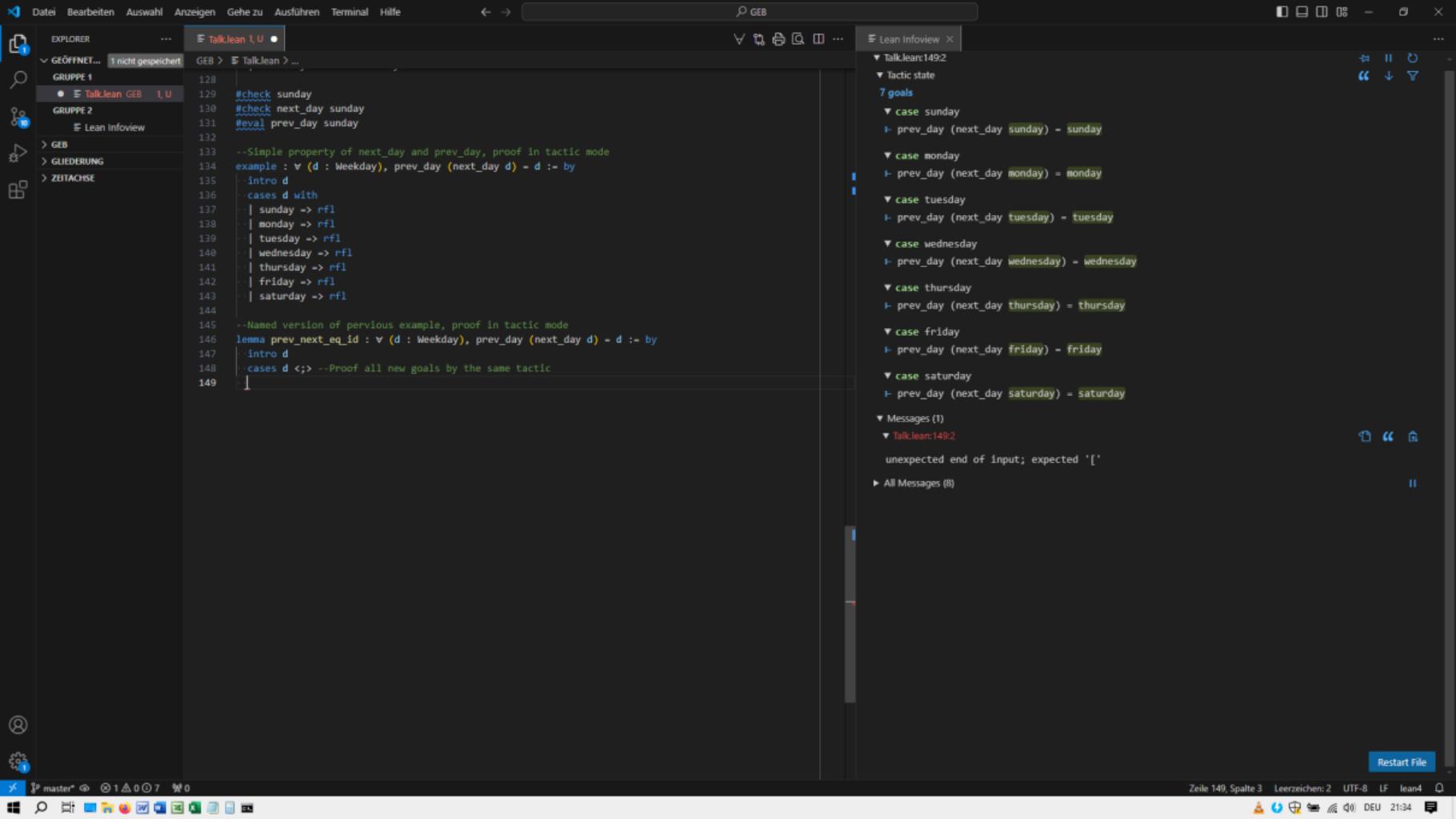
- ▼ Talklean:148:2
- ▼ Tactic state
- 1 goal
- d : Weekday
- |- prev_day (next_day d) = d

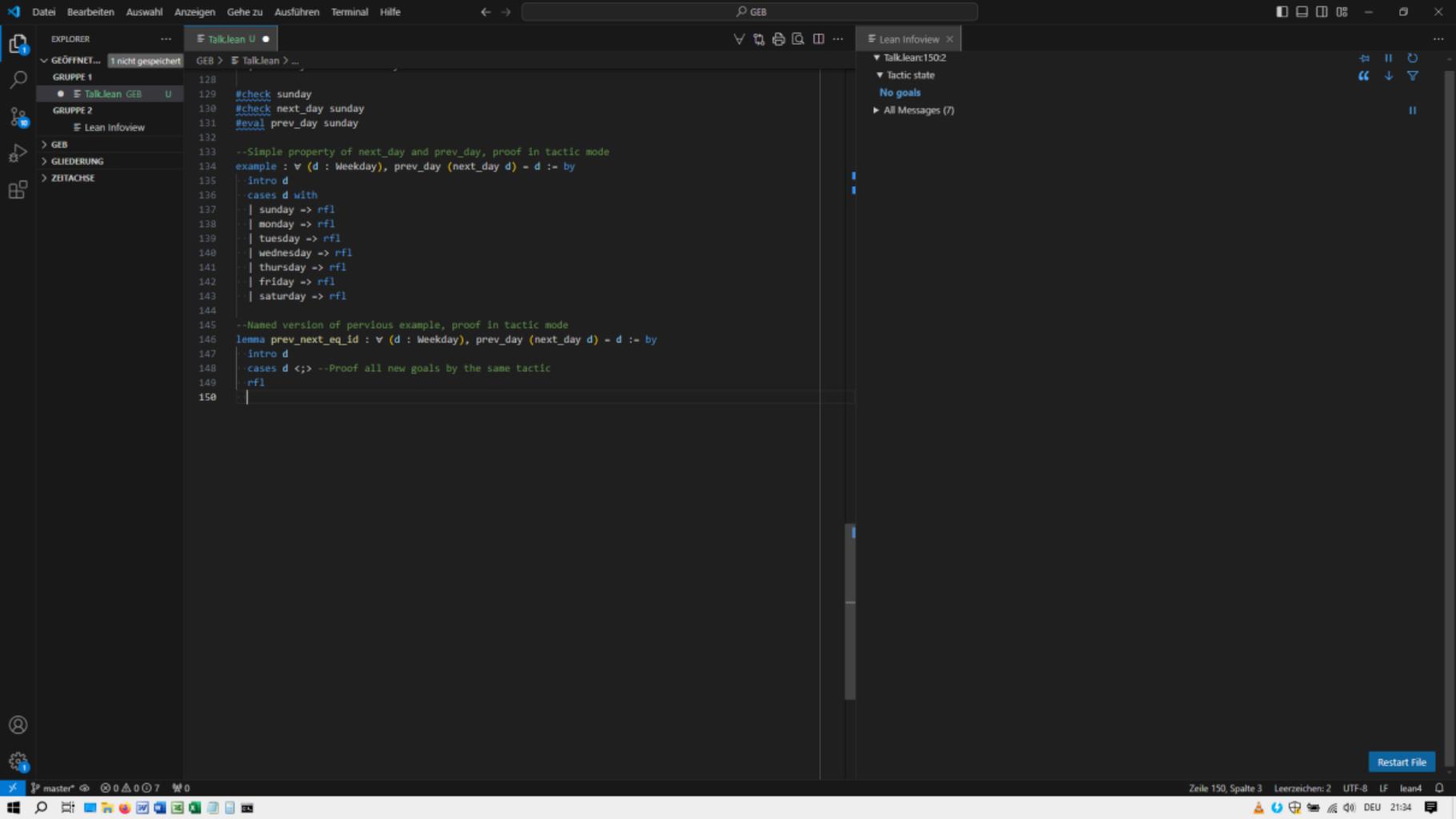
All Messages (8)

Restart File

Zeile 148, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4

Windows Taskbar





File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

Explorer ...

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E Talklean GEB 2, U

GRUPPE 2

- E Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

Talklean 2, U

```
128 | #check sunday
129 | #check next_day sunday
130 | #eval prev_day sunday
131 |
132 |
133 --Simple property of next_day and prev_day, proof in tactic mode
134 example : ∀ (d : Weekday), prev_day (next_day d) = d := by
135 intro d
136 cases d with
137 | sunday => rfl
138 | monday => rfl
139 | tuesday => rfl
140 | wednesday => rfl
141 | thursday => rfl
142 | friday => rfl
143 | saturday => rfl
144 |
145 --Named version of previous example, proof in tactic mode
146 lemma prev_next_eq_id : ∀ (d : Weekday), prev_day (next_day d) = d := by
147 intro d
148 cases d <:> --Proof all new goals by the same tactic
149 rfl
150
151 --Elements are equal to themselves by definition
152 example : sunday = sunday := by
153 rfl
```

Lean Infoview

- ▼ Talklean:153:2
- ▼ Tactic state
- 1 goal
 - ↳ sunday = sunday
- ▼ Messages (1)
 - ▼ Talklean:153:2
 - unexpected end of input; expected '{'

All Messages (9)

Restart File

master* ④ ⑤ 2 ▲ 0 ⑦ ⑧ ⑨ Zeile 153, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4

Windows Taskbar

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

128
129 ~~#check sunday~~
130 ~~#check next_day sunday~~
131 ~~#eval prev_day sunday~~
132
133 --Simple property of next_day and prev_day, proof in tactic mode
134 example : ∀ (d : Weekday), prev_day (next_day d) = d := by
135 intro d
136 cases d with
137 | sunday => rfl
138 | monday => rfl
139 | tuesday => rfl
140 | wednesday => rfl
141 | thursday => rfl
142 | friday => rfl
143 | saturday => rfl
144
145 --Named version of previous example, proof in tactic mode
146 lemma prev_next_eq_id : ∀ (d : Weekday), prev_day (next_day d) = d := by
147 intro d
148 cases d <:> --Proof all new goals by the same tactic
149 rfl
150
151 --Elements are equal to themselves by definition
152 example : sunday = sunday := by
153 rfl
154

Lean Infoview

Talklean:154:2
Tactic state
No goals
All Messages (7)

Restart File

Zeile 154, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:35

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

RESTART FILE

EXPLORER

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E Talklean GEB 2, U

GRUPPE 2

- E Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

Talklean 2, U

```
128 #check sunday
129 #check next_day sunday
130 #eval prev_day sunday
132
133 --Simple property of next_day and prev_day, proof in tactic mode
134 example : ∀ (d : Weekday), prev_day (next_day d) = d := by
135 intro d
136 cases d with
137 | sunday => rfl
138 | monday => rfl
139 | tuesday => rfl
140 | wednesday => rfl
141 | thursday => rfl
142 | friday => rfl
143 | saturday => rfl
144
145 --Named version of previous example, proof in tactic mode
146 lemma prev_next_eq_id : ∀ (d : Weekday), prev_day (next_day d) = d := by
147 intro d
148 cases d <;> --Proof all new goals by the same tactic
149 rfl
150
151 --Elements are equal to themselves by definition
152 example : sunday = sunday := by
153 rfl
154
155 --Elements created by different constructors are not equal by definition
156 example : sunday ≠ monday := by
```

Lean Infoview

- ▼ Talklean:157:2
- ▼ Tactic state
- 1 goal
 - ↳ sunday ≠ monday
- ▼ Messages (1)
 - ▼ Talklean:157:2
 - unexpected end of input; expected '{'
- ▶ All Messages (9)

Zeile 157, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:36

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

GRUPPE 1

- Talklean GEB U

GRUPPE 2

- Lean Infoview

> GEB

> GUEDERUNG

> ZETACHSE

EXPLORER

Talklean U

128
129 `#check sunday`
130 `#check next_day sunday`
131 `#eval prev_day sunday`
132
133 --Simple property of next_day and prev_day, proof in tactic mode
134 example : ∀ (d : Weekday), prev_day (next_day d) = d := by
135 intro d
136 cases d with
137 | sunday => rfl
138 | monday => rfl
139 | tuesday => rfl
140 | wednesday => rfl
141 | thursday => rfl
142 | friday => rfl
143 | saturday => rfl
144
145 --Named version of previous example, proof in tactic mode
146 lemma prev_next_eq_id : ∀ (d : Weekday), prev_day (next_day d) = d := by
147 intro d
148 cases d <>; --Proof all new goals by the same tactic
149 rfl
150
151 --Elements are equal to themselves by definition
152 example : sunday = sunday := by
153 rfl
154
155 --Elements created by different constructors are not equal by definition
156 example : sunday ≠ monday := by
157 exact Weekday.noConfusion --noConfusion is created with every inductive type
158

Lean Infoview

Talklean:158:2

Tactic state

No goals

All Messages (7)

Restart File

Zeile 158, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:36

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

GRUPPE 1

- E Talklean GEB U

GRUPPE 2

- E Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

REPL

EXPLORER

Talklean U

155 --Elements creates by different constructors are not equal by definition
156 example : sunday ≠ monday := by
157 exact Weekday.noConfusion --noConfusion is created with every inductive type
158
159
160
161 --A more more sophisticated inductive definition involving constructors with arguments
162 --Natural numbers are defined in exactly this way in mathlib
163 inductive MyNat where
164 | zero : MyNat
165 | succ : MyNat → MyNat
166 deriving Repr
167
168 open MyNat
169

Lean Infoview

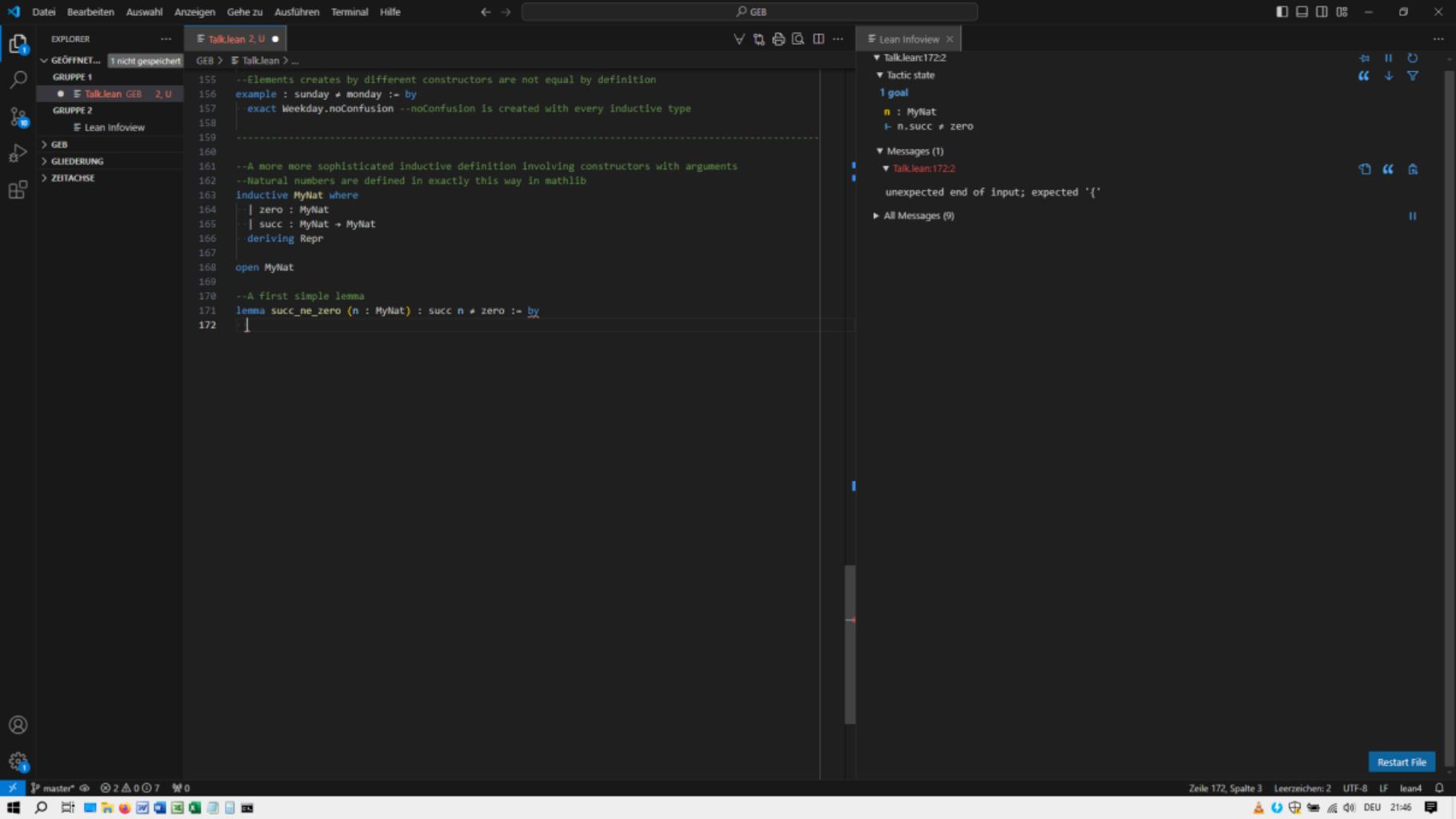
Talklean:169:0

No info found.

All Messages (7)

Restart File

Zeile 169, Spalte 1 Leerzeichen:2 UTF-8 LF lean4 DEU 21:46



File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

GlÖFFNET... 1 nicht gespeichert

GRUPPE 1 • Talklean GEB U

GRUPPE 2 🔍 Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

Talklean U

GEB > Talklean > ...

```
155 --Elements creates by different constructors are not equal by definition
156 example : sunday ≠ monday := by
157 | exact Weekday.noConfusion --noConfusion is created with every inductive type
158
159 -----
160
161 --A more more sophisticated inductive definition involving constructors with arguments
162 --Natural numbers are defined in exactly this way in mathlib
163 inductive MyNat where
164 | zero : MyNat
165 | succ : MyNat → MyNat
166 | deriving Repr
167
168 open MyNat
169
170 --A first simple lemma
171 lemma succ_ne_zero (n : MyNat) : succ n ≠ zero := by
172 | exact MyNat.noConfusion
173 |
```

Lean Infoview

Talklean:173-2

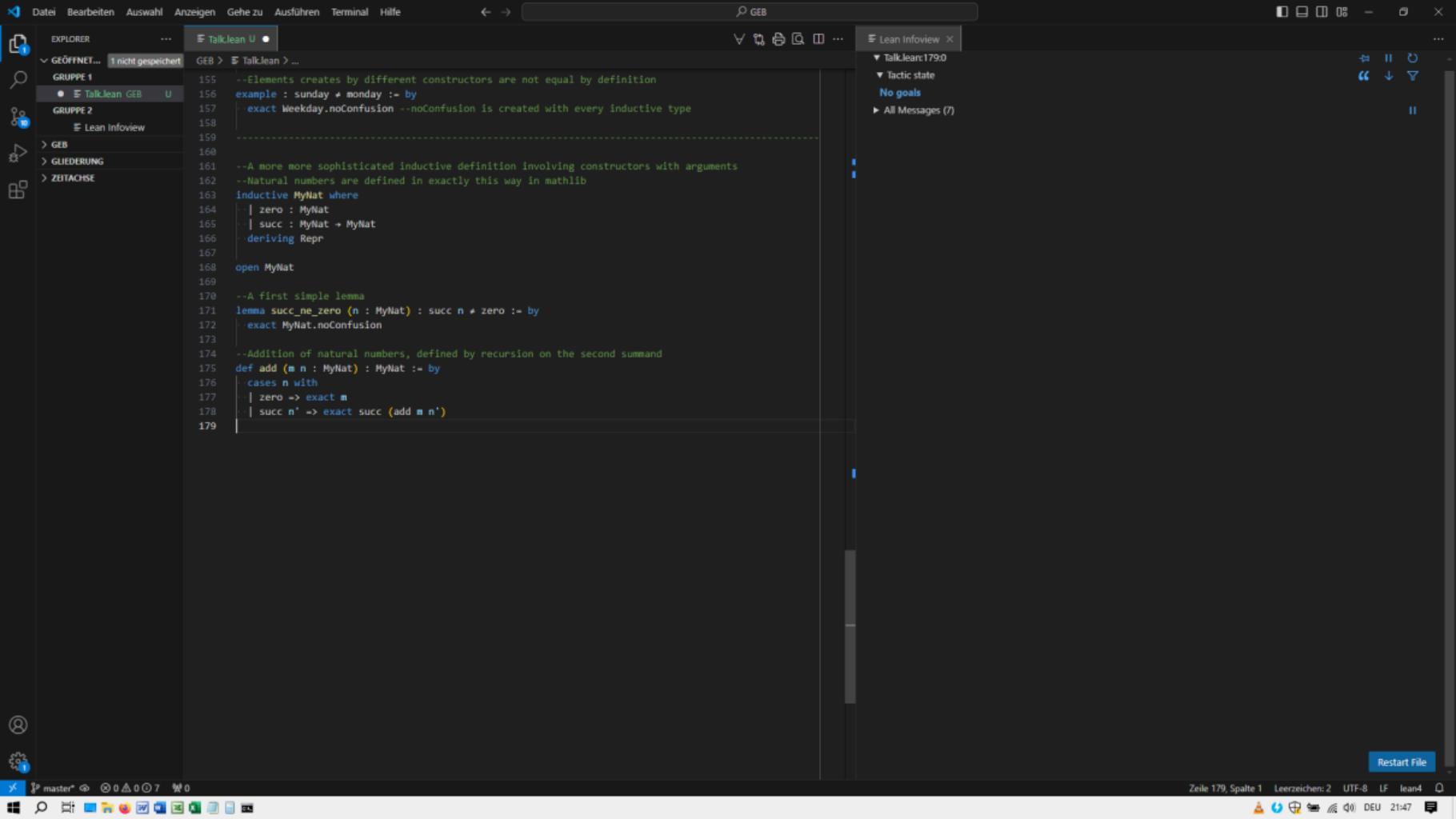
Tactic state

No goals

All Messages (7)

Restart File

Zeile 173, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:46



File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

REPL GEB > Talklean > ...

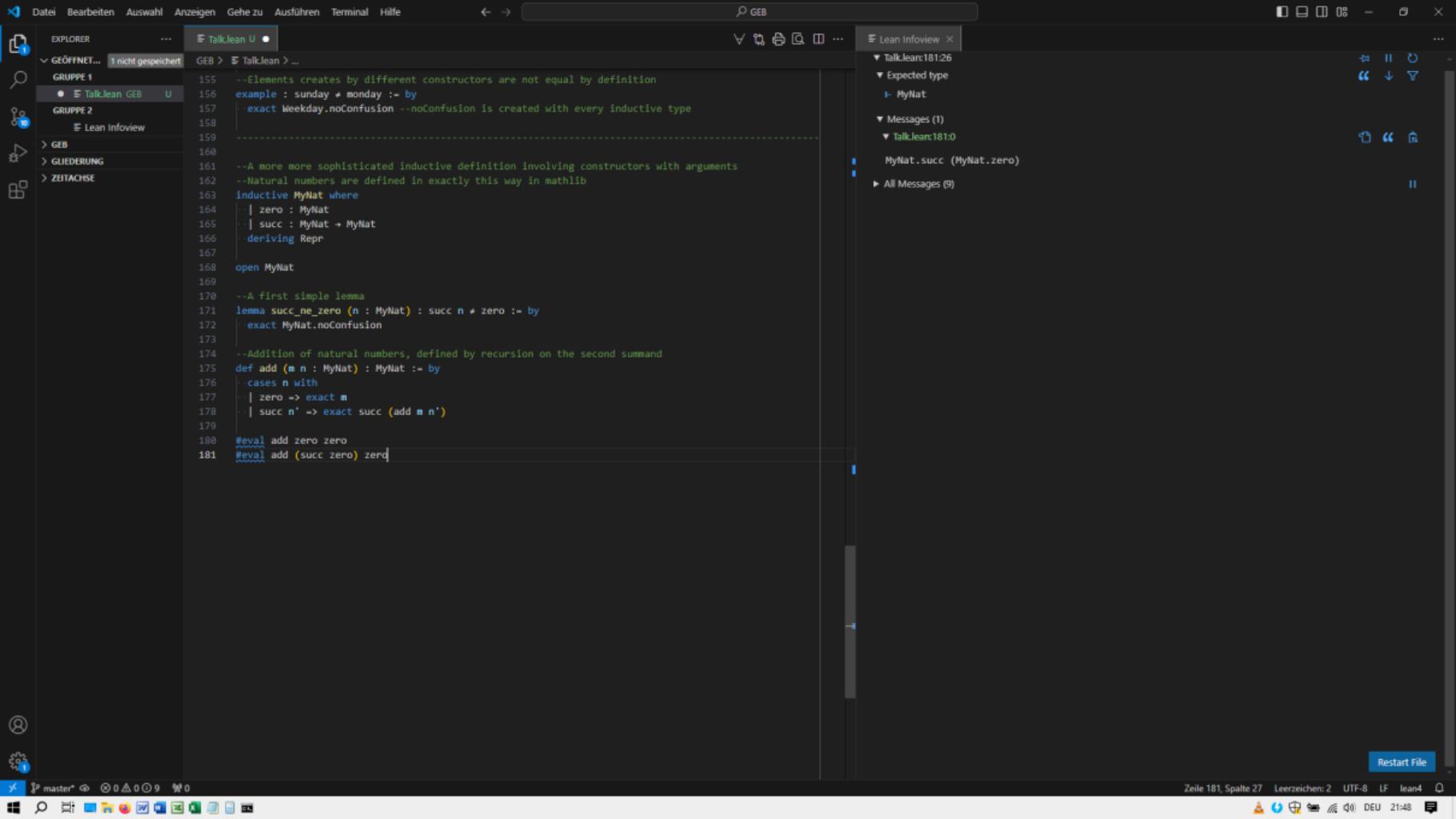
155 --Elements created by different constructors are not equal by definition
156 example : sunday ≠ monday := by
157 · exact Weekday.noConfusion --noConfusion is created with every inductive type
158
159
160
161 --A more more sophisticated inductive definition involving constructors with arguments
162 --Natural numbers are defined in exactly this way in mathlib
163 inductive MyNat where
164 ·| zero : MyNat
165 ·| succ : MyNat → MyNat
166 ·| deriving Repr
167
168 open MyNat
169
170 --A first simple lemma
171 lemma succ_ne_zero (n : MyNat) : succ n ≠ zero := by
172 · exact MyNat.noConfusion
173
174 --Addition of natural numbers, defined by recursion on the second summand
175 def add (m n : MyNat) : MyNat := by
176 · cases n with
177 · | zero => exact m
178 · | succ n' => exact succ (add m n')
179
180 #eval add zero zero

Lean Infoview

- ▼ Talklean:180:19
- ▼ Expected type
- ↳ MyNat
- ▼ Messages (1)
- ▼ Talklean:180:0
- MyNat.zero
- ▶ All Messages (8)

Restart File

Zeile 180, Spalte 20 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:48



File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

155 --Elements creates by different constructors are not equal by definition
156 example : sunday + monday := by
157 · exact Weekday.noConfusion --noConfusion is created with every inductive type
158
159
160
161 --A more more sophisticated inductive definition involving constructors with arguments
162 --Natural numbers are defined in exactly this way in mathlib
163 inductive MyNat where
164 ·| zero : MyNat
165 ·| succ : MyNat → MyNat
166 ·| deriving Repr
167
168 open MyNat
169
170 --A first simple lemma
171 lemma succ_ne_zero (n : MyNat) : succ n ≠ zero := by
172 · exact MyNat.noConfusion
173
174 --Addition of natural numbers, defined by recursion on the second summand
175 def add (m n : MyNat) : MyNat := by
176 · cases n with
177 · | zero => exact m
178 · | succ n' => exact succ (add m n')
179
180 #eval add zero zero
181 #eval add (succ zero) zero
182 #eval add (succ zero) [succ zero]

Lean Infoview

Talklean:182:33
Expected type
↳ MyNat
Messages (1)
Talklean:182:0
MyNat.succ (MyNat.succ (MyNat.zero))
All Messages (10)

Restart File

Zeile 182, Spalte 34 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:48

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

REPL master* 0 0 0 0 11 ⌂ 0

EXPLORER ...

GLÖFFNET... **Talklean U** [nicht gespeichert]

GRUPPE 1 • **Talklean GEB** U

GRUPPE 2

Lean Infoview

GEB

ÜBERSICHT

ZEITACHSE

Talklean U

155 --Elements created by different constructors are not equal by definition
156 example : sunday + monday := by
157 · exact Weekday.noConfusion --noConfusion is created with every inductive type
158
159
160
161 --A more more sophisticated inductive definition involving constructors with arguments
162 --Natural numbers are defined in exactly this way in mathlib
163 inductive MyNat where
164 ·| zero : MyNat
165 ·| succ : MyNat → MyNat
166 ·| deriving Repr
167
168 open MyNat
169
170 --A first simple lemma
171 lemma succ_ne_zero (n : MyNat) : succ n ≠ zero := by
172 · exact MyNat.noConfusion
173
174 --Addition of natural numbers, defined by recursion on the second summand
175 def add (m n : MyNat) : MyNat := by
176 · cases n with
177 · | zero => exact m
178 · | succ n' => exact succ (add m n')
179
180 #eval add zero zero
181 #eval add (succ zero) zero
182 #eval add (succ zero) (succ zero)
183 #eval succ [add (succ zero) zero]

Lean Infoview

Talklean:183:33

Expected type

↳ MyNat

Messages (1)

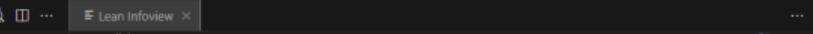
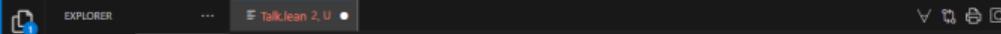
Talklean:183:0

MyNat.succ (MyNat.succ (MyNat.zero))

All Messages (11)

Restart File

Zeile 183, Spalte 34 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:48

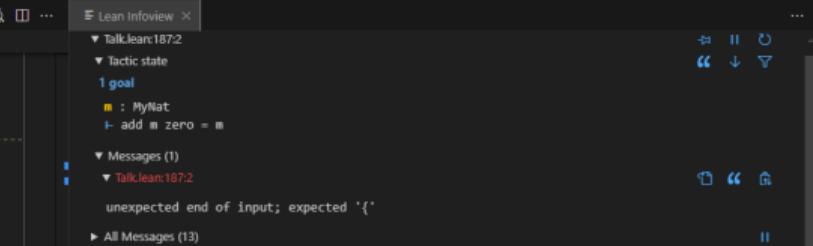
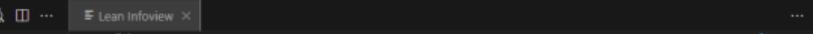


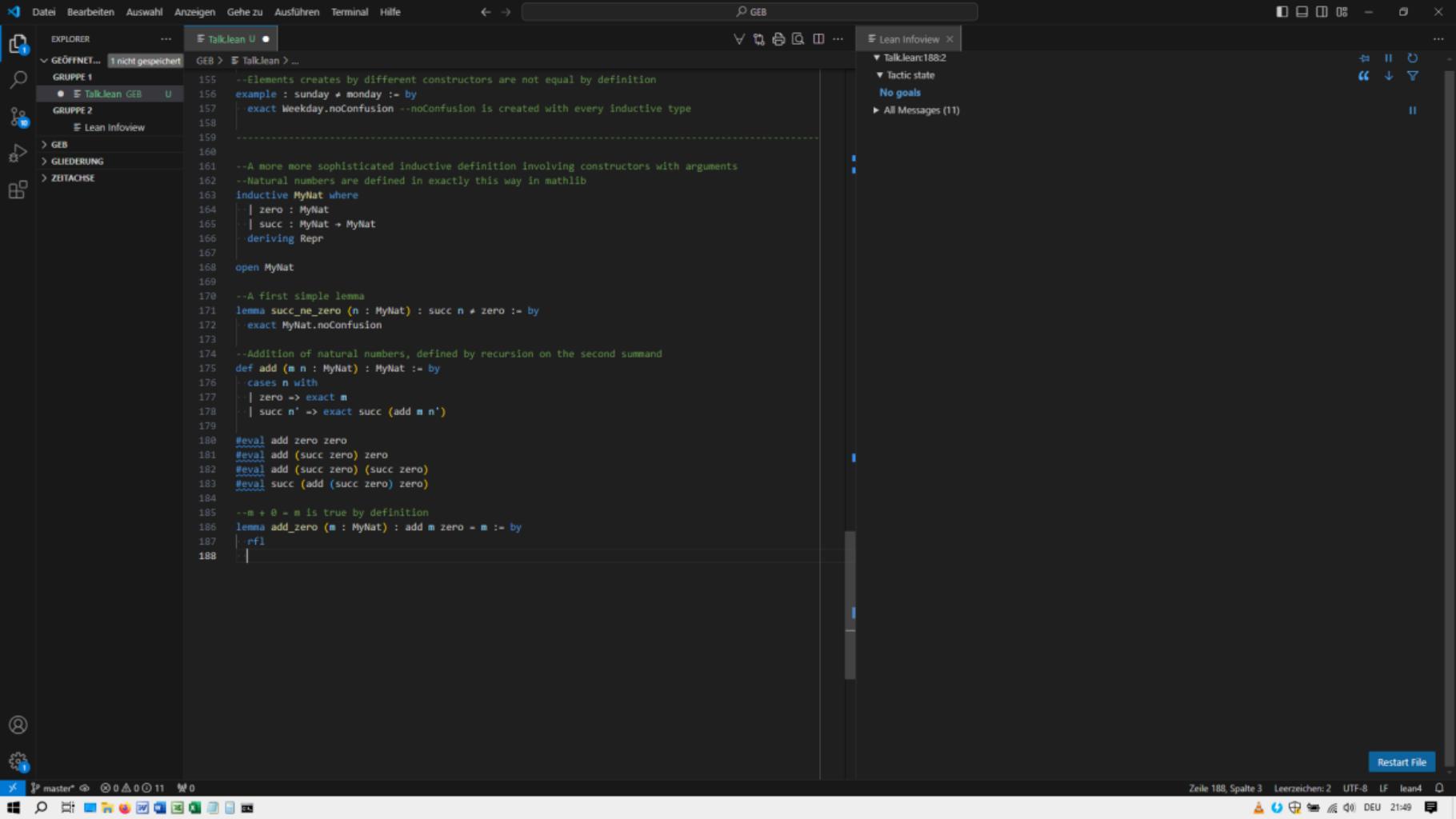
EXPLORER ...

GLÖFFNET... **Talklean 2, U** [nicht gespeichert]

GEB > Talklean > ...

```
155 --Elements creates by different constructors are not equal by definition
156 example : sunday + monday := by
157   exact Weekday.noConfusion --noConfusion is created with every inductive type
158
159
160
161 --A more more sophisticated inductive definition involving constructors with arguments
162 --Natural numbers are defined in exactly this way in mathlib
163 inductive MyNat where
164   | zero : MyNat
165   | succ : MyNat → MyNat
166   deriving Repr
167
168 open MyNat
169
170 --A first simple lemma
171 lemma succ_ne_zero (n : MyNat) : succ n ≠ zero := by
172   exact MyNat.noConfusion
173
174 --Addition of natural numbers, defined by recursion on the second summand
175 def add (m n : MyNat) : MyNat := by
176   cases n with
177   | zero => exact m
178   | succ n' => exact succ (add m n')
179
180 #eval add zero zero
181 #eval add (succ zero) zero
182 #eval add (succ zero) (succ zero)
183 #eval succ (add (succ zero) zero)
184
185 --m + 0 = m is true by definition
186 lemma add_zero (m : MyNat) : add m zero = m := by
187   
```





File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

EXPLORER ...

GLÖFFNET... **Talklean 2, U** [nicht gespeichert]

GRUPPE 1 • **TalkLean** GEB 2, U

GRUPPE 2

Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

Talklean 2, U

GEB > Talklean > ...

```
155 --Elements creates by different constructors are not equal by definition
156 example : sunday ≠ monday := by
157   exact Weekday.noConfusion --noConfusion is created with every inductive type
158
159
160
161 --A more more sophisticated inductive definition involving constructors with arguments
162 --Natural numbers are defined in exactly this way in mathlib
163 inductive MyNat where
164   | zero : MyNat
165   | succ : MyNat → MyNat
166   deriving Repr
167
168 open MyNat
169
170 --A first simple lemma
171 lemma succ_ne_zero (n : MyNat) : succ n ≠ zero := by
172   exact MyNat.noConfusion
173
174 --Addition of natural numbers, defined by recursion on the second summand
175 def add (m n : MyNat) : MyNat := by
176   cases n with
177   | zero => exact m
178   | succ n' => exact succ (add m n')
179
180 #eval add zero zero
181 #eval add (succ zero) zero
182 #eval add (succ zero) (succ zero)
183 #eval succ (add (succ zero) zero)
184
185 --m + n = m is true by definition
186 lemma add_zero (m : MyNat) : add m zero = m := by
187   rfl
188
189 --0 + m = m needs to be proven by induction
190 lemma zero_add (n : MyNat) : add zero n = n := by
191   
```

Lean Infoview

Talklean:191:2

Tactic state

1 goal

n : MyNat

|- add zero n = n

Messages (1)

Talklean:191:2

unexpected end of input; expected '{'

All Messages (13)

Restart File

Zeile 191, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:49

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

EXPLORER ...

GLÖFFNET... **TalkLean 1, U** [nicht gespeichert]

GRUPPE 1

- **TalkLean** GEB 1, U

GRUPPE 2

- Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

TalkLean 1, U

```
155 --Elements created by different constructors are not equal by definition
156 example : sunday ≠ monday := by
157   exact Weekday.noConfusion --noConfusion is created with every inductive type
158
159 -----
160
161 --A more more sophisticated inductive definition involving constructors with arguments
162 --Natural numbers are defined in exactly this way in mathlib
163 inductive MyNat where
164   | zero : MyNat
165   | succ : MyNat → MyNat
166   deriving Repr
167
168 open MyNat
169
170 --A first simple lemma
171 lemma succ_ne_zero (n : MyNat) : succ n ≠ zero := by
172   exact MyNat.noConfusion
173
174 --Addition of natural numbers, defined by recursion on the second summand
175 def add (m n : MyNat) : MyNat := by
176   cases n with
177   | zero => exact m
178   | succ n' => exact succ (add m n')
179
180 #eval add zero zero
181 #eval add (succ zero) zero
182 #eval add (succ zero) (succ zero)
183 #eval succ (add (succ zero) zero)
184
185 --m + 0 = m is true by definition
186 lemma add_zero (m : MyNat) : add m zero = m := by
187   rfl
188
189 --0 + m = m needs to be proven by induction
190 lemma zero_add (n : MyNat) : add zero n = n := by
191   induction n with
192   | zero =>
193     sorry
194   | succ n' ih =>
195     sorry
```

Lean Infoview

- **TalkLean:195:9**
- **Tactic state**
- **No goals**
- ▶ **All Messages (12)**

Restart File

Zeile 195, Spalte 10 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:49

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

REPL GEB > Talklean 1, U

EXPLORER GLÖFFNET... 1 nicht gespeichert GRUPPE 1 • E TalkLean GEB 1, U GRUPPE 2 E Lean Infoview GEB GUEDERUNG ZETACHSE

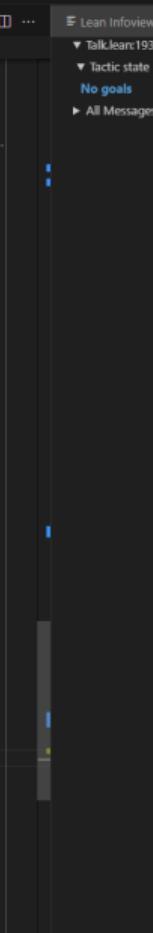
```
155 --Elements created by different constructors are not equal by definition
156 example : sunday ≠ monday := by
157   exact Weekday.noConfusion --noConfusion is created with every inductive type
158
159
160
161 --A more more sophisticated inductive definition involving constructors with arguments
162 --Natural numbers are defined in exactly this way in mathlib
163 inductive MyNat where
164   | zero : MyNat
165   | succ : MyNat → MyNat
166   deriving Repr
167
168 open MyNat
169
170 --A first simple lemma
171 lemma succ_ne_zero (n : MyNat) : succ n ≠ zero := by
172   exact MyNat.noConfusion
173
174 --Addition of natural numbers, defined by recursion on the second summand
175 def add (m n : MyNat) : MyNat := by
176   cases n with
177   | zero => exact m
178   | succ n' => exact succ (add m n')
179
180 #eval add zero zero
181 #eval add (succ zero) zero
182 #eval add (succ zero) (succ zero)
183 #eval succ (add (succ zero) zero)
184
185 --m + 0 = m is true by definition
186 lemma add_zero (m : MyNat) : add m zero = m := by
187   rfl
188
189 --0 + m = m needs to be proven by induction
190 lemma zero_add (n : MyNat) : add zero n = n := by
191   induction n with
192   | zero =>
193     sorry
194   | succ n' ih =>
195     sorry
```

Lean Infoview

TalkLean:193:4
Tactic state
1 goal
case zero
add zero zero = zero
All Messages (12)

Restart File

Zeile 193, Spalte 5 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:50



File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

EXPLORER ...

GLÖFFNET... [nicht gespeichert]

GRUPPE 1

- TalkLean GEB 1, U

GRUPPE 2

- Lean Infoview

> GEB

> GLÜEDERUNG

> ZEITACHSE

TalkLean 1, U

GEB > Talklean > ...

```
155 --Elements creates by different constructors are not equal by definition
156 example : sunday ≠ monday := by
157   exact Weekday.noConfusion --noConfusion is created with every inductive type
158
159
160
161 --A more more sophisticated inductive definition involving constructors with arguments
162 --Natural numbers are defined in exactly this way in mathlib
163 inductive MyNat where
164   | zero : MyNat
165   | succ : MyNat → MyNat
166   deriving Repr
167
168 open MyNat
169
170 --A first simple lemma
171 lemma succ_ne_zero (n : MyNat) : succ n ≠ zero := by
172   exact MyNat.noConfusion
173
174 --Addition of natural numbers, defined by recursion on the second summand
175 def add (m n : MyNat) : MyNat := by
176   cases n with
177   | zero => exact m
178   | succ n' => exact succ (add m n')
179
180 #eval add zero zero
181 #eval add (succ zero) zero
182 #eval add (succ zero) (succ zero)
183 #eval succ (add (succ zero) zero)
184
185 --m + n = m is true by definition
186 lemma add_zero (m : MyNat) : add m zero = m := by
187   rfl
188
189 --0 + m = m needs to be proven by induction
190 lemma zero_add (n : MyNat) : add zero n = n := by
191   induction n with
192   | zero =>
193     rfl
194   | succ n' ih =>
195     sorry
```

Lean Infoview

TalkLean:195:4

Tactic state

1 goal

case succ

n' : MyNat

ih : add zero n' = n'

|- add zero n'.succ = n'.succ

All Messages (12)

Restart File

Zeile 195, Spalte 5 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:52

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe 🔍 GEB

EXPLORER ...

GLÖFFNET... [nicht gespeichert]

GRUPPE 1

- TalkLean GEB 1, U

GRUPPE 2

- Lean Infoview

> GEB

> QUEDERUNG

> ZEITACHSE

Talklean 1, U

```
155 --Elements created by different constructors are not equal by definition
156 example : sunday ≠ monday := by
157   exact Weekday.noConfusion --noConfusion is created with every inductive type
158
159
160
161 --A more more sophisticated inductive definition involving constructors with arguments
162 --Natural numbers are defined in exactly this way in mathlib
163 inductive MyNat where
164   | zero : MyNat
165   | succ : MyNat → MyNat
166   deriving Repr
167
168 open MyNat
169
170 --A first simple lemma
171 lemma succ_ne_zero (n : MyNat) : succ n ≠ zero := by
172   exact MyNat.noConfusion
173
174 --Addition of natural numbers, defined by recursion on the second summand
175 def add (m n : MyNat) : MyNat := by
176   cases n with
177   | zero => exact m
178   | succ n' => exact succ (add m n')
179
180 #eval add zero zero
181 #eval add (succ zero) zero
182 #eval add (succ zero) (succ zero)
183 #eval succ (add (succ zero) zero)
184
185 --m + n = m is true by definition
186 lemma add_zero (m : MyNat) : add m zero = m := by
187   rfl
188
189 --0 + m = m needs to be proven by induction
190 lemma zero_add (n : MyNat) : add zero n = n := by
191   induction n with
192   | zero =>
193     rfl
194   | succ n' ih =>
195     rw [add] ⋯ rewrite add
196   ⋯
```

Lean Infoview

TalkLean:196:4

Tactic state

1 goal

case succ

n' : MyNat

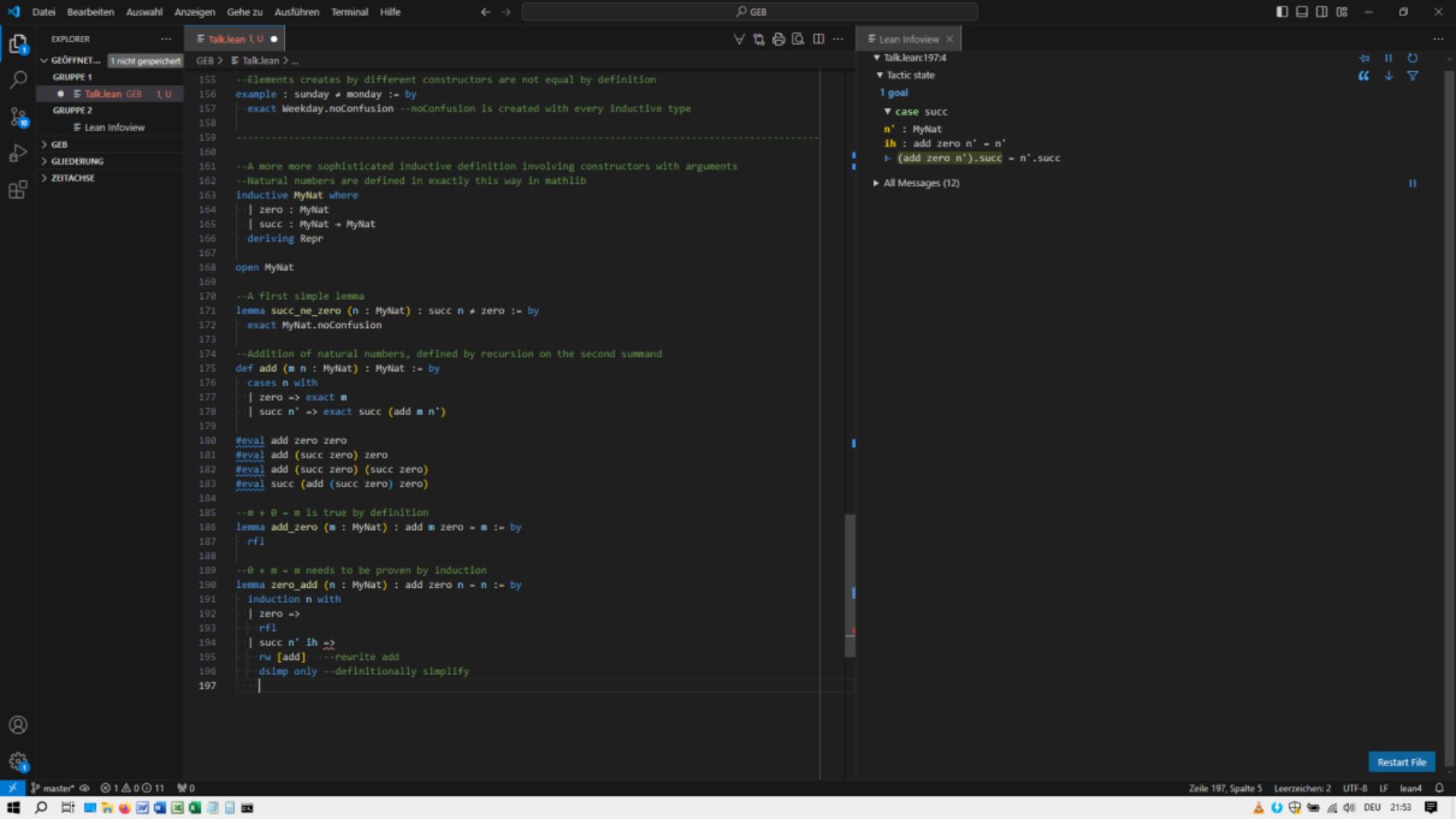
ih : add zero n' = n'

MyNat.casesOn (motive := fun t ↦ t + MyNat) n'.succ (fun h ↦ zero) (fun n'_1 h => (add zero n'_1).succ) ⋯ = n'.succ

All Messages (12)

Restart File

Zeile 196, Spalte 5 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:53



EXPLORER ...

GlÖFFNET... **Talklean U** ●

GEB > Talklean > ...

```
155 --Elements creates by different constructors are not equal by definition
156 example : sunday ≠ monday := by
157   exact Weekday.noConfusion --noConfusion is created with every inductive type
158
159 -----
160
161 --A more more sophisticated inductive definition involving constructors with arguments
162 --Natural numbers are defined in exactly this way in mathlib
163 inductive MyNat where
164   | zero : MyNat
165   | succ : MyNat → MyNat
166   deriving Repr
167
168 open MyNat
169
170 --A first simple lemma
171 lemma succ_ne_zero (n : MyNat) : succ n ≠ zero := by
172   exact MyNat.noConfusion
173
174 --Addition of natural numbers, defined by recursion on the second summand
175 def add (m n : MyNat) : MyNat := by
176   cases n with
177   | zero => exact m
178   | succ n' => exact succ (add m n')
179
180 #eval add zero zero
181 #eval add (succ zero) zero
182 #eval add (succ zero) (succ zero)
183 #eval succ (add (succ zero) zero)
184
185 --m + 0 = m is true by definition
186 lemma add_zero (m : MyNat) : add m zero = m := by
187   rfl
188
189 --0 + m = m needs to be proven by induction
190 lemma zero_add (n : MyNat) : add zero n = n := by
191   induction n with
192   | zero =>
193     rfl
194   | succ n' ih =>
195     rw [add] -----rewrite add
196     dsimp only -----definitionally simplify
197     rw [ih] -----rewrite using identity ih
198
```

Lean Infoview

▼ Talklean:198:2
▼ Tactic state
No goals
► All Messages (11)

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

LEADERBOARD

EXPLORER

GlÖFFNET... [nicht gespeichert]

GRUPPE 1

- Talklean GEB U

GRUPPE 2

- Lean Infoview

> GEB

> GUEDERUNG

> ZETACHSE

TALKLEAN

SEARCH

CODE

REPO

LOG

RESTART

RESTART FILE

Zeile 204, Spalte 1 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:56

188 --0 + m - m needs to be proven by induction
189 lemma zero_add (n : MyNat) : add zero n - n := by
190 induction n with
191 | zero =>
192 | rfl
193 | succ n' ih =>
194 rw [add] -----rewrite add
195 dsimp only -----definitionally simplify
196 rw [ih] -----rewrite using identity ih
197
198 --Multiplication of natural numbers, defined by recursion on the second factor
199 def mul (m n : MyNat) : MyNat := by
200 cases n with
201 | zero => exact zero
202 | succ n' => exact add (mul m n') s
203
204

Lean Infoview

Talklean:204:0

Tactic state

No goals

All Messages (11)

Restart File

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe GEB

Explorer ...

GlÖFFNET... nicht gespeichert

GRUPPE 1

- E Talklean GEB 2, U

GRUPPE 2

- E Lean Infoview

> GEB

> GUEDERUNG

> ZETACHSE

Talklean 2, U

```
188 --0 + m - m needs to be proven by induction
189 lemma zero_add (n : MyNat) : add zero n = n := by
190   induction n with
191   | zero =>
192     rfl
193   | succ n' ih =>
194     rw [add] -----rewrite add
195     dsimp only ---definitionally simplify
196     rw [ih] -----rewrite using identity ih
197
198 --Multiplication of natural numbers, defined by recursion on the second factor
199 def mul (m n : MyNat) : MyNat := by
200   cases n with
201   | zero => exact zero
202   | succ n' => exact add (mul m n') m
203
204 --m * 1 = m
205 lemma mul_one (m : MyNat) : mul m (succ zero) = m := by
206
```

Lean Infoview

- Talklean:207:2
- Tactic state
- 1 goal
 - m : MyNat
 - |- mul m zero.succ = m
- Messages (1)
 - Talklean:207:2
 - unexpected end of input; expected '{'
- All Messages (13)

Restart File

master ④ ⑤ 2 ▲ 0 ① 11 ⑥ 0 Zeile 207, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4

Windows Taskbar

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

Explorer ▾ GÖFFNET... [nicht gespeichert] GRUPPE 1 • Talklean GEB 1, U GRUPPE 2 Lean Infoview > GEB > GLIEDERUNG > ZEITACHSE

Talklean 1, U

```
188 --0 + m - m needs to be proven by induction
189 lemma zero_add (n : MyNat) : add zero n = n := by
190   induction n with
191   | zero =>
192     rfl
193   | succ n' ih =>
194     rw [add] -----rewrite add
195     dsimp only -----definitionally simplify
196     rw [ih] -----rewrite using identity ih
197
198 --Multiplication of natural numbers, defined by recursion on the second factor
199 def mul (m n : MyNat) : MyNat := by
200   cases n with
201   | zero => exact zero
202   | succ n' => exact add (mul m n') m
203
204 --m * 1 = m
205 lemma mul_one (m : MyNat) : mul m (succ zero) = m := by
206   rw [mul]
207   dsimp only
```

Lean Infoview

- ▼ Talklean:209-2
- ▼ Tactic state

1 goal

m : MyNat
|- add (mul m zero) m = m

All Messages (12)

Restart File

Zeile 209, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:57

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

Explorer ▾ GÖFFNET... [nicht gespeichert] GRUPPE 1 • Talklean GEB 1, U GRUPPE 2 Lean Infoview > GEB > GUEDERUNG > ZETACHSE

Talklean 1, U

```
188 --0 + m - m needs to be proven by induction
189 lemma zero_add (n : MyNat) : add zero n = n := by
190   induction n with
191   | zero =>
192     rfl
193   | succ n' ih =>
194     rw [add] -----rewrite add
195     dsimp only -----definitionally simplify
196     rw [ih] -----rewrite using identity ih
197
198 --Multiplication of natural numbers, defined by recursion on the second factor
199 def mul (m n : MyNat) : MyNat := by
200   cases n with
201   | zero => exact zero
202   | succ n' => exact add (mul m n') m
203
204 --m * 1 = m
205 lemma mul_one (m : MyNat) : mul m (succ zero) = m := by
206   rw [mul]
207   dsimp only
208   rw [mul]
209   dsimp only
210
211 |
```

Lean Infoview

Talklean:211:2
Tactic state
1 goal
m : MyNat
|- add zero m = m
All Messages (12)

Restart File

master* ① 1 ▲ 0 ① 11 ④ 0 Zeile 211, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4

Windows Taskbar: Search, Task View, Start, File Explorer, Edge, Microsoft Store, Mail, Photos, OneDrive, Settings, Control Panel, Task Manager, Help & Support, DEU, 21:57

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

Explorer ▾ GEFÖFFNET... [nicht gespeichert] GRUPPE 1 • Talklean GEB U GRUPPE 2 Lean Infoview > GEB > GUEDERUNG > ZEITACHSE

188 --0 + m - m needs to be proven by induction
189 lemma zero_add (n : MyNat) : add zero n = n := by
190 induction n with
191 | zero =>
192 | rfl
193 | succ n' ih =>
194 rw [add] -----rewrite add
195 dsimp only -----definitionally simplify
196 rw [ih] -----rewrite using identity ih
197
198 --Multiplication of natural numbers, defined by recursion on the second factor
199 def mul (m n : MyNat) : MyNat := by
200 cases n with
201 | zero => exact zero
202 | succ n' => exact add (mul m n') m
203
204 --m * 1 = m
205 lemma mul_one (m : MyNat) : mul m (succ zero) = m := by
206 rw [mul]
207 dsimp only
208 rw [mul]
209 dsimp only
210 rw [zero_add]
211 |

Lean Infoview ▾
Talklean:212-2
Tactic state
No goals
All Messages (11)

Restart File

Zeile 212, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:57

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB

EXPLORER

GlÖFFNET... [nicht gespeichert]

GRUPPE 1

- E Talklean GEB 2, U

GRUPPE 2

- E Lean Infoview

> GEB

> GUEDERUNG

> ZETACHSE

Talklean 2, U

```
188 --0 + m - m needs to be proven by induction
189 lemma zero_add (n : MyNat) : add zero n = n := by
190   induction n with
191   | zero =>
192     rfl
193   | succ n' ih =>
194     rw [add] .....rewrite add
195     dsimp only .....definitionally simplify
196     rw [ih] .....rewrite using identity ih
197
198 --Multiplication of natural numbers, defined by recursion on the second factor
199 def mul (m n : MyNat) : MyNat := by
200   cases n with
201   | zero => exact zero
202   | succ n' => exact add (mul m n') m
203
204 --m * 1 = m
205 lemma mul_one (m : MyNat) : mul m (succ zero) = m := by
206   rw [mul]
207   dsimp only
208   rw [mul]
209   dsimp only
210   rw [zero_add]
211
212 --1 * m = m
213 lemma one_mul (n : MyNat) : mul (succ zero) n = n := by
214   rw [one_mul]
```

Lean Infoview

- Talklean215:2
- Tactic state
- 1 goal
- n : MyNat
- |- mul zero.succ n = n
- Messages (1)
- Talklean215:2
- unexpected end of input; expected '{'

All Messages (13)

Restart File

Zeile 215, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:57

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

Explorer ▾ GÖFFNET... [nicht gespeichert] GRUPPE 1 • TalkLean GEB 1, U GRUPPE 2 Lean Infoview > GEB > GUEDERUNG > ZETACHSE

Talklean 1, U

```
188 --0 + m - m needs to be proven by induction
189 lemma zero_add (n : MyNat) : add zero n = n := by
190   induction n with
191   | zero =>
192     rfl
193   | succ n' ih =>
194     rw [add] -----rewrite add
195     dsimp only -----definitionally simplify
196     rw [ih] -----rewrite using identity ih
197
198 --Multiplication of natural numbers, defined by recursion on the second factor
199 def mul (m n : MyNat) : MyNat := by
200   cases n with
201   | zero => exact zero
202   | succ n' => exact add (mul m n') m
203
204 --m * 1 = m
205 lemma mul_one (m : MyNat) : mul m (succ zero) = m := by
206   rw [mul]
207   dsimp only
208   rw [mul]
209   dsimp only
210   rw [zero_add]
211
212 --1 * m = m
213 lemma one_mul (n : MyNat) : mul (succ zero) n = n := by
214   induction n with
215   | zero =>
216     sorry
217   | succ n' ih =>
218     sorry
219     sorry|
```

Lean Infoview

- ▼ TalkLean:219:9
- ▼ Tactic state
- No goals

All Messages (12)

Restart File

Zeile 219, Spalte 10 Leerzeichen: 2 UTF-8 LF lean4

0 0 1 0 11 90

Windows Taskbar

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

Explorer

GLÖFFNET... [nicht gespeichert]

GRUPPE 1

- E TalkLean GEB 1, U

GRUPPE 2

- E Lean Infoview

> GEB

> GLIEDERUNG

> ZEITACHSE

Talklean 1, U

```
188 --0 + m - m needs to be proven by induction
189 lemma zero_add (n : MyNat) : add zero n = n := by
190   induction n with
191   | zero =>
192     rfl
193   | succ n' ih =>
194     rw [add] -----rewrite add
195     dsimp only -----definitionally simplify
196     rw [ih] -----rewrite using identity ih
197
198 --Multiplication of natural numbers, defined by recursion on the second factor
199 def mul (m n : MyNat) : MyNat := by
200   cases n with
201   | zero => exact zero
202   | succ n' => exact add (mul m n') m
203
204 --m * 1 = m
205 lemma mul_one (m : MyNat) : mul m (succ zero) = m := by
206   rw [mul]
207   dsimp only
208   rw [mul]
209   dsimp only
210   rw [zero_add]
211
212 --1 * m = m
213 lemma one_mul (n : MyNat) : mul (succ zero) n = n := by
214   induction n with
215   | zero =>
216     sorry
217   | succ n' ih =>
218     sorry
219
```

Lean Infoview

TalkLean:217:4

Tactic state

1 goal

case zero

mul zero.succ zero = zero

All Messages (12)

Restart File

Zeile 217, Spalte 5 Leerzeichen: 2 UTF-8 LF lean4

0 0 1 0 11 90

Windows Taskbar

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

LEADERBOARD

EXPLORER

GRUPPE 1

GRUPPE 2

Lean Infoview

GEB

GUEDERUNG

ZETACHSE

TALKLEAN 1, U

188
189 --0 + m - m needs to be proven by induction
190 lemma zero_add (n : MyNat) : add zero n = n := by
191 induction n with
192 | zero =>
193 | rfl
194 | succ n' ih =>
195 rw [add] -----rewrite add
196 dsimp only -----definitionally simplify
197 rw [ih] -----rewrite using identity ih
198
199 --Multiplication of natural numbers, defined by recursion on the second factor
200 def mul (m n : MyNat) : MyNat := by
201 cases n with
202 | zero => exact zero
203 | succ n' => exact add (mul m n') m
204
205 --m * 1 = m
206 lemma mul_one (m : MyNat) : mul m (succ zero) = m := by
207 | rw [mul]
208 | dsimp only
209 | rw [mul]
210 | dsimp only
211 | rw [zero_add]
212
213 --1 * m = m
214 lemma one_mul (n : MyNat) : mul (succ zero) n = n := by
215 induction n with
216 | zero =>
217 | rfl|
218 | succ n' ih =>
219 | sorry

Lean Infoview

Talklean:217:
Tactic state
No goals
All Messages (12)

Restart File

Zeile 217, Spalte 8 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:58

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB

EXPLORER

GLÖFFNET... 1 nicht gespeichert

GRUPPE 1

- E Talklean GEB 2, U

GRUPPE 2

- E Lean Infoview

> GEB

> GLÜEDERUNG

> ZEITACHSE

Talklean 2, U

```
188 --θ + m - m needs to be proven by induction
189 lemma zero_add (n : MyNat) : add zero n = n := by
190   induction n with
191   | zero =>
192     rfl
193   | succ n' ih =>
194     rw [add] -----rewrite add
195     dsimp only -----definitionally simplify
196     rw [ih] -----rewrite using identity ih
197
198 --Multiplication of natural numbers, defined by recursion on the second factor
199 def mul (m n : MyNat) : MyNat := by
200   cases n with
201   | zero => exact zero
202   | succ n' => exact add (mul m n') m
203
204 --m * 1 = m
205 lemma mul_one (m : MyNat) : mul m (succ zero) = m := by
206   rw [mul]
207   dsimp only
208   rw [mul]
209   dsimp only
210   rw [zero_add]
211
212 --1 * m = m
213 lemma one_mul (n : MyNat) : mul (succ zero) n = n := by
214   induction n with
215   | zero =>
216     rfl
217   | succ n' ih =>
218     
```

Lean Infoview

- ▼ Talklean:219:4
- ▼ Tactic state
- 1 goal
- ▼ case succ

 - n' : MyNat
 - ih : mul zero.succ n' = n'
 - |- mul zero.succ n'.succ = n'.succ

- ▼ Messages (1)
- ▼ Talklean:219:4

 - unexpected end of input; expected '?', '_' or '{'

- All Messages (13)

Restart File

Zeile 219, Spalte 5 Leerzeichen: 2 UTF-8 LF lean4

master 2 ▲ 0 ○ 11 ⏪

Windows Taskbar

DEU DEU 21:58

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

Explorer

GlÖFFNET... [nicht gespeichert]

GRUPPE 1

- Talklean GEB 1, U

GRUPPE 2

- Lean Infoview

> GEB

> GUEDERUNG

> ZETACHSE

Talklean 1, U

```
188 --0 + m - m needs to be proven by induction
189 lemma zero_add (n : MyNat) : add zero n = n := by
190   induction n with
191   | zero =>
192     rfl
193   | succ n' ih =>
194     rw [add] -----rewrite add
195     dsimp only -----definitionally simplify
196     rw [ih] -----rewrite using identity ih
197
198 --Multiplication of natural numbers, defined by recursion on the second factor
199 def mul (m n : MyNat) : MyNat := by
200   cases n with
201   | zero => exact zero
202   | succ n' => exact add (mul m n') m
203
204 --m * 1 = m
205 lemma mul_one (m : MyNat) : mul m (succ zero) = m := by
206   rw [mul]
207   dsimp only
208   rw [mul]
209   dsimp only
210   rw [zero_add]
211
212 --1 * m = m
213 lemma one_mul (n : MyNat) : mul (succ zero) n = n := by
214   induction n with
215   | zero =>
216     rfl
217   | succ n' ih =>
218     rw [mul]
219     dsimp only
220
221
```

Lean Infoview

Talklean:221:4

Tactic state

1 goal

case succ

n' : MyNat

ih : mul zero.succ n' = n'

|- add (mul zero.succ n') zero.succ = n'.succ

All Messages (12)

Restart File

Zeile 221, Spalte 5 Leerzeichen: 2 UTF-8 LF lean4 DEU 21:58

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

Explorer ▾ GlÖFFNET... [nicht gespeichert] GEB > Talklean > ...

GRUPPE 1 • TalkLean GEB 1, U

GRUPPE 2 Lean Infoview

> GEB

> GUEDERUNG

> ZEITACHSE

188 --0 + m - m needs to be proven by induction

189 lemma zero_add (n : MyNat) : add zero n = n := by

190 induction n with

191 | zero =>

192 rfl

193 | succ n' ih =>

194 rw [add] -----rewrite add

195 dsimp only -----definitionally simplify

196 rw [ih] -----rewrite using identity ih

197

198 --Multiplication of natural numbers, defined by recursion on the second factor

199 def mul (m n : MyNat) : MyNat := by

200 cases n with

201 | zero => exact zero

202 | succ n' => exact add (mul m n') m

203

204 --m * 1 = m

205 lemma mul_one (m : MyNat) : mul m (succ zero) = m := by

206 rw [mul]

207 dsimp only

208 rw [mul]

209 dsimp only

210 rw [zero_add]

211

212 --1 * m = m

213 lemma one_mul (n : MyNat) : mul (succ zero) n = n := by

214 induction n with

215 | zero =>

216 rfl

217 | succ n' ih =>

218 rw [mul]

219 dsimp only

220 rw [ih]

221

222

Lean Infoview

TalkLean:222-4

Tactic state

1 goal

case succ

n' : MyNat

ih : mul zero.succ n' = n'

|- add n' zero.succ = n'.succ

All Messages (12)

Restart File

Zeile 222, Spalte 5 Leerzeichen: 2 UTF-8 LF lean4

1 0 11 90

Windows Taskbar

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

Explorer

GRUPPE 1

- E Talklean GEB 1, U

GRUPPE 2

- E Lean Infoview

> GEB

> GRÜNDERUNG

> ZEITACHSE

Talklean 1, U

```
188 --0 + m - m needs to be proven by induction
189 lemma zero_add (n : MyNat) : add zero n = n := by
190   induction n with
191   | zero =>
192     rfl
193   | succ n' ih =>
194     rw [add] -----rewrite add
195     dsimp only -----definitionally simplify
196     rw [ih] -----rewrite using identity ih
197
198 --Multiplication of natural numbers, defined by recursion on the second factor
199 def mul (m n : MyNat) : MyNat := by
200   cases n with
201   | zero => exact zero
202   | succ n' => exact add (mul m n') m
203
204 --m * 1 = m
205 lemma mul_one (m : MyNat) : mul m (succ zero) = m := by
206   rw [mul]
207   dsimp only
208   rw [mul]
209   dsimp only
210   rw [zero_add]
211
212 --1 * m = m
213 lemma one_mul (n : MyNat) : mul (succ zero) n = n := by
214   induction n with
215   | zero =>
216     rfl
217   | succ n' ih =>
218     rw [mul]
219     dsimp only
220     rw [ih]
221     rw [add]
222     dsimp only
223
224 |
```

Lean Infoview

Talklean:224:4

Tactic state

1 goal

case succ

n' : MyNat

ih : mul zero.succ n' = n'

|- (add n' zero).succ = n'.succ

All Messages (12)

Restart File

Zeile 224, Spalte 5 Leerzeichen: 2 UTF-8 LF lean4

1 ▲ 0 ◑ 11 ⌂ 0

Windows Taskbar

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

GEB > Talklean > ...

Explorer ▾ GÖFFNET... [nicht gespeichert] GRUPPE 1 • E Talklean GEB U GRUPPE 2 E Lean Infoview > GEB > GUEDERUNG > ZETACHSE

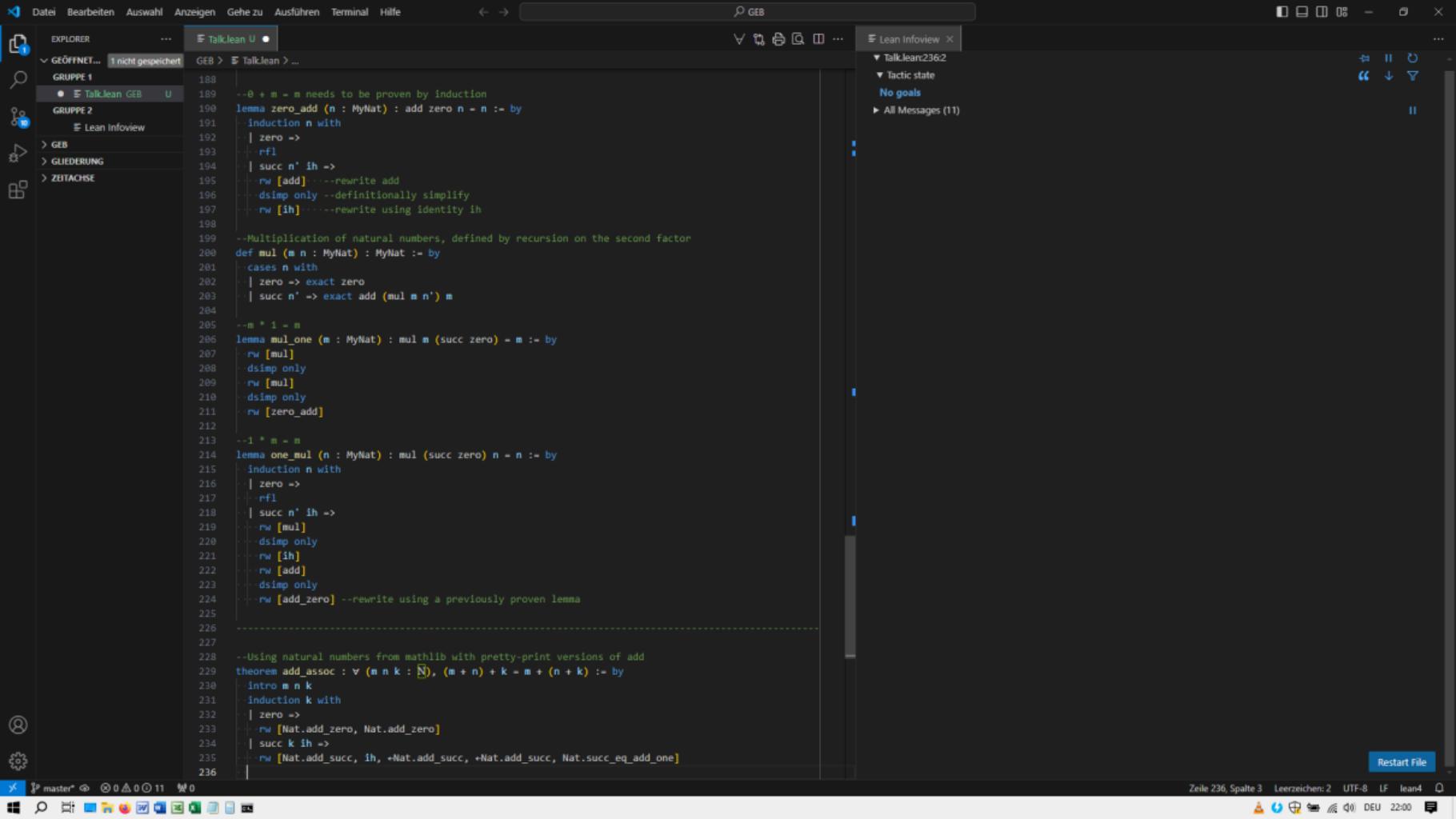
Lean Infoview ▾

Talklean:225-2
Tactic state
No goals
All Messages (11)

```
188 --0 + m - m needs to be proven by induction
189 lemma zero_add (n : MyNat) : add zero n = n := by
190   induction n with
191   | zero =>
192     rfl
193   | succ n' ih =>
194     rw [add] -----rewrite add
195     dsimp only -----definitionally simplify
196     rw [ih] -----rewrite using identity ih
197
198 --Multiplication of natural numbers, defined by recursion on the second factor
199 def mul (m n : MyNat) : MyNat := by
200   cases n with
201   | zero => exact zero
202   | succ n' => exact add (mul m n') m
203
204 --m * 1 = m
205 lemma mul_one (m : MyNat) : mul m (succ zero) = m := by
206   rw [mul]
207   dsimp only
208   rw [mul]
209   dsimp only
210   rw [zero_add]
211
212 --1 * m = m
213 lemma one_mul (n : MyNat) : mul (succ zero) n = n := by
214   induction n with
215   | zero =>
216     rfl
217   | succ n' ih =>
218     rw [mul]
219     dsimp only
220     rw [ih]
221     rw [add]
222     dsimp only
223     rw [add_zero] --rewrite using a previously proven lemma
224
225 |
```

Restart File

Zeile 225, Spalte 3 Leerzeichen: 2 UTF-8 LF lean4



File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

padic_systems

EXPLORER ...

GRUPPE 1

- nat_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic_rationals.lean U
- padic_integers.lean U

GRUPPE 2

- padic_integers.lean U
- padic_integers.lean U
- padic_systems.lean U
- Notes.lean Padic...

GRUPPE 2

Lean Infoview

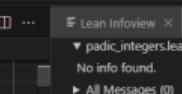
No info found.

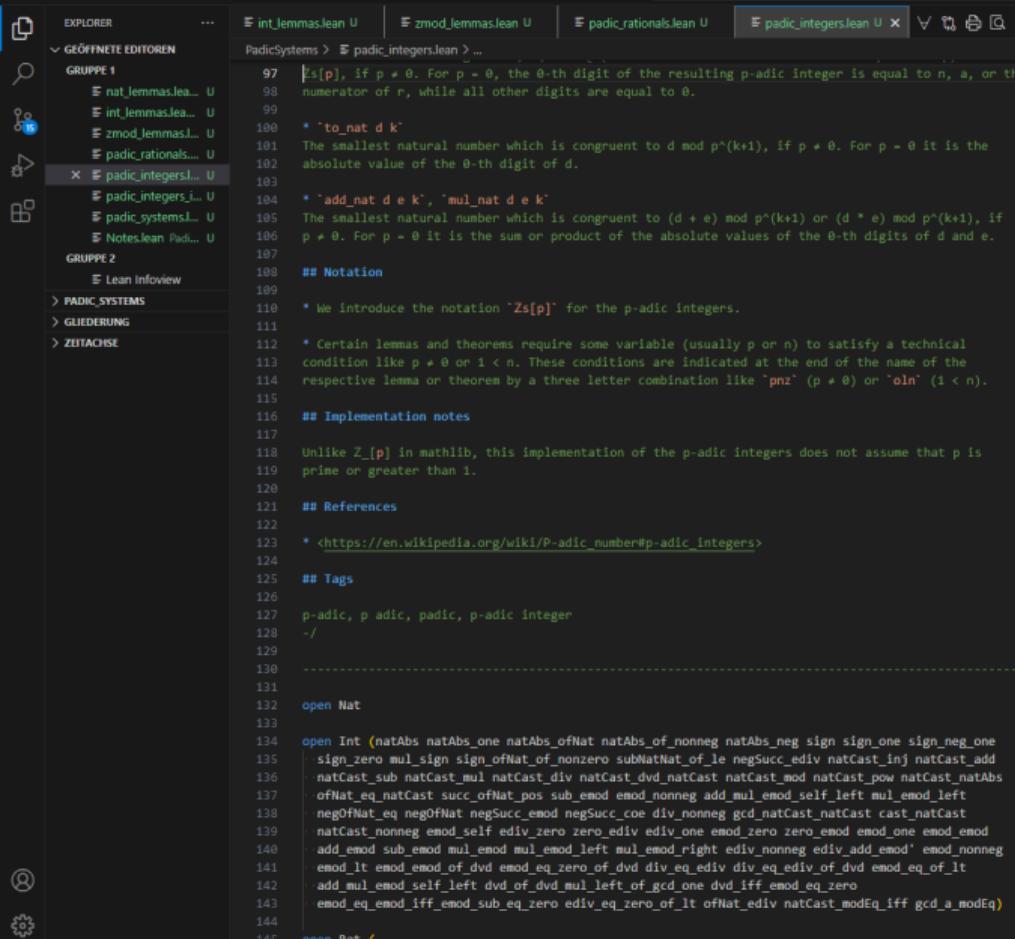
All Messages (0)

Restart File

padicSystems > padic_integers.lean > ...

```
1 |-
2 Copyright (c) 2023 Mario Weitzer. All rights reserved.
3 Released under Apache 2.0 license as described in the file LICENSE.
4 Authors: Mario Weitzer
5 -/
6
7 import PadicSystems.nat_lemmas
8 import PadicSystems.int_lemmas
9 import PadicSystems.zmod_lemmas
10 import PadicSystems.padic_rationals
11
12 set_option autoImplicit false
13
14 /-
15 ## Overview
16
17 This file implements the p-adic integers `Zs[p]`, where  $p : \mathbb{N}$  (not necessarily prime and not necessarily greater than 1), as a sequence  $d : \mathbb{N} \rightarrow \text{zmod } p$  of digits. The 's' in  $Zs[p]$  stands for both "sequence" and "sub" and distinguishes the p-adic integers as defined here from the p-adic integers  $\mathbb{Z}_{[p]}$  in mathlib. In contrast to  $\mathbb{Z}_{[p]}$ , the entire theory established in this file around  $Zs[p]$  is computable.
18
19 The main theorems proved in this file are
20
21 *  $Zs[p]$  is inhabited.
22
23 *  $Zs[p]$  forms a commutative ring
24
25 * If  $p$  is prime,  $Zs[p]$  is isomorphic to  $\mathbb{Z}_{[p]}$  (proved in the separate file padic_integers_isomorphism to avoid any non-computable theorems in this file).
26
27 *  $Zs[\theta] = \mathbb{Z}^N$ .
28
29 *  $Zs[1] = 0$ .
30
31 *  $Zs[p]$  is non-trivial iff  $p \neq 1$ .
32
33 *  $Zs[p]$  is a subsingleton iff  $p = 1$ .
34
35 * There is an embedding  $\text{rats}[p] \rightarrow Zs[p]$  of the p-adic rationals into the p-adic integers.
36
37 * The embedding of the p-adic rationals into the p-adic integers is injective if  $p \neq 1$ .
38
39 * If  $p \neq 1$ , the canonical embeddings of  $\text{nat}$ ,  $\text{int}$ , and  $\text{rat}$  of  $\mathbb{N}$ ,  $\mathbb{Z}$ , and  $\mathbb{Q}$  into  $Zs[p]$  are injective.
40
41 * The characteristic of  $Zs[p]$  is 0 iff  $p \neq 1$ .
42
43 *  $Zs[p]$  is isomorphic to  $Zs[u] \times Zs[v]$  if  $p = u * v$  and  $u$  and  $v$  are coprime.
```





File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

padic_systems

EXPLORER ...

GRUPPE 1

- nat_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic_rationals.lean U
- padic_integers.lean U
- padic_integers_.lean U
- padic_systems.lean U
- Notes.lean Padic...

GRUPPE 2

- Lean Infoview

PADIC SYSTEMS

GLIEDERUNG

ZUTASCHEN

int_lemmas.lean U | zmod_lemmas.lean U | padic_rationals.lean U | padic_integers.lean U | Lean Infoview

PadicSystems > padic_integers.lean > {} p > {} padic_integer > instance : CommRing Zs[p]

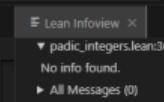
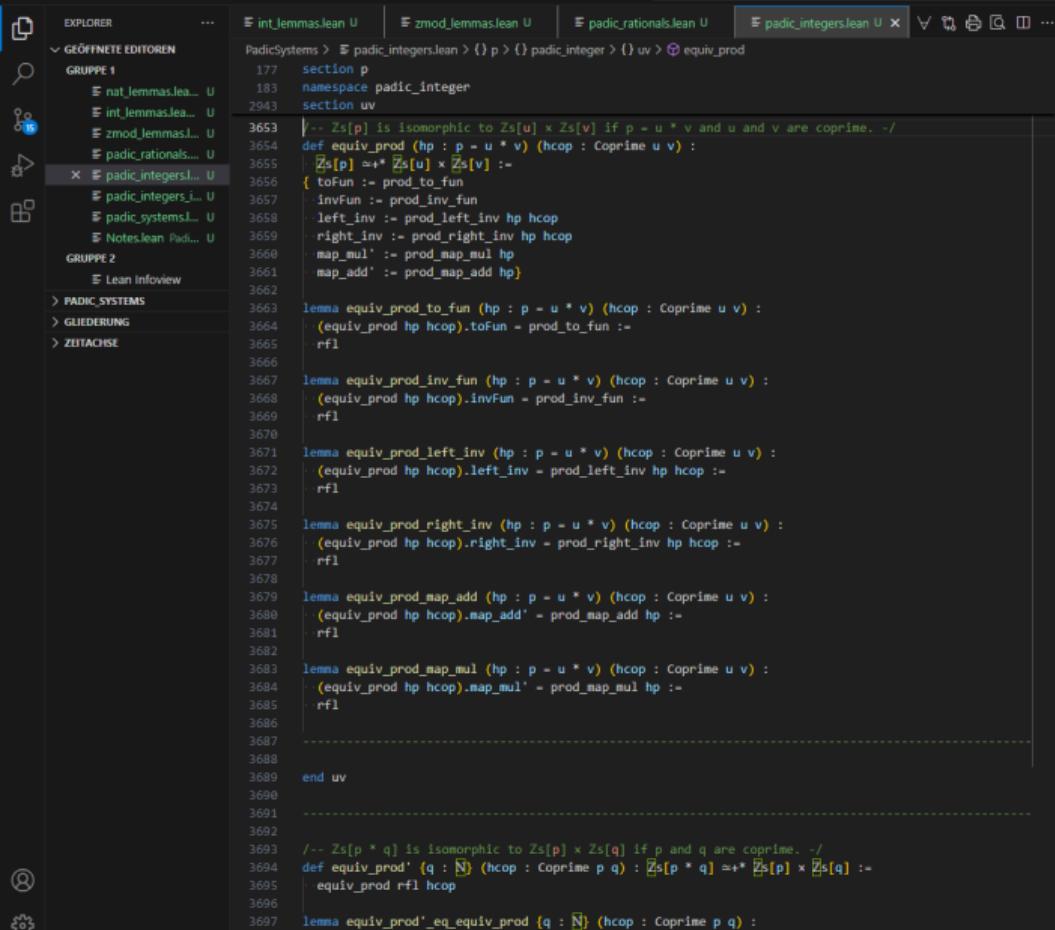
```
177 section p
183 namespace padic_integer
1844 /-- Zs[p] is a commutative ring. -/
1845 instance : CommRing Zs[p] :=
1846 { add := padic_integer.add,
1847   add_assoc := padic_integer.add_assoc,
1848   zero := padic_integer.zero,
1849   zero_add := padic_integer.zero_add,
1850   add_zero := padic_integer.add_zero,
1851   nsmul := padic_integer.nsmul,
1852   nsmul_zero := padic_integer.nsmul_zero,
1853   nsmul_succ := padic_integer.nsmul_succ,
1854   neg := padic_integer.neg,
1855   sub := padic_integer.sub,
1856   sub_eq_add_neg := padic_integer.sub_eq_add_neg,
1857   zsmul := padic_integer.zsmul,
1858   zsmul_zero' := padic_integer.zsmul_zero',
1859   zsmul_succ' := padic_integer.zsmul_succ',
1860   zsmul_neg' := padic_integer.zsmul_neg',
1861   neg_add_cancel := padic_integer.neg_add_cancel,
1862   add_comm := padic_integer.add_comm,
1863   natCast := padic_integer.natCast,
1864   intCast := padic_integer.intCast,
1865   one := padic_integer.one,
1866   natCast_zero := padic_integer.natCast_zero,
1867   natCast_succ := padic_integer.natCast_succ,
1868   intCast_ofNat := padic_integer.intCast_ofNat,
1869   intCast_negSucc := padic_integer.intCast_negSucc,
1870   mul := padic_integer.mul,
1871   mul_assoc := padic_integer.mul_assoc,
1872   zero_mul := padic_integer.zero_mul,
1873   mul_zero := padic_integer.mul_zero,
1874   one_mul := padic_integer.one_mul,
1875   mul_one := padic_integer.mul_one,
1876   npow := padic_integer.npow,
1877   npow_zero := padic_integer.npow_zero,
1878   npow_succ := padic_integer.npow_succ,
1879   left_distrib := padic_integer.left_distrib,
1880   right_distrib := padic_integer.right_distrib,
1881   mul_comm := padic_integer.mul_comm}
1882
1883 -----
1884 /-
1885 In this section is shown that Zs[0] = Z^N and Zs[1] = 0, Zs[p] is non-trivial iff p ≠ 1, and Zs[p]
1886 is a subsingleton iff p = 1.
1887 -/
1888
1889 /-- Zs[0] = Z^N. -/
1890
```

No info found.

All Messages (0)

Restart File

Zeile 1844, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:02



File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

padic_systems

EXPLORER ...

GRUPPE 1

- nat_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic_rationals.lean U
- padic_integers.lean U
- padic_systems.lean U
- Notes.lean Padic... U

GRUPPE 2

- Lean Infoview

PADIC SYSTEMS

GLIEDERUNG

ZUTAUSCH

int_lemmas.lean U | zmod_lemmas.lean U | padic_rationals.lean U | padic_integers.lean U | equiv_pow

177 section p
183 namespace padic_integer
3718 section ku
4093 *-- Zs[p] is isomorphic to Zs[u] if u = p^k and p - 1 ∨ 0 < k. --*
def equiv_pow' (hp : u = p^k) (hpk : p = 1 ∨ 0 < k) : Zs[p] ≈* Zs[u] :=
4095 { toFun := pow_to_fun
invFun := pow_inv_fun
left_inv := pow_left_inv hp hpk
right_inv := pow_right_inv hp hpk
map_mul' := pow_map_mul hp hpk
map_add' := pow_map_add hp hpk}
4101
4102
4103 end ku
4104
4105
4106
4107
4108 *-- Zs[p] is isomorphic to Zs[p^k] if p = 1 ∨ 0 < k. --*
def equiv_pow' (k : ℕ) (hpk : p = 1 ∨ 0 < k) : Zs[p] ≈* Zs[p^k] :=
4109 | equiv_pow rfl hpk
4110
4111 lemma equiv_pow'_eq_equiv_pow' (k : ℕ) (hpk : p = 1 ∨ 0 < k) :
4112 | equiv_pow' hpk = equiv_pow rfl hpk :=
4113 rfl
4114
4115
4116
4117
4118 /-
4119 In this section it is shown that Zs[p] is isomorphic to ∏ i, Zs[q_i] where q_1 * ... * q_n is the
prime power decomposition of p.
4120
4121 /-
4122 /-
4123 Possibly to do:
4124 <https://leanprover.zulipchat.com/#narrow/stream/113488-general/topic/How.20to.20deal.20with.20rewriting.20in.20types.3F>
4125
4126
4127 Yael Dillies:
4128 "Okay then I would highly suggest you use equivalences of the form
4129 $(\prod x : \alpha \oplus \beta, R x) \equiv (\prod a : \alpha, R (\text{Sum.inl } a)) \times (\prod b : \beta, R (\text{Sum.inr } b))$.
4130 This is morally equivalent to what you're trying to do, but it's much more idiomatic for
formalisation."
4131
4132 /-
4133 section u
4134
4135 universe u
4136
4137
4138

Lean Infoview

padic_integers.lean/4093:0

No info found.

All Messages (0)

Restart File

Zeile 4093, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:09

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe < > padic_systems

GRUPPE 1

- nat_lemmas.lean
- int_lemmas.lean
- zmod_lemmas.lean
- padic_rationals.lean
- padic_integers.lean

GRUPPE 2

- padic_integers.lean
- padic_integers.lean
- padic_systems.lean
- Notes.lean

Lean Infoview

padic_integers.lean/4494:0
No info found.
All Messages (0)

```
section p
namespace padic_integer
variables {p : PadicInteger}
variables {l i : ℕ}
variables {n : ℤ}
variables {h : l = prime_pow_decomp p}

-- Zs[p] is isomorphic to ℙ i, Zs[getD l i 1] if l = prime_pow_decomp p. -/
def equiv_prime_pow_decomp {l : List ℤ} (hl : l = prime_pow_decomp p) :
  Zs[p] ≃+* ℙ i, Zs[getD l i 1] := by
  rw [hl]
  apply rec_on_prime_powers p
  . intro n hn
  cases n with
  | zero =>
    rw [prime_pow_decomp_def_zero]
    apply RingEquiv.trans _ (RingEquiv.symm (Pi_equiv_head_of_length_eq_one _))
    . rw [length, length]
    . rw [List.get]
  | succ n =>
    rw [eq_head_prime_pow_decomp_iff_eq_factor_pow_count_nn (succ_ne_zero _)] at hn
    dsimp only at hn
    split_ifs at hn with hn1
    . rw [succ_eq_add_one, hn, prime_pow_decomp_def_one]
    apply RingEquiv.trans _ (RingEquiv.symm (Pi_equiv_head_of_length_eq_one _))
    . rw [length, length]
    . rw [List.get]
    have heq := eq_head_prime_pow_decomp_iff_eq_factor_pow_count_nn (succ_ne_zero n)
    rw [if_neg hn1] at heq
    dsimp only at heq
    rw [heq] at hn
    apply RingEquiv.trans _ (RingEquiv.symm (Pi_equiv_head_of_length_eq_one _))
    . exact (eq_head_prime_pow_decomp_iff_length_eq_one _).mp hn
    have get_zero_head_of_ne_nil {d : Type _} {h₀ : Inhabited d} {l : List d}
      (hl : l ≠ []) : List.get l 0, (length_pos.mpr hl) = head! l := by
      rw [head_of_head?]
      rw [get_eq_getElem, getElem_zero]
      exact head?.eq_head (length_pos.mp (length_pos.mpr hl))
      rw [get_zero_eq_head!_of_ne_nil (prime_pow_decomp_ne_nil _), +hn]
    intro n u v _ hn uv hv
    have hn := (eq_prod_and_coprime_of_prime_pow_decomp_eq_append hn uv).1
    have hcop := (eq_prod_and_coprime_of_prime_pow_decomp_eq_append hn uv).2
    rw [hn]
    apply RingEquiv.trans _ (RingEquiv.symm (Pi_append_equiv_prod _))
    apply RingEquiv.trans (equiv_prod hm hcop)
    exact equiv_prod_equiv hm hv

-- Zs[p] is isomorphic to ℙ i, Zs[getD (prime_pow_decomp p) i 1]. -/
def equiv_prime_pow_decomp' : Zs[p] ≃+* ℙ i, Zs[getD (prime_pow_decomp p) i 1] :=
  equiv_prime_pow_decomp rfl

/-
def d10 : Zs[10] := of_nat 1234567
4494
```

Restart File

Zeile 4494, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:12

File Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

padic_systems

EXPLORER ...

GRUPPE 1

- nat_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic_rationals.lean U
- padic_integers.lean U
- padic_integers_1.lean U
- padic_integers_2.lean U
- padic_systems.lean U
- Notes.lean Padic...

GRUPPE 2

- Lean Infowiew

PADIC SYSTEMS

GLIEDERUNG

ZUTASCHEN

int_lemmas.lean U | zmod_lemmas.lean U | padic_rationals.lean U | padic_integers.lean U | padic_integers.lean U

PadicSystems > padic_integers.lean > {} p > {} padic_integer

```
177 section p
183 namespace padic_integer
184
185 def d10 : Zs[10] := of_nat 1234567
186
187 lemma ten_eq_two_times_five : 10 = 2 * 5 := rfl
188 lemma coprime_two_five : Coprime 2 5 := rfl
189
190 #eval map (λ k => (equiv_prod ten_eq_two_times_five coprime_two_five) d10).1 k) (range 20)
191 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0]
192 #eval map (λ k => (equiv_prod ten_eq_two_times_five coprime_two_five) d10).2 k) (range 20)
193 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
194
195 #eval prime_pow_decomp 10
196 --[2, 5]
197
198 #eval map (λ k => equiv_prime_pow_decomp' d10 0 k) (range 20)
199 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]
200 #eval map (λ k => equiv_prime_pow_decomp' d10 1 k) (range 20)
201 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
202
203 def l10 := [2, 5]
204
205 lemma hl10 : l10 = prime_pow_decomp 10 := by
206 rfl
207
208 #eval map (λ k => (equiv_prime_pow_decomp hl10) d10 0 k) (range 20)
209 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]
210 #eval map (λ k => (equiv_prime_pow_decomp hl10) d10 1 k) (range 20)
211 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
212
213 def d60 : Zs[60] := of_nat 1234567
214
215 #eval prime_pow_decomp 60
216 --[4, 3, 5]
217
218 #eval map (λ k => equiv_prime_pow_decomp' d60 0 k) (range 20)
219 --[3, 1, 0, 2, 2, 1, 1, 3, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
220 #eval map (λ k => equiv_prime_pow_decomp' d60 1 k) (range 20)
221 --[1, 0, 2, 1, 1, 0, 2, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0]
222 #eval map (λ k => equiv_prime_pow_decomp' d60 2 k) (range 20)
223 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]
224
225 def l60 := [4, 3, 5]
226
227 lemma hl60 : l60 = prime_pow_decomp 60 := by
228 rfl
```

Lean Infowiew

padic_integers.lean:4544:0

Messages (1)

padic_integers.lean:4544:0

[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0]

All Messages (14)

Restart File

Zeile 4544, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:20

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

padic_systems

EXPLORER ...

GRUPPE 1

- nat_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic_rationals.lean U
- padic_integers.lean U
- padic_integers_1.lean U
- padic_integers_L.lean U
- Notes.lean Padic_U

GRUPPE 2

- Lean Infowiew

PADIC SYSTEMS

GLIEDERUNG

ZUTRACHSE

int_lemmas.lean U | zmod_lemmas.lean U | padic_rationals.lean U | padic_integers.lean U | padic_integers.lean U

PadicSystems > padic_integers.lean > {} p > {} padic_integer

177 section p
183 namespace padic_integer

4538
4539 def d10 : Zs[10] := of_nat 1234567
4540
4541 lemma ten_eq_two_times_five : 10 = 2 * 5 := rfl
4542 lemma coprime_two_five : Coprime 2 5 := rfl
4543
4544 #eval map (λ k -> (equiv_prod ten_eq_two_times_five coprime_two_five) d10).1 k) (range 20)
4545 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0]
4546 #eval map (λ k -> (equiv_prod ten_eq_two_times_five coprime_two_five) d10).2 k) (range 20)
4547 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4548
4549 #eval prime_pow_decomp 10
4550 --[2, 5]
4551
4552 #eval map (λ k -> equiv_prime_pow_decomp' d10 0 k) (range 20)
4553 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]
4554 #eval map (λ k -> equiv_prime_pow_decomp' d10 1 k) (range 20)
4555 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
4556
4557 def l10 := [2, 5]
4558
4559 lemma hl10 : l10 = prime_pow_decomp 10 := by
4560 rfl
4561
4562 #eval map (λ k -> (equiv_prime_pow_decomp hl10) d10 0 k) (range 20)
4563 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]
4564 #eval map (λ k -> (equiv_prime_pow_decomp hl10) d10 1 k) (range 20)
4565 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
4566
4567 def d60 : Zs[60] := of_nat 1234567
4568
4569 #eval prime_pow_decomp 60
4570 --[4, 3, 5]
4571
4572
4573 #eval map (λ k -> equiv_prime_pow_decomp' d60 0 k) (range 20)
4574 --[3, 1, 0, 2, 2, 1, 1, 3, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0]
4575 #eval map (λ k -> equiv_prime_pow_decomp' d60 1 k) (range 20)
4576 --[1, 0, 2, 1, 1, 1, 0, 2, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0]
4577 #eval map (λ k -> equiv_prime_pow_decomp' d60 2 k) (range 20)
4578 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4579
4580 def l60 := [4, 3, 5]
4581
4582 lemma hl60 : l60 = prime_pow_decomp 60 := by
4583 rfl
4584

Lean Infowiew

padic_integers.lean:4546:0

Messages (1)

padic_integers.lean:4546:0

[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

All Messages (14)

Restart File

Zeile 4546, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:20

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

padic_systems

EXPLORER ...

GRUPPE 1

- nat_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic_rationals.lean U
- padic_integers.lean U
- padic_integers.1 U
- padic_integersL. U
- Notes.lean Padic.. U

GRUPPE 2

- Lean Infowiew

PADIC SYSTEMS

GLIEDERUNG

ZUTASCHEN

int_lemmas.lean U | zmod_lemmas.lean U | padic_rationals.lean U | padic_integers.lean U | padic_integers.1 U | padic_integersL. U | Notes.lean Padic.. U

177 section p
183 namespace padic_integer
4538
4539 def d10 : Zs[10] := of_nat 1234567
4540
4541 lemma ten_eq_two_times_five : 10 = 2 * 5 := rfl
4542 lemma coprime_two_five : Coprime 2 5 := rfl
4543
4544 #eval map (λ k -> (equiv_prod ten_eq_two_times_five coprime_two_five) d10).1 k) (range 20)
--[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 0]
#eval map (λ k -> (equiv_prod ten_eq_two_times_five coprime_two_five) d10).2 k) (range 20)
--[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4545
4546
4547
4548 #eval prime_pow_decomp 10
4549 --[2, 5]
4550
4551
4552 #eval map (λ k -> equiv_prime_pow_decomp' d10 0 k) (range 20)
--[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0]
#eval map (λ k -> equiv_prime_pow_decomp' d10 1 k) (range 20)
--[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4553
4554
4555
4556
4557 def l10 := [2, 5]
4558
4559 lemma h110 : l10 = prime_pow_decomp 10 := by
4560 rfl
4561
4562 #eval map (λ k -> (equiv_prime_pow_decomp h110) d10 0 k) (range 20)
--[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0]
#eval map (λ k -> (equiv_prime_pow_decomp h110) d10 1 k) (range 20)
--[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4563
4564
4565
4566
4567
4568 def d60 : Zs[60] := of_nat 1234567
4569
4570 #eval prime_pow_decomp 60
4571 --[4, 3, 5]
4572
4573 #eval map (λ k -> equiv_prime_pow_decomp' d60 0 k) (range 20)
--[3, 1, 0, 2, 2, 1, 1, 3, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
#eval map (λ k -> equiv_prime_pow_decomp' d60 1 k) (range 20)
--[1, 0, 2, 1, 1, 1, 0, 2, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0]
#eval map (λ k -> equiv_prime_pow_decomp' d60 2 k) (range 20)
--[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4574
4575
4576
4577
4578
4579
4580 def l60 := [4, 3, 5]
4581
4582 lemma h160 : l60 = prime_pow_decomp 60 := by
4583 rfl
4584

Lean Infowiew

padic_integers.lean/4549:0

Messages (1)

padic_integers.lean/4549:0

[2, 5]

All Messages (14)

Restart File

Zeile 4549, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:21

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

padic_systems

EXPLORER ...

GRUPPE 1

- nat_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic_rationals.lean U
- padic_integers.lean U
- padic_integers... 1 U
- padic_integers... 2 U
- padic_integers... 3 U

GRUPPE 2

- Lean Infowiew

PADIC SYSTEMS

GLIEDERUNG

ZUTASCHEN

int_lemmas.lean U | zmod_lemmas.lean U | padic_rationals.lean U | padic_integers.lean U | padic_integers.lean U

PadicSystems > padic_integers.lean > {} p > {} padic_integer

```
177 section p
183 namespace padic_integer
184
185 def d10 : Zs[10] := of_nat 1234567
186
187 lemma ten_eq_two_times_five : 10 = 2 * 5 := rfl
188 lemma coprime_two_five : Coprime 2 5 := rfl
189
190 #eval map (λ k => (equiv_prod ten_eq_two_times_five coprime_two_five) d10).1 k) (range 20)
191 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0]
192 #eval map (λ k => (equiv_prod ten_eq_two_times_five coprime_two_five) d10).2 k) (range 20)
193 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]
194
195 #eval prime_pow_decomp 10
196 --[2, 5]
197
198 #eval map (λ k => equiv_prime_pow_decomp' d10 0 k) (range 20)
199 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0]
200 #eval map (λ k => equiv_prime_pow_decomp' d10 1 k) (range 20)
201 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
202
203 def l10 := [2, 5]
204
205 lemma hl10 : l10 = prime_pow_decomp 10 := by
206   rfl
207
208 #eval map (λ k => (equiv_prime_pow_decomp hl10) d10 0 k) (range 20)
209 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0]
210 #eval map (λ k => (equiv_prime_pow_decomp hl10) d10 1 k) (range 20)
211 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
212
213 def d60 : Zs[60] := of_nat 1234567
214
215 #eval prime_pow_decomp 60
216 --[4, 3, 5]
217
218 #eval map (λ k => equiv_prime_pow_decomp' d60 0 k) (range 20)
219 --[3, 1, 0, 2, 2, 1, 1, 3, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
220 #eval map (λ k => equiv_prime_pow_decomp' d60 1 k) (range 20)
221 --[1, 0, 2, 1, 1, 1, 0, 2, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0]
222 #eval map (λ k => equiv_prime_pow_decomp' d60 2 k) (range 20)
223 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
224
225 def l60 := [4, 3, 5]
226
227 lemma hl60 : l60 = prime_pow_decomp 60 := by
228   rfl
```

Lean Infowiew

padic_integers.lean:4552:0

Messages (1)

padic_integers.lean:4552:0

[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0]

All Messages (14)

Restart File

Zeile 4552, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:21

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

padic_systems

EXPLORER ...

GRUPPE 1

- int_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic_rationals.lean U
- padic_integers.lean U
- padic_integers.lean U
- padic_integers.lean U
- padic_integers.lean U

GRUPPE 2

- Lean Infowiew

PADIC SYSTEMS

GLIEDERUNG

ZUTASCHEN

int_lemmas.lean U | zmod_lemmas.lean U | padic_rationals.lean U | padic_integers.lean U | ...

PadicSystems > padic_integers.lean > {} p > {} padic_integer

```
177 section p
183 namespace padic_integer
184
185 def d10 : Zs[10] := of_nat 1234567
186
187 lemma ten_eq_two_times_five : 10 = 2 * 5 := rfl
188 lemma coprime_two_five : Coprime 2 5 := rfl
189
190 #eval map (λ k => (equiv_prod ten_eq_two_times_five coprime_two_five) d10).1 k) (range 20)
191 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0]
192 #eval map (λ k => (equiv_prod ten_eq_two_times_five coprime_two_five) d10).2 k) (range 20)
193 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
194
195 #eval prime_pow_decomp 10
196 --[2, 5]
197
198 #eval map (λ k => equiv_prime_pow_decomp' d10 0 k) (range 20)
199 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]
200 #eval map (λ k => equiv_prime_pow_decomp' d10 1 k) (range 20)
201 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
202
203 def l10 := [2, 5]
204
205 lemma hl10 : l10 = prime_pow_decomp 10 := by
206 rfl
207
208 #eval map (λ k => (equiv_prime_pow_decomp hl10) d10 0 k) (range 20)
209 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]
210 #eval map (λ k => (equiv_prime_pow_decomp hl10) d10 1 k) (range 20)
211 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
212
213 def d60 : Zs[60] := of_nat 1234567
214
215 #eval prime_pow_decomp 60
216 --[4, 3, 5]
217
218 #eval map (λ k => equiv_prime_pow_decomp' d60 0 k) (range 20)
219 --[3, 1, 0, 2, 2, 1, 1, 3, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
220 #eval map (λ k => equiv_prime_pow_decomp' d60 1 k) (range 20)
221 --[1, 0, 2, 1, 1, 0, 2, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0]
222 #eval map (λ k => equiv_prime_pow_decomp' d60 2 k) (range 20)
223 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]
224
225 def l60 := [4, 3, 5]
226
227 lemma hl60 : l60 = prime_pow_decomp 60 := by
228 rfl
```

Lean Infowiew

padic_integers.lean:4554:0

Messages (1)

padic_integers.lean:4554:0

[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

All Messages (14)

Restart File

Zeile 4554, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:21

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe ↺ ↻ padic_systems

EXPLORER ...

GRUPPE 1

- nat_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic_rationals.lean U
- padic_integers.lean U
- padic_integers_1.lean U
- padic_integers_2.lean U
- padic_systems.lean U
- Notes.lean Padic U

GRUPPE 2

- Lean Infoview

PADIC SYSTEMS

GLIEDERUNG

ZUTASCHEN

int_lemmas.lean U | zmod_lemmas.lean U | padic_rationals.lean U | padic_integers.lean U | padic_integers.lean U | ...

PadicSystems > padic_integers.lean > {} p > {} padic_integer

```
177 section p
183 namespace padic_integer
4538
4539 def d10 : Zs[10] := of_nat 1234567
4540
4541 lemma ten_eq_two_times_five : 10 = 2 * 5 := rfl
4542 lemma coprime_two_five : Coprime 2 5 := rfl
4543
4544 #eval map (λ k -> (equiv_prod ten_eq_two_times_five coprime_two_five) d10).1 k) (range 20)
--[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0]
4545 #eval map (λ k -> (equiv_prod ten_eq_two_times_five coprime_two_five) d10).2 k) (range 20)
--[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4546
4547 #eval prime_pow_decomp 10
4548 --[2, 5]
4549
4550 #eval map (λ k -> equiv_prime_pow_decomp' d10 0 k) (range 20)
--[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]
4551 #eval map (λ k -> equiv_prime_pow_decomp' d10 1 k) (range 20)
--[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
4552
4553 def l10 := [2, 5]
4554
4555 lemma hl10 : l10 = prime_pow_decomp 10 := by
4556 rfl
4557
4558 #eval map (λ k -> (equiv_prime_pow_decomp hl10) d10 0 k) (range 20)
--[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]
4559 #eval map (λ k -> (equiv_prime_pow_decomp hl10) d10 1 k) (range 20)
--[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
4560
4561 def d60 : Zs[60] := of_nat 1234567
4562
4563 #eval prime_pow_decomp 60
4564 --[4, 3, 5]
4565
4566 #eval map (λ k -> equiv_prime_pow_decomp' d60 0 k) (range 20)
--[3, 1, 0, 2, 2, 1, 1, 3, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
4567 #eval map (λ k -> equiv_prime_pow_decomp' d60 1 k) (range 20)
--[1, 0, 2, 1, 1, 1, 0, 2, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0]
4568 #eval map (λ k -> equiv_prime_pow_decomp' d60 2 k) (range 20)
--[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4569
4570 def l60 := [4, 3, 5]
4571
4572 lemma hl60 : l60 = prime_pow_decomp 60 := by
4573 rfl
```

Lean Infoview

padic_integers.lean:4559:0

No info found.

All Messages (14)

Restart File

Zeile 4559, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:21

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

padic_systems

EXPLORER ...

GRUPPE 1

- nat_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic_rationals.lean U
- padic_integers.lean U
- padic_integers_1.lean U
- padic_integers_L.lean U
- Notes.lean Padic..U

GRUPPE 2

- Lean Infowiew

PADIC SYSTEMS

GLIEDERUNG

ZUTASCHEN

int_lemmas.lean U | zmod_lemmas.lean U | padic_rationals.lean U | padic_integers.lean U | padic_integers_1.lean U | padic_integers_L.lean U | Notes.lean Padic..U

177 section p
183 namespace padic_integer
4538
4539 def d10 : Zs[10] := of_nat 1234567
4540
4541 lemma ten_eq_two_times_five : 10 = 2 * 5 := rfl
4542 lemma coprime_two_five : Coprime 2 5 := rfl
4543
4544 #eval map (λ k -> (equiv_prod ten_eq_two_times_five coprime_two_five) d10).1 k) (range 20)
--[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0]
4545 #eval map (λ k -> (equiv_prod ten_eq_two_times_five coprime_two_five) d10).2 k) (range 20)
--[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4546
4547 #eval prime_pow_decomp 10
--[2, 5]
4548
4549 #eval map (λ k -> equiv_prime_pow_decomp' d10 0 k) (range 20)
--[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0]
4550 #eval map (λ k -> equiv_prime_pow_decomp' d10 1 k) (range 20)
--[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
4551
4552 def l10 := [2, 5]
4553
4554 lemma hl10 : l10 = prime_pow_decomp 10 := by
| rfl
4555 #eval map (λ k -> (equiv_prime_pow_decomp hl10) d10 0 k) (range 20)
--[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0]
4556 #eval map (λ k -> (equiv_prime_pow_decomp hl10) d10 1 k) (range 20)
--[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
4557
4558 def d60 : Zs[60] := of_nat 1234567
4559
4560 #eval prime_pow_decomp 60
--[4, 3, 5]
4561
4562 #eval map (λ k -> equiv_prime_pow_decomp' d60 0 k) (range 20)
--[3, 1, 0, 2, 2, 1, 1, 3, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
4563 #eval map (λ k -> equiv_prime_pow_decomp' d60 1 k) (range 20)
--[1, 0, 2, 1, 1, 1, 0, 2, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0]
4564 #eval map (λ k -> equiv_prime_pow_decomp' d60 2 k) (range 20)
--[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4565
4566
4567 def l60 := [4, 3, 5]
4568
4569 lemma hl60 : l60 = prime_pow_decomp 60 := by
| rfl

Lean Infowiew

padic_integers.lean:4562:0

Messages (1)

padic_integers.lean:4562:0

[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0]

All Messages (14)

Restart File

Zeile 4562, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:21

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

padic_systems

EXPLORER ...

GRUPPE 1

- nat_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic_rationals.lean U
- padic_integers.lean U
- padic_integers_1.lean U
- padic_integers_L.lean U
- Notes.lean Padic..U

GRUPPE 2

- Lean Infowiew

PADIC SYSTEMS

GLIEDERUNG

ZUTASCHEN

int_lemmas.lean U | zmod_lemmas.lean U | padic_rationals.lean U | padic_integers.lean U | padic_integers_1.lean U | padic_integers_L.lean U | Notes.lean Padic..U

177 section p
183 namespace padic_integer
4538
4539 def d10 : Zs[10] := of_nat 1234567
4540
4541 lemma ten_eq_two_times_five : 10 = 2 * 5 := rfl
4542 lemma coprime_two_five : Coprime 2 5 := rfl
4543
4544 #eval map (λ k -> (equiv_prod ten_eq_two_times_five coprime_two_five) d10).1 k) (range 20)
--[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0]
4545 #eval map (λ k -> (equiv_prod ten_eq_two_times_five coprime_two_five) d10).2 k) (range 20)
--[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4546
4547 #eval prime_pow_decomp 10
4548 --[2, 5]
4549
4550 #eval map (λ k -> equiv_prime_pow_decomp' d10 0 k) (range 20)
--[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]
4551 #eval map (λ k -> equiv_prime_pow_decomp' d10 1 k) (range 20)
--[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
4552
4553 def l10 := [2, 5]
4554
4555 lemma hl10 : l10 = prime_pow_decomp 10 := by
4556 rfl
4557
4558 #eval map (λ k -> (equiv_prime_pow_decomp hl10) d10 0 k) (range 20)
--[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]
4559 #eval map (λ k -> (equiv_prime_pow_decomp hl10) d10 1 k) (range 20)
--[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
4560
4561 def d60 : Zs[60] := of_nat 1234567
4562
4563 #eval prime_pow_decomp 60
4564 --[4, 3, 5]
4565
4566 #eval map (λ k -> equiv_prime_pow_decomp' d60 0 k) (range 20)
--[3, 1, 0, 2, 2, 1, 1, 3, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
4567 #eval map (λ k -> equiv_prime_pow_decomp' d60 1 k) (range 20)
--[1, 0, 2, 1, 1, 1, 0, 2, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0]
4568 #eval map (λ k -> equiv_prime_pow_decomp' d60 2 k) (range 20)
--[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4569
4570 def l60 := [4, 3, 5]
4571
4572 lemma hl60 : l60 = prime_pow_decomp 60 := by
4573 rfl

Lean Infowiew

padic_integers.lean:4564:0

Messages (1)

padic_integers.lean:4564:0

[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

All Messages (14)

Restart File

Zeile 4564, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:21

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

padic_systems

EXPLORER ...

GRUPPE 1

- nat_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic_rationals.lean U
- padic_integers.lean U
- padic_integer... 1 U
- padic_systems.lean U
- Notes.lean Padic... U

GRUPPE 2

- Lean Infowiew

PADIC SYSTEMS

GLIEDERUNG

ZUTASCHEN

int_lemmas.lean U | zmod_lemmas.lean U | padic_rationals.lean U | padic_integers.lean U | padic_integer... 1 U | padic_systems.lean U | Notes.lean Padic... U

padicSystems > padic_integers.lean > {} p > {} padic_integer

```
177 section p
183 namespace padic_integer
184
185 def d10 : Zs[10] := of_nat 1234567
186
187 lemma ten_eq_two_times_five : 10 = 2 * 5 := rfl
188 lemma coprime_two_five : Coprime 2 5 := rfl
189
190 #eval map (λ k => (equiv_prod ten_eq_two_times_five coprime_two_five) d10).1 k) (range 20)
191 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0]
192 #eval map (λ k => (equiv_prod ten_eq_two_times_five coprime_two_five) d10).2 k) (range 20)
193 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
194
195 #eval prime_pow_decomp 10
196 --[2, 5]
197
198 #eval map (λ k => equiv_prime_pow_decomp' d10 0 k) (range 20)
199 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]
200 #eval map (λ k => equiv_prime_pow_decomp' d10 1 k) (range 20)
201 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
202
203 def l10 := [2, 5]
204
205 lemma hl10 : l10 = prime_pow_decomp 10 := by
206 rfl
207
208 #eval map (λ k => (equiv_prime_pow_decomp hl10) d10 0 k) (range 20)
209 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]
210 #eval map (λ k => (equiv_prime_pow_decomp hl10) d10 1 k) (range 20)
211 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
212
213 def d60 : Zs[60] := of_nat 1234567
214
215 #eval prime_pow_decomp 60
216 --[4, 3, 5]
217
218 #eval map (λ k => equiv_prime_pow_decomp' d60 0 k) (range 20)
219 --[3, 1, 0, 2, 2, 1, 1, 3, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
220 #eval map (λ k => equiv_prime_pow_decomp' d60 1 k) (range 20)
221 --[1, 0, 2, 1, 1, 1, 0, 2, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0]
222 #eval map (λ k => equiv_prime_pow_decomp' d60 2 k) (range 20)
223 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]
224
225 def l60 := [4, 3, 5]
226
227 lemma hl60 : l60 = prime_pow_decomp 60 := by
228 rfl
```

Lean Infowiew

padic_integers.lean:4570:0

Messages (1)

padic_integers.lean:4570:0

[4, 3, 5]

All Messages (14)

Restart File

Zeile 4570, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:21

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

padic_systems

EXPLORER ...

GRUPPE 1

- nat_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic_rationals.lean U
- padic_integers.lean U
- padic_integers..1 U
- padic_integers..2 U
- padic_integers..3 U
- padic_integers..4 U

GRUPPE 2

- Lean Infowiew

PADIC SYSTEMS

GLIEDERUNG

ZUTASCHEN

int_lemmas.lean U | zmod_lemmas.lean U | padic_rationals.lean U | padic_integers.lean U | padic_integers.lean U

PadicSystems > padic_integers.lean > {} p > {} padic_integer

```
177 section p
183 namespace padic_integer
184
185 def d10 : Zs[10] := of_nat 1234567
186
187 lemma ten_eq_two_times_five : 10 = 2 * 5 := rfl
188 lemma coprime_two_five : Coprime 2 5 := rfl
189
190 #eval map (λ k -> (equiv_prod ten_eq_two_times_five coprime_two_five) d10).1 k) (range 20)
191 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0]
192 #eval map (λ k -> (equiv_prod ten_eq_two_times_five coprime_two_five) d10).2 k) (range 20)
193 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]
194
195 #eval prime_pow_decomp 10
196 --[2, 5]
197
198 #eval map (λ k -> equiv_prime_pow_decomp' d10 0 k) (range 20)
199 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]
200 #eval map (λ k -> equiv_prime_pow_decomp' d10 1 k) (range 20)
201 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
202
203 def l10 := [2, 5]
204
205 lemma hl10 : l10 = prime_pow_decomp 10 := by
206 rfl
207
208 #eval map (λ k -> (equiv_prime_pow_decomp hl10) d10 0 k) (range 20)
209 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]
210 #eval map (λ k -> (equiv_prime_pow_decomp hl10) d10 1 k) (range 20)
211 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
212
213 def d60 : Zs[60] := of_nat 1234567
214
215 #eval prime_pow_decomp 60
216 --[4, 3, 5]
217
218 #eval map (λ k -> equiv_prime_pow_decomp' d60 0 k) (range 20)
219 --[3, 1, 0, 2, 2, 1, 1, 3, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0]
220 #eval map (λ k -> equiv_prime_pow_decomp' d60 1 k) (range 20)
221 --[1, 0, 2, 1, 1, 1, 0, 2, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0]
222 #eval map (λ k -> equiv_prime_pow_decomp' d60 2 k) (range 20)
223 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
224
225 def l60 := [4, 3, 5]
226
227 lemma hl60 : l60 = prime_pow_decomp 60 := by
228 rfl
```

Lean Infowiew

padic_integers.lean:4573:0

Messages (1)

padic_integers.lean:4573:0

[3, 1, 0, 2, 2, 1, 1, 3, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]

All Messages (14)

Restart File

Zeile 4573, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:21

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

padic_systems

EXPLORER ...

GRUPPE 1

- nat_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic_rationals.lean U
- padic_integers.lean U
- padic_integers_1.lean U
- padic_integers_L.lean U
- Notes.lean Padic..U

GRUPPE 2

- Lean Infowiew

PADIC SYSTEMS

GLIEDERUNG

ZUTASCHEN

int_lemmas.lean U | zmod_lemmas.lean U | padic_rationals.lean U | padic_integers.lean U | padic_integers.lean U

PadicSystems > padic_integers.lean > {} p > {} padic_integer

```
177 section p
183 namespace padic_integer
184
185 def d10 : Zs[10] := of_nat 1234567
186
187 lemma ten_eq_two_times_five : 10 = 2 * 5 := rfl
188 lemma coprime_two_five : Coprime 2 5 := rfl
189
190 #eval map (λ k => (equiv_prod ten_eq_two_times_five coprime_two_five) d10).1 k) (range 20)
191 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0]
192 #eval map (λ k => (equiv_prod ten_eq_two_times_five coprime_two_five) d10).2 k) (range 20)
193 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]
194
195 #eval prime_pow_decomp 10
196 --[2, 5]
197
198 #eval map (λ k => equiv_prime_pow_decomp' d10 0 k) (range 20)
199 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]
200 #eval map (λ k => equiv_prime_pow_decomp' d10 1 k) (range 20)
201 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
202
203 def l10 := [2, 5]
204
205 lemma h110 : l10 = prime_pow_decomp 10 := by
206   rfl
207
208 #eval map (λ k => (equiv_prime_pow_decomp h110) d10 0 k) (range 20)
209 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]
210 #eval map (λ k => (equiv_prime_pow_decomp h110) d10 1 k) (range 20)
211 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
212
213 def d60 : Zs[60] := of_nat 1234567
214
215 #eval prime_pow_decomp 60
216 --[4, 3, 5]
217
218 #eval map (λ k => equiv_prime_pow_decomp' d60 0 k) (range 20)
219 --[3, 1, 0, 2, 2, 1, 1, 3, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
220 #eval map (λ k => equiv_prime_pow_decomp' d60 1 k) (range 20)
221 --[1, 0, 2, 1, 1, 0, 2, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0]
222 #eval map (λ k => equiv_prime_pow_decomp' d60 2 k) (range 20)
223 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]
224
225 def l100 := [4, 3, 5]
226
227 lemma h160 : l100 = prime_pow_decomp 60 := by
228   rfl
```

Lean Infowiew

padic_integers.lean:4575:0

Messages (1)

padic_integers.lean:4575:0

[1, 0, 2, 1, 1, 1, 0, 2, 2, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0]

All Messages (14)

Restart File

Zeile 4575, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:21

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe ↺ ↻ padic_systems

EXPLORER ...

GRUPPE 1

- nat_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic_rationals.lean U
- padic_integers.lean U
- padic_integers_1.lean U
- padic_integers_2.lean U
- padic_integers_3.lean U
- padic_integers_4.lean U

GRUPPE 2

- Lean Infowiew

PADIC SYSTEMS

GLIEDERUNG

ZUTASCHEN

int_lemmas.lean U | zmod_lemmas.lean U | padic_rationals.lean U | padic_integers.lean U | padic_integers_1.lean U | padic_integers_2.lean U | padic_integers_3.lean U | padic_integers_4.lean U

PadicSystems > padic_integers.lean > {} p > {} padic_integer

```
177 section p
183 namespace padic_integer
184
185 def d10 : Zs[10] := of_nat 1234567
186
187 lemma ten_eq_two_times_five : 10 = 2 * 5 := rfl
188 lemma coprime_two_five : Coprime 2 5 := rfl
189
190 #eval map (λ k -> (equiv_prod ten_eq_two_times_five coprime_two_five) d10).1 k) (range 20)
191 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0]
192 #eval map (λ k -> (equiv_prod ten_eq_two_times_five coprime_two_five) d10).2 k) (range 20)
193 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
194
195 #eval prime_pow_decomp 10
196 --[2, 5]
197
198 #eval map (λ k -> equiv_prime_pow_decomp' d10 0 k) (range 20)
199 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]
200 #eval map (λ k -> equiv_prime_pow_decomp' d10 1 k) (range 20)
201 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
202
203 def l10 := [2, 5]
204
205 lemma h110 : l10 = prime_pow_decomp 10 := by
206 rfl
207
208 #eval map (λ k -> (equiv_prime_pow_decomp h110) d10 0 k) (range 20)
209 --[1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0]
210 #eval map (λ k -> (equiv_prime_pow_decomp h110) d10 1 k) (range 20)
211 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0]
212
213 def d60 : Zs[60] := of_nat 1234567
214
215 #eval prime_pow_decomp 60
216 --[4, 3, 5]
217
218 #eval map (λ k -> equiv_prime_pow_decomp' d60 0 k) (range 20)
219 --[3, 1, 0, 2, 2, 1, 1, 3, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
220 #eval map (λ k -> equiv_prime_pow_decomp' d60 1 k) (range 20)
221 --[1, 0, 2, 1, 1, 0, 2, 2, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0]
222 #eval map (λ k -> equiv_prime_pow_decomp' d60 2 k) (range 20)
223 --[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]
224
225 def l100 := [4, 3, 5]
226
227 lemma h160 : l100 = prime_pow_decomp 60 := by
228 rfl
```

Lean Infowiew

padic_integers.lean:4577:0

Messages (1)

padic_integers.lean:4577:0

[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

All Messages (14)

Restart File

Zeile 4577, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:21

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe ↺ ↻ padic_systems

EXPLORER ...

GRUPPE 1

- nat_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic_rationals.lean U
- padic_integers.lean U
- padic_integers.1 U
- padic_integersL. U
- Notes.lean Padic. U

GRUPPE 2

- Lean Infowiew

PADIC SYSTEMS

GLIEDERUNG

ZUTACHSE

int_lemmas.lean U | zmod_lemmas.lean U | padic_rationals.lean U | padic_integers.lean U | padic_integers.1 U | padic_integersL. U | Notes.lean Padic. U

177 section p
183 namespace padic_integer
4579
4580 def 160 := [4, 3, 5]
4581
4582 lemma h160 : 160 = prime_pow_decomp 60 := by
| rfl
4583
4584
4585 ~~eval map (λ k -> (equiv_prime_pow_decomp h160) d60 0 k) (range 20)~~
--[3, 1, 0, 2, 2, 1, 1, 3, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4586 ~~eval map (λ k -> (equiv_prime_pow_decomp h160) d60 1 k) (range 20)~~
--[1, 0, 2, 1, 1, 1, 0, 2, 2, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0]
4587 ~~eval map (λ k -> (equiv_prime_pow_decomp h160) d60 2 k) (range 20)~~
--[2, 3, 2, 1, 0, 0, 4, 0, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0]
4588
4589
4590
4591
4592
4593
4594
4595 /-1
4596 In this section it is shown that Zs[p] is a domain iff p is a proper prime power, Zs[p] has no
4597 zero divisors iff p is a prime power (including the case p = 1) and that a p-adic integer d is a
4598 zero divisor iff
4599
4600 * there is a q dividing p with q ≠ 1 and q and p / q coprime such that the projection of d to Zs[q]
4601 (where Zs[p] ≈+* Zs[q] × Zs[p / q]) is equal to 0, if 1 < p, or
4602
4603 * d k = 0 for some k, if p = 0.
4604
4605 Since Zs[1] is trivial, it doesn't have any zero divisors (not even 0).
4606 /
4607
4608 /--
4609 The inverse of a p-adic integer d if p ≠ 0 and the 0-th digit of d and p are coprime, and the
4610 pointwise sign (in Zs[0] + Z^N) if p = 0.
4611 /
4612 protected def inv : Zs[p] → Zs[p] :=
|ite (p = 0) (λ d k -> sign (valMinAbs (d k))) (λ d k -> @digit_rat p (t(to_nat d k))⁻¹ k)
4613
4614 /--
4615 The inverse of a p-adic integer d if p ≠ 0 and the 0-th digit of d and p are coprime, and the
4616 pointwise sign (in Zs[0] + Z^N) if p = 0.
4617 /
4618 instance : Inv Zs[p] :=
|@padic_integer.inv p
4619
4620 lemma inv_def :
|@padic_integer.inv p =
|ite (p = 0)
| (λ d k -> sign (valMinAbs (d k)) : Zs[p] → Zs[p])

Lean Infowiew

padic_integers.lean:4585:0

Messages (1)

padic_integers.lean:4585:0

[3, 1, 0, 2, 2, 1, 1, 3, 2, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]

All Messages (14)

Restart File

Zeile 4585, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:22

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

padic_systems

EXPLORER ...

GRUPPE 1

- nat_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic_rationals.lean U
- padic_integers.lean U
- padic_integers_1.lean U
- padic_systems.lean U
- Notes.lean Padic... U

GRUPPE 2

- Lean Infoview

PADIC SYSTEMS

GLIEDERUNG

ZUTASCHEN

padicSystems > padic_integers.lean > {} p > {} padic_integer > is_domain_iff_ppp

177 section p
183 namespace padic_integer

5110 *-- Zs[p] is a domain iff p is a proper prime power. --*
5111 theorem is_domain_iff_ppp : IsDomain Zs[p] ↔ ∃ q k, Nat.Prime q ∧ 0 < k ∧ p = q^k := by
5112 constructor
5113 , intro h
5114 apply Classical.by_contradiction
5115 intro hnx
5116 have hp1 : 1 < p := by
5117 apply Classical.by_contradiction
5118 intro hp1
5119 rw [not_lt] at hp1
5120 cases le_iff_eq_or_lt.mp hp1 with
5121 | inl heq =>
5122 rw [heq] at h
5123 cases h with | @mk hIsCancelMulZero hNontrivial =>
5124 cases hIsCancelMulZero with | @mk hIsLeftCancelMulZero hIsRightCancelMulZero =>
5125 cases hNontrivial with | mk hpair =>
5126 cases hIsLeftCancelMulZero with | mk hleft =>
5127 cases hIsRightCancelMulZero with | mk hright =>
5128 rcases hpair with (d, e, hde)
5129 exact hde (Subsingleton.elim d e)
5130 | inr hlt =>
5131 rw [lt_one_iff] at hlt
5132 rw [hlt] at h
5133 replace h := @IsDomain.to_noZeroDivisors _ _ h
5134 cases h with | mk h =>
5135 let a : Zs[0] := (λ k => ite (k = 0) 1 0)
5136 let b : Zs[0] := (λ k => ite (k = 1) 1 0)
5137 have hmul : a * b = 0 := by
5138 rw [mul_def, if_pos rfl, zero_def]
5139 funext k
5140 cases k with
5141 | zero =>
5142 rfl
5143 | succ k =>
5144 cases k with
5145 | zero =>
5146 rfl
5147 | succ k =>
5148 rfl
5149 have ha : a ≠ 0 := by
5150 intro ha
5151 rw [zero_def] at ha
5152 replace ha := Function.funext_iff.mp ha 0
5153 exact one_ne_zero ha
5154 have hb : b ≠ 0 := by
5155 intro hb
5156 rw [zero_def] at hb

Lean Infoview

padic_integers.lean:51100

No info found.

All Messages (14)

Restart File

Zeile 5110, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:24

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe

padic_systems

EXPLORER ...

GRUPPE 1

- nat_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic_rationals.lean U
- padic_integers.lean U
- padic_integers.1, U
- padic_systems.lean U
- Notes.lean Padic... U

GRUPPE 2

- Lean Infoview

PADIC SYSTEMS

GLIEDERUNG

ZUTAUSCHE

padicSystems > padic_integers.lean > {} p > {} padic_integer > zero_divisor_if

177 section p
183 namespace padic_integer
5274 /--
5275 | p-adic integer d is a zero divisor iff
5276
5277 * there is a q dividing p with q ≠ 1 and q and p / q coprime such that the projection of d to Zs[q] (where Zs[p] ≈ Zs[q] × Zs[p / q]) is equal to 0, if 1 < p, or
5278
5279
5280 * d k = 0 for some k, if p = 0.
5281
5282 Since Zs[1] is trivial, it doesn't have any zero divisors (not even 0).
5283 /-
5284 theorem zero_divisor_iff {d : Zs[p]} :
5285 (exists e, e ≠ 0 ∧ d * e = 0) ↔
5286 ite (p = 0) (exists k, d k = 0) (exists q, q ≠ 1 ∧ q ∤ p ∧ coprime q (p / q) ∧
5287 ite (p = 1) (false) (
5288 exists q, q ≠ 1 ∧ q ∤ p ∧ coprime q (p / q) ∧
5289 ite (q : q ≠ 1) (hdvd : q ∣ p) (hcop : coprime q (p / q)),
5290 ((equiv_prod `` hdvd hcop).toFun d).1 = 0) := by
5291 constructor
5292 intro h
5293 split_ifs with hp0 hp1
5294 rcases h with (e, he1, he2)
5295 rw [zero_def] at he2 he1
5296 rw [Function.ne_if] at he1
5297 cases he1 with | intro k hk =>
5298 replace he2 := congr_fun he2 k
5299 rw [mul_def, if_pos hp0] at he2
5300 use k
5301 cases p with
5302 | zero =>
5303 have he2' : valMinAbs (d k) * valMinAbs (e k) = 0 := by
5304 rw [valMinAbs_def_zero, valMinAbs_def_zero, he2]
5305 replace he2' := congr_arg (λ (x : Z) → Int.div x (valMinAbs (e k))) he2'
5306 dsimp only at he2'
5307 rw [Int.mul_div_cancel, valMinAbs_def_zero, valMinAbs_def_zero, Int.zero_div] at he2'
5308 exact he2'
5309 exact hk
5310 | succ p =>
5311 exfalso
5312 exact succ_ne_zero _ hp0
5313 rcases h with (e, he1, _)
5314 apply he1
5315 exact \$Subsingleton.elim _ (subsingleton_iff_peo.mpr hp1) _ _
5316 induction p using nat_lemmas.induction_on_prime_powers with
5317 | h0 =>
5318 | exfalso
5319 | exact hp0 rfl
5320 | hpow p hpow =>

Lean Infoview

padic_integers.lean:5275:0

No info found.

All Messages (14)

Restart File

Zeile 5275, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:26

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe < > padic_systems

EXPLORER ... s.lean U padic_rationals.lean U padic_integers.lean U padic_integers_isomorphism.lean U

GRUPPE 1 nat_lemmas.lean U int_lemmas.lean U zmod_lemmas.lean U padic rationals.lean U padic integers.lean U

X padic integers.lean U padic systems.lean U Notes.lean Padic.lean

GRUPPE 2 Lean Infoview

PADIC SYSTEMS

GLIEDERUNG

ZUTACHSE

padicSystems > padic_integers_isomorphism.lean

1 | -
2 Copyright (c) 2023 Mario Weitzer. All rights reserved.
3 Released under Apache 2.0 license as described in the file LICENSE.
4 Authors: Mario Weitzer
5 -/
6
7 import PadicSystems.padic_integers
8 import Mathlib.NumberTheory.Padics.PadicIntegers
9 import Mathlib.Algebra.Ring.Equiv
10
11 set_option autoImplicit false
12
13 /-!
14 ## Overview
15
16 In this file it is shown that the two implementations of the p-adic integers $\text{Zs}[p]$ and $\mathbb{Z}_{(p)}$ (mathlib) are isomorphic (as commutative rings) if p is prime.
17
18 ## Important definitions
19
20 None.
21
22 ## Notation
23
24 None.
25
26
27 ## Implementation notes
28
29 None.
30
31 ## References
32
33 * <https://en.wikipedia.org/wiki/P-adic_number#p-adic_integers>
34
35 ## Tags
36
37 p-adic, p adic, padic, p-adic integer, ring isomorphism
38 -/
39
40 -----
41
42 open nat_lemmas
43 open int_lemmas
44 open zmod_lemmas
45
46 -----
47
48 section p
49

Lean Infoview padic_integers_isomorphism.lean:1:0
No info found.
All Messages (0)

Restart File

Zeile 1, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:28

File Edit Bearbeiten Auswahl Anzeigen Gehe zu Ausführen Terminal Hilfe < > padic_systems

EXPLORER ... s.jean U padic_rationals.lean U padic_integers.lean U padic_integers_isomorphism.lean U

GEÖFFNETE EDITOREN

GRUPPE 1

- nat_lemmas.lean U
- int_lemmas.lean U
- zmod_lemmas.lean U
- padic rationals.lean U
- padic integers.lean U

X padic_integers.lean U

- padic_systems.lean U
- Notes.lean Padic... U

GRUPPE 2

- Lean Infoview

> PADIC SYSTEMS

> GLIEDERUNG

> ZUTATSCHE

padicSystems > padic_integers_isomorphism.lean

```
48 section p
49   namespace padic_integer
50     constructor <:>
51     .rw [mul_comm]
52     apply
53     .mul_lt_of_lt_of_le_one_of_nonneg _ (@padicNorm.of_int _ (by assumption) _)
54     .(padicNorm.nonneg _)
55     assumption
56
57 lemma to_padic_int_map_mul_aux1 {f g : N → Z} (hf : IsCauSeq (padicNorm p) (λ k => t(f k))) :
58   (hg : IsCauSeq (padicNorm p) (λ k => t(g k))) :
59   PadicInt.ofIntSeq f hf * PadicInt.ofIntSeq g hg =
60   PadicInt.ofIntSeq (λ k => t(f k * g k)) (to_padic_int_map_mul_aux1 hf hg) := by
61   iterate 3 (rw [PadicInt.ofIntSeq])
62   simp_rw [CauSeq.Completion.mk_eq_mk, Int.cast_mul]
63   rfl
64
65 lemma to_padic_int_map_mul_aux2 {f : N → Z} (hf : IsCauSeq (padicNorm p) (λ k => t(f k))) :
66   (IsCauSeq (padicNorm p) (λ k => t(f k % p^(k+1)))) :=
67   to_padic_int_map_add_aux2 hf
68
69 lemma to_padic_int_map_mul_aux2' {f : N → Z} (hf : IsCauSeq (padicNorm p) (λ k => t(f k))) :
70   PadicInt.ofIntSeq f hf =
71   PadicInt.ofIntSeq (λ k => t(f k % p^(k+1))) (to_padic_int_map_add_aux2 hf) :=
72   to_padic_int_map_add_aux2 hf
73
74 lemma to_padic_int_map_mul {x y : Zs[p]} :
75   to_padic_int (x * y) = to_padic_int x * to_padic_int y := by
76   iterate 3 (rw [to_padic_int])
77   rw [to_padic_int_map_mul_aux1 p (by assumption) (λ (k : N) => t(to_nat x k))
78       (λ (k : N) => t(to_nat y k))]
79   dsimp only
80   simp_rw [+Nat.cast_mul]
81   apply Eq.symm
82   rw [to_padic_int_map_mul_aux2]
83   congr
84   funext k
85   funext k
86   rw [←Int.natCast_pow, +Int.natCast_mod,
87       ←to_nat_mul_pnz (Nat.Prime.ne_zero (Fact.elim (by assumption)))]
88
89 |-- Zs[p] and Z_[p] are isomorphic (in terms of mathlib's ring_equiv). -/
90 noncomputable
91 def equiv_padic_int : Zs[p] ≈* Z_[p] :=
92 { toFun := to_padic_int
93   invFun := of_padic_int
94   left_inv := of_padic_int_to_padic_int_inv
95   right_inv := to_padic_int_of_padic_int_inv
96   map_mul' := to_padic_int_map_mul
97   map_add' := to_padic_int_map_add }
```

Lean Infoview

padic_integers_isomorphism.lean:1026:0

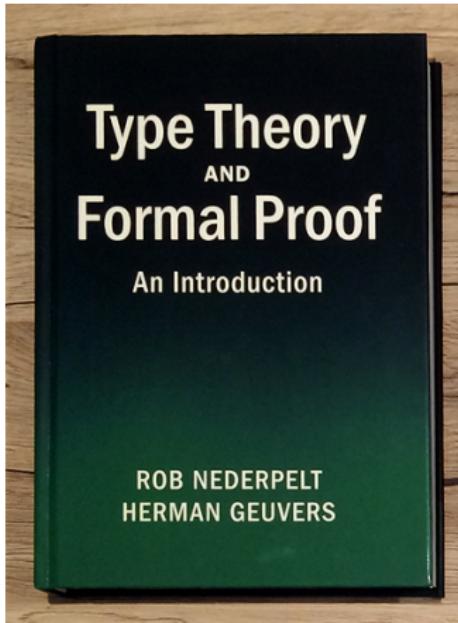
No info found.

All Messages (0)

Restart File

Zeile 1026, Spalte 1 Leerzeichen: 2 UTF-8 CRLF lean4 DEU 14:38

- 1) Optional but highly recommended: read



(do all the exercises!)

2) Optional: read my lecture notes

<https://marioweitzer.com/sites/default/files/Type%20Theory.pdf>

(ignore first page, only makes sense together with other material)

2) Optional: read my lecture notes

<https://marioweitzer.com/sites/default/files/Type%20Theory.pdf>

(ignore first page, only makes sense together with other material)

Overlaps with the book but fills in some gaps:

2) Optional: read my lecture notes

<https://marioweitzer.com/sites/default/files/Type%20Theory.pdf>

(ignore first page, only makes sense together with other material)

Overlaps with the book but fills in some gaps:

- Proof of the Church-Rosser theorem (confluence of the λ -calculus)

2) Optional: read my lecture notes

<https://marioweitzer.com/sites/default/files/Type%20Theory.pdf>

(ignore first page, only makes sense together with other material)

Overlaps with the book but fills in some gaps:

- Proof of the Church-Rosser theorem (confluence of the λ -calculus)
- Proof of Turing completeness of the λ -calculus

2) Optional: read my lecture notes

<https://marioweitzer.com/sites/default/files/Type%20Theory.pdf>

(ignore first page, only makes sense together with other material)

Overlaps with the book but fills in some gaps:

- Proof of the Church-Rosser theorem (confluence of the λ -calculus)
- Proof of Turing completeness of the λ -calculus
- Solutions to selected examples from the book

2) Optional: read my lecture notes

<https://marioweitzer.com/sites/default/files/Type%20Theory.pdf>

(ignore first page, only makes sense together with other material)

Overlaps with the book but fills in some gaps:

- Proof of the Church-Rosser theorem (confluence of the λ -calculus)
- Proof of Turing completeness of the λ -calculus
- Solutions to selected examples from the book

3) Install Lean (version 4) on Windows, Linux or MacOS

https://leanprover-community.github.io/get_started.html

and create your first project depending on Mathlib

<https://leanprover-community.github.io/install/project.html>

4) Read

https://leanprover.github.io/theorem_proving_in_lean4

(do all the exercises!)

4) Read

https://leanprover.github.io/theorem_proving_in_lean4

(do all the exercises!)

Find more material at

<https://leanprover-community.github.io/papers.html>

4) Read

https://leanprover.github.io/theorem_proving_in_lean4

(do all the exercises!)

Find more material at

<https://leanprover-community.github.io/papers.html>

Ask questions at

<https://leanprover.zulipchat.com>

(very active and helpful community of Lean developers and enthusiasts)

Thank you!