

Alphabetic List of Functions

Standard Dictionary for Path Semantics

by Sven Nilsen, 2017

A

$\text{abs} := \lambda(a) = \text{if } a < 0 \{ -a \} \text{ else } \{ a \}$

$\text{add} := \lambda(a, b) = a + b$

When written $\lambda a : [+ b] c$ it means λa plus λb is equal to λc .

$\text{and} := \lambda(a : \text{bool}, b : \text{bool}) = a \wedge b$

In C-like programming languages this is equivalent to $\lambda a \ \&\& \ b$.

When written $\lambda a : (\wedge b)$ it means both λa and λb are λtrue , or neither are.

$\text{acos} : \text{real} \rightarrow \text{real}$

The trigonometric inverse cosine function.

$\text{asin} : \text{real} \rightarrow \text{real}$

The trigonometric inverse sinus function.

$\text{atan} : \text{real} \rightarrow \text{real}$

The trigonometric inverse tangent function.

$\text{atan}_2 : \text{real} \times \text{real} \rightarrow \text{real}$

The trigonometric inverse tangent function with 2 arguments.

Returns the angle of a vector in radians $\lambda \text{atan}_2(y, x)$.

C

$\text{cardinality} : \text{set} \rightarrow \text{nat} \mid \aleph^{\mathbb{N}}$

Returns the cardinality of a set.

$\text{cardinality}(\text{nat}) = \aleph^0$

$\text{cardinality}(\text{real}) = \aleph^1$

$\text{concat} : \text{list} \times \text{list} \rightarrow \text{list}$

Appends the second list to the first list, returning a new list.

$\text{construct}_a := \lambda() = a$

Constructs an object.

$\text{cos} : \text{real} \rightarrow \text{real}$

The trigonometric cosine function.

$\text{cross} : \text{vector} \times \text{vector} \rightarrow \text{vector}$

$\text{cross}\{[\text{vec_dim}] \ 3, [\text{vec_dim}] \ 3\}$

Returns the cross product between two vectors.

This is defined only for vectors in 3 dimensions.

D

$\text{dec} := \backslash(a) = a - 1$

$\text{dedup} : \text{list} \rightarrow \text{list}$

Removes duplicates from list, returning a new list.

$\text{det} : \text{matrix} \rightarrow \text{real}$

Returns the determinant of a matrix.

$\text{diag} := \backslash(m : \text{matrix}) = \forall i, j \{ \text{if } i == j \{ \text{continue} \} \text{ else } \{ m[i][j] == 0 \} \}$

Returns `true` if matrix is a diagonal matrix.

$\text{dim} : \text{matrix} \rightarrow (\text{nat}, \text{nat})$

Returns the dimensions of the matrix `(rows, columns)`.

$\text{div} := \backslash(a, b) = a / b$

When written `a : [/ b] c` it means `a` divided by `b` is equal to `c`.

$\text{dot} : \text{vector} \times \text{vector} \rightarrow \text{real}$

Returns the dot product between two vectors.

$\text{dup} : \backslash(a) = (a, a)$

E

$\text{el} : \text{nat} \times \text{nat} \times \text{matrix} \rightarrow \text{any}$

Returns element of matrix at row and column index.

Notice that this is row major, such that `y` becomes before `x`.

$\text{even} := \backslash(a : \text{nat}) = (a \% 2) == 0$

$\text{even} <=> \text{linear}(0, 2)$

Returns `true` if a number is even.

$\text{eq} := \backslash(a, b) = a == b$

$\text{exc} := \backslash(a : \text{bool}, b : \text{bool}) = a \wedge \neg b$

In C-like programming languages this is equivalent to `a && !b`.

$\text{exclude} : \text{set} \times \text{set} \rightarrow \text{set}$

Excludes elements from the second set from the first set.

$\text{exp}_{\mathbb{A}} := \backslash(a) = e^a$

Returns the natural exponent of a number.

$\text{exp}_{\mathbb{R}} : \text{real} \rightarrow \text{real}$

$\text{exp}_{\mathbb{C}} := \backslash(a : \text{complex}) = \cos(\text{re}(a)) + \mathbf{i} \cdot \sin(\text{im}(a))$

F

`factorize : nat → list`

Returns a sorted list of prime factors of natural number.

`factorial := \ (x : nat) = $\prod_{i \in [0, x+1)} \{ i \}$`

`falseN := \ (⏟, ⏟, ...) = false`

A function that always returns `false`.

`false0 := \ () = false`

`false1 := \ (⏟) = false`

`fract := \ (a : real) = a % 1`

`fst := \ ((a, b)) = a`

Returns the first element in a tuple.

G

$\text{ge} := \lambda(a, b) = a \geq b$

When written $\lambda a : (>= b)$ it means λa is greater than or equal to λb .

$\text{gt} := \lambda(a, b) = a > b$

When written $\lambda a : (> b)$ it means λa is greater than λb .

I

$\text{id}_A := \lambda(x : A) = x$

$\text{if} := A \times A \rightarrow (\text{bool} \rightarrow A)$

A higher order function used to construct boolean functions.

$\text{inc} := \lambda(a) = a + 1$

$\text{intersect} : \text{set} \times \text{set} \rightarrow \text{set}$

Returns a new set containing elements belonging to both sets.

$\text{inv} : \lambda(a) = 1 / a$

$\text{invert} \Leftrightarrow \text{mat_inv}$

$\text{im} : \text{complex} \rightarrow \text{real}$

Returns the imaginary part of a complex number.

J

$\text{join} \Leftrightarrow \text{add}$

Used to reason about circuit diagrams.

$\text{len} : \text{list} \rightarrow \text{nat}$

L

$\text{le} := \lambda(a, b) = a \leq b$

When written $\lambda a : (<= b)$ it means λa is less than or equal to λb .

$\text{linear} := \lambda(a : \text{nat}, b : \text{nat} \wedge (> 0)) = \lambda(x) = \text{if } x < a \{ \text{false} \} \text{ else } \{ ((x - a) \% b) == 0 \}$

Returns `true` if a natural number is in a linear sequence of natural numbers.

$\text{ln} : \text{real} \rightarrow \text{real}$

Returns the natural logarithm of a number.

$\text{lt} := \lambda(a, b) = a < b$

When written $\lambda a : (< b)$ it means λa is less than λb .

M

$\text{mat_add} : \text{matrix} \times \text{matrix} \rightarrow \text{matrix}$

Matrix addition.

$\text{mat_id} : \text{nat} \rightarrow \text{matrix}$

Constructs an identity matrix.

$\text{mat_inv} : \text{matrix} \rightarrow \text{matrix}$

Returns the inverse matrix.

$\text{mat_mul} : \text{matrix} \times \text{matrix} \rightarrow \text{matrix}$

Matrix multiplication, row major.

$\text{max} := \lambda(a : \text{list}) = \max i \{ a[i] \}$

$\text{max}_2 := \lambda(a, b) = \text{if } a > b \{ a \} \text{ else } \{ b \}$

$\text{min} := \lambda(a : \text{list}) = \min i \{ a[i] \}$

$\text{min}_2 := \lambda(a, b) = \text{if } a < b \{ a \} \text{ else } \{ b \}$

$\text{mul}_A := \lambda(a : A, b : A) = a \cdot b$

When written $\lambda a : [.. b] c$ it means λa multiplied with b is equal to c .

$\text{mul}_{\mathbb{N}} : \text{nat} \times \text{nat} \rightarrow \text{nat}$

$\text{mul}_{\mathbb{Q}} : \text{rational} \times \text{rational} \rightarrow \text{rational}$

$\text{mul}_{\mathbb{R}} : \text{real} \times \text{real} \rightarrow \text{real}$

$\text{mul}_{\mathbb{C}} : \text{complex} \times \text{complex} \rightarrow \text{complex}$

N

$\text{nand} := \lambda(a : \text{bool}, b : \text{bool}) = \text{not}(\text{and}(a, b))$

$\text{neg} := \lambda(a) = \neg a$

$\text{neq} \iff \text{xor}$

$\text{nexc} := \lambda(a : \text{bool}, b : \text{bool}) = \text{not}(\text{exc}(a, b))$

$\text{nor} := \lambda(a : \text{bool}, b : \text{bool}) = \text{not}(\text{or}(a, b))$

$\text{not} := \lambda(a : \text{bool}) = \neg a$

In C-like programming languages this is written $!a$.

$\text{nrex} := \lambda(a : \text{bool}, b : \text{bool}) = \text{not}(\text{rex}(a, b))$

$\text{nxor} \iff \text{eq}$

O

$\text{odd} := \lambda(a : \text{nat}) = (a \% 2) == 1$

$\text{odd} \iff \text{linear}(1, 2)$

Returns `true` if a number is odd.

$\text{or} := \lambda(a : \text{bool}, b : \text{bool}) = a \vee b$

In C-like programming languages this is equivalent to $a \parallel b$.

When written $\lambda a : (\vee b)$ it means λa or λb are `true`.

P

$\text{pair} := \backslash(a) = \backslash(b) = (a, b)$

$\text{prime} : \text{nat} \rightarrow \text{bool}$

Returns `true` if natural number is a prime number.

$\text{pop} : \text{list} \rightarrow (\text{list}, \text{any})$

Removes an item from a list, returning a new list and the item removed.

$\text{pow}_A : A \times A \rightarrow A$

Returns the power of a number.

When written `a : [^ b] c` it means `a` powered by `b` is equal to `c`.

$\text{pow}_{\mathbb{N}} : \text{nat} \times \text{nat} \rightarrow \text{nat}$

$\text{pow}_{\mathbb{Q}} : \text{rational} \times \text{rational} \rightarrow \text{rational}$

$\text{pow}_{\mathbb{R}} : \text{real} \times \text{real} \rightarrow \text{real}$

$\text{pow}_{\mathbb{C}} : \text{complex} \times \text{complex} \rightarrow \text{complex}$

$\text{prod} := \backslash(a : \text{list}) = \prod i \{ a[i] \}$

$\text{push} : \text{list} \times \text{any} \rightarrow \text{list}$

Pushes an item to the end of a list

R

$\text{re} := \text{complex} \rightarrow \text{real}$

Returns the real part of a complex number.

$\text{rem} := \backslash(a, b) = a \% b$

Also called “modulus binary operator”.

This is the rest value you get after integer division.

When written `a : [% b] c` it means `a` modulus `b` is equal to `c`.

$\text{rexc} := \backslash(a : \text{bool}, b : \text{bool}) = b \wedge \neg a$

In C-like programming languages this is equivalent to `b && !a`.

S

$sc := \lambda(sc, f) = \lambda(n) = f(sc(sc, f), n)$
 $sc(sc) : ((A \rightarrow B) \times A \rightarrow B) \rightarrow (A \rightarrow B)$
A convenient fixed point combinator that allows anonymous recursive calls, using the first parameter as a `self` function.
Here is an example of generating the numbers in the Fibonacci sequence:
 $fib := \lambda(self : nat \rightarrow nat, n : nat) = \text{if } n == 0 \{ 0 \} \text{ else if } n == 1 \{ 1 \} \text{ else } \{ self(n-1) + self(n-2) \}$
 $call_fib := sc(sc, fib)$
 $call_fib(20) \quad \quad \quad // 6765$
 $sequence := \lambda(a : nat, b : nat \wedge (> 0)) = \lambda(x) = a + b \cdot x$
Maps from natural numbers to a linear sequence of natural numbers.
 $\sin : real \rightarrow real$
The trigonometric sinus function.
 $snd := \lambda((a, b)) = b$
Returns the second element of a tuple.
 $sort_f := list \rightarrow list$
Sorts a list by function `f`.
When `f` is not specified, default ascending order is used.
 $sorted_f := list \rightarrow bool$
Returns `true` if list is sorted by function `f`.
When `f` is not specified, default ascending order is used.
 $split := \lambda(s : real) = \lambda(x : real) = (s \cdot x, (1 - s) \cdot x)$
Used to reason about circuit diagrams.
 $square_len := \lambda(a : vector) = \sum i \{ a[i] \cdot a[i] \}$
 $\sqrt{}_A : A \rightarrow A$
Takes the square root of a number.
 $\sqrt{}_{\mathbb{N}} : nat \rightarrow nat$
Defined only for square numbers.
 $\sqrt{}_{\mathbb{R}} : real \rightarrow real$
Defined only for non-negative numbers.
 $\sqrt{}_{\mathbb{C}} : complex \rightarrow complex$
Automatic conversion from real to complex number.
 $strict_subset : set \times set \rightarrow bool$
Returns `true` if all elements of the first set belongs to the second set, and the two sets do not have equal cardinality.
When written `a : (\subset b)` it means `a` is a strict subset of `b`.
 $sub := \lambda(a, b) = a - b$
 $subset : set \times set \rightarrow bool$
Returns `true` if all elements of the first set belongs to the second set.
When written `a : (\subseteq b)` it means `a` is a subset of `b`.
 $sum := \lambda(a : list) = \sum i \{ a[i] \}$
 $swap := \lambda((a, b)) = (b, a)$

T

$\tan : real \rightarrow real$

The trigonometric tangent function.

$\text{trace} := \lambda(m : \text{matrix}) = \sum i, i \{ m[i][i] \}$

$\text{transpose} : \text{matrix} \rightarrow \text{matrix}$

Returns the transposed matrix, where rows are swapped with columns.

$\text{true}_N := \lambda(_, _, \dots) = \text{true}$

A function that always returns `true`.

$\text{true}_0 := \lambda() = \text{true}$

$\text{false}_1 := \lambda() = \text{false}$

U

$\text{union} : \text{set} \times \text{set} \rightarrow \text{set}$

Returns the union of two sets.

When written `a : [∪ b] c` it means `a` union `b` results in `c`.

$\text{unit} : \text{any} \rightarrow ()$

Used to erase information about an input argument.

V

$\text{vec_dim} : \text{vector} \rightarrow \text{nat}$

Returns the number of dimensions of a vector.

X

$x : \text{vector} \rightarrow \text{real}$

Returns the x-component of a vector.

$\text{xor} := \lambda(a : \text{bool}, b : \text{bool}) = a \wedge \neg b \vee \neg a \wedge b$

In C-like programming languages this is equivalent to “a && !b || !a && b”.

When written `a : (∨ b)` it means either `a` or `b` is `true`, but not both.

Y

$y : \text{vector} \rightarrow \text{real}$

Returns the y-component of a vector.

Z

$z : \text{vector} \rightarrow \text{real}$

Returns the z-component of a vector.

W

$w : \text{vector} \rightarrow \text{real}$

Returns the w-component of a vector.