Alphabetic List of Paths

Standard Dictionary for Path Semantics

by Sven Nilsen, 2017

A

add[(= 0)] <=> and
add[(¬= 0)] <=> or
$add_{\mathbb{C},\mathbb{Q},\mathbb{R},\mathbb{Z}}[neg_{\mathbb{C},\mathbb{Q},\mathbb{R},\mathbb{Z}}]$ <=> $add_{\mathbb{C},\mathbb{Q},\mathbb{R},\mathbb{Z}}$
$add_{\mathbb{N}}[even]$ <=> eq
$add_{\mathbb{N}}[odd]$ <=> xor
$add_{\mathbb{R}}[swap \rightarrow id]$ <=> $add_{\mathbb{R}}$
$add_{\mathbb{R}}[exp]$ <=> $mul_{\mathbb{R}}$
and[not] <=> or

C

concat[len] <=> add
concat[sum] <=> add
concat[min] <=> min_2
concat[max] <=> max_2
cross{[vec_dim] 3, [vec_dim] 3}[unit × unit → vec_dim] <=> $construct_3$

D

dec{(¬= 0)}[even] <=> not
$div_{\mathbb{R}}${(¬= 0), (¬= 0)}[swap → id] <=> inv · div

E

eq[not] <=> xor
exc[not] <=> nrexc

I

id[id] <=> id
$id_A[id \to f]$ <=> f $f : A \to B$
$id_A[f]$ <=> id_B $f : A \to B$
$id_A[f \to id_A]$ <=> f^{-1} $f : A \to B$
inc[even] <=> not
inc[inc] <=> inc

M

mat_id[id → trace] <=> id
mat_inv{[det] (¬= 0)}[id → mat_inv] <=> id
mat_mul[det] <=> mul
mat_mul[fst · dim ⨯ snd · dim → dim] <=> id
$mul_\mathbb{N}[(= 0)]$ <=> or
$mul_\mathbb{N}[(¬= 0)]$ <=> and
$mul_\mathbb{N}\{(>= 0), (>= 0)\}[(>= 0)]$ <=> $true_1$
$mul_\mathbb{N}\{(¬= 1), (¬= 1)\}[prime]$ <=> $false_1$
$mul_\mathbb{N}[(\% k: (¬= 0))]$ <=> $(\% k) · mul_\mathbb{N}$
$mul_\mathbb{N}[even]$ <=> or
$mul_\mathbb{N}[odd]$ <=> and
mull $n_R[neg → id]$ <=> $mul_\mathbb{N}$
$mul_\mathbb{R}[ln]$ <=> $add_\mathbb{R}$
mul[swap → id] <=> mul

N

nand[not] <=> nor
nexc[not] <=> rexc
nor[not] <=> nand
not[not] <=> not
nrexc[not] <=> exc

O

or[not] <=> and
not[not] <=> not

P

push[len × unit → len] <=> inc
push[sum × id → sum] <=> add
push[max × id → max] <=> max_2
push[min × id → min] <=> min_2
pop{[len] (¬= 0)}[len → len · fst] <=> dec
pop{[len] (¬= 0)}[(sum, snd · pop) → sum · fst] <=> sub
pop{[len] (¬= 0)}[(sum, fst · pop) → sum · snd] <=> sub

R

rexc[not] <=> nexc

S

$sort_f$[unit → $sorted_f$] <=> $true_1$
split(_)[id → join] <=> id
$sub_\mathbb{R}$[swap → id] <=> neg

T

transpose[el(i, j) → el(j, i)] <=> id
transpose[dim] <=> swap

U

unit[unit] <=> unit

X

xor[not] <=> eq