# Alphabetic List of Functions

## Standard Dictionary for Path Semantics

by Sven Nilsen, 2017

## A

add := \(a, b) = a + b

*When written `a : [+ b] c` it means `a` plus `b` is equal to `c`.*

and := \(a : bool, b : bool) = a ∧ b

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

*When written `a : (∧ b)` it means both `a` and `b` are `true`, or neither are.*

acos : real → real

*The trigonometric inverse cosine function.*

asin : real → real

*The trigonometric inverse sinus function.*

atan : real → real

*The trigonometric inverse tangent function.*

atan2 : real × real → real

*The trigonometric inverse tangent function with 2 arguments.*

*Returns the angle of a vector in radians `atan2(y, x)`.*

## C

cardinality : set → nat | $\mathfrak{N}^N$

*Returns the cardinality of a set.*

cardinality(nat) = $\mathfrak{N}^0$

cardinality(real) = $\mathfrak{N}^1$

concat : list × list → list

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

construct$_a$ : () = a

*Constructs an object.*

cos : real → real

*The trigonometric cosine function.*

cross : vector × vector → vector

*Returns the cross product between two vectors.*

## D

dec := \(a) = a – 1

dedup : list → list

*Removes duplicates from list, returning a new list.*

det : matrix → real

*Returns the determinant of a matrix.*

div := \(a, b) = a / b

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

dot : vector × vector → real

*Returns the dot product between two vectors.*

## E

even := \(a : nat) = (a % 2) == 0

even <=> linear(0, 2)

*Returns `true` if a number is even.*

eq := \(a, b) = a == b

exc := \(a : bool, b : bool) = a ∧ ¬b

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

exclude : set × set → set

*Excludes elements from the second set from the first set.*

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

*Returns the natural exponent of a number.*

Exp$_{\mathbb{R}}$ : real → real

exp$_{\mathbb{C}}$ := \(a : complex) = cos(re(a)) + **i** · sin(im(a))

## F

false$_{N}$ := \(_, _, …) = false

*A function that always returns `false`.*

false$_{0}$ := \() = false

false$_{1}$ := \(_) = false

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

*Returns the first element in a tuple.*

## G

ge := \(a, b) = a >= b

*When written `a : (>= b)` it means `a` is greater than or equal to `b`.*

gt := \(a, b) = a > b

*When written `a : (> b)` it means `a` is greater than `b`.*

# I

id := \(x) = x

if := A × A → (bool → A)

*A higher order function used to construct boolean functions.*

inc := \(a) = a + 1

intersect : set × set → set

*Returns a new set containing elements belonging to both sets.*

im : complex → real

*Returns the imaginary part of a complex number.*

# J

join <=> add

*Used to reason about circuit diagrams.*

len : list → nat

# L

le := \(a, b) = a <= b

*When written `a : (<= b)` it means `a` is less than or equal to `b`.*

linear := \(a : nat, b : nat ∧ (> 0)) = \(x) = if x < a { false } else { ((x – a) % b) == 0 }

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

ln : real → real

*Returns the natural logarithm of a number.*

lt := \(a, b) = a < b

*When written `a : (< b)` it means `a` is less than `b`.*

# M

mat_add : matrix × matrix → matrix

*Matrix addition.*

mat_dim : matrix → (nat, nat)

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

mat_mul : matrix × matrix → matrix

*Matrix multiplication, row major.*

mat_trace : matrix → vector

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

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

mul := \(a, b) = a · b

*When written `a : [· b] c` it means `a` multiplied with `b` is equal to `c`.*

# N

nand := \(a : bool, b : bool) = not(and(a, b))
neq <=> xor
nexc := \(a : bool, b : bool) = not(exc(a, b))
nor := \(a : bool, b : bool) = not(or(a, b))
not := \(a : bool) = ¬a
*In C-like programming languages this is written `!a`.*
nrexc := \(a : bool, b : bool) = not(rexc(a, b))
nxor <=> eq

# O

odd := \(a : nat) = (a % 2) == 1
odd <=> linear(1, 2)
*Returns `true` if a number is odd.*
or := \(a : bool, b : bool) = a ∨ b
*In C-like programming languages this is equivalent to `a || b`.*
*When written `a : (∨ b)` it means `a` or `b` are `true`.*

# P

prime : nat → bool
*Returns `true` if natural number is a prime number.*
pop : list → (list, any)
*Removes an item from a list, returning a new list and the item removed.*
pow$_{A}$ : A × A → A
*Returns the power of a number.*
*When written `a : [^ b] c` it means `a` powered by `b` is equal to `c`.*
pow$_{\mathbb{N}}$ : nat × nat → nat
pow$_{\mathbb{R}}$ : real × real → real
pow$_{\mathbb{C}}$ : complex × complex → complex
prod := \(a : list) = ∏ i { a[i] }
push : list × any → list
Pushes an item to the end of a list

# R

re := complex → real
*Returns the real part of a complex number.*
rem := \(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`.*
rexc := \(a : bool, b : bool) = b ∧ ¬a
*In C-like programming languages this is equivalent to `b && !a`.*

## S

sequence := \(a : nat, b : nat ∧ (> 0)) = \(x) = a + b · x
*Maps from natural numbers to a linear sequence of natural numbers.*

sin : real → real
*The trigonometric sinus function.*

snd := \((a, b)) = b
*Returns the second element of a tuple.*

$\text{sort}_f$ := list → list
*Sorts a list by function `f`.*
*When `f` is not specified, default ascending order is used.*

split := \(s : real) = \(x : real) = (s · x, (1 – s) · x)
*Used to reason about circuit diagrams.*

square_len := \(a : vector) = ∑ i { a[i] · a[i] }

$\text{sqrt}_A$ : A → A
*Takes the square root of a number.*
$\text{sqrt}_{\mathbb{N}}$ : nat → nat
*Defined only for square numbers.*
$\text{sqrt}_{\mathbb{R}}$ : real → real
*Defined only for non-negative numbers.*
$\text{sqrt}_{\mathbb{C}}$ : complex → complex
*Automatic conversion from real to complex number.*

strict_subset : set × set → 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 : (⊂ b)` it means `a` is a strict subset of `b`.*

sub := \(a, b) = a – b

subset : set × set → bool
*Returns `true` if all elements of the first set belongs to the second set.*
*When written `a : (⊆ b)` it means `a` is a subset of `b`.*

sum := \(a : list) = ∑ i { a[i] }

## T

tan : real → real
*The trigonometric tangent function.*

$\text{true}_N$ := \(_, _, …) = true
*A function that always returns `true`.*
$\text{true}_0$ := \() = true
$\text{false}_1$ := \() = false

# U

union : set × set → set

*Returns the union of two sets.*
*When written `a : [∪ b] c` it means `a` union `b` results in `c`.*

unit : any → ()

*Used to erase information about an input argument.*

# V

vec_dim : vector → nat

*Returns the number of dimensions of a vector.*

# X

x : vector → real

*Returns the x-component of a vector.*

xor := \(a : bool, b : bool) = a ∧ ¬b ∨ ¬a ∧ 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 : vector → real

*Returns the y-component of a vector.*

# Z

z : vector → real

*Returns the z-component of a vector.*

# W

w : vector → real

*Returns the w-component of a vector.*