FuN to Haskell conversion

description	**FuN**	**Haskell**	**Mathematics**
function type	$f : A \to B$	`f :: a->b`	$f : A \to B$
function application	$f \cdot x$	`f x`	$f\,(x)$
function composition	$f \circ g$	`f.g`	$f \circ g$
lambda expression	f whr $f \cdot x = E$ end	`\x ->` E	$(\,\lambda x . E\,)$
identity function	I	`id`	$(\,\lambda x . x\,)$
constant a function	$K \cdot a$	`const a`	$(\,\lambda x . a\,)$
empty tuple	$\langle\,\rangle$	`()`	—
one-element tuple (singleton)	$\langle x \rangle$	—	—
two-element tuple (pair)	$\langle x\,, y \rangle$	`(x,y)`	$(x\,, y)$
three-element tuple (triple)	$\langle x\,, y\,, z \rangle$	`(x,y,z)`	$(x\,, y\,, z)$
pair: left element selection	$\langle x\,, y \rangle \cdot 0 \;=\; x$	`fst (x,y) = x`	—
pair: right element selection	$\langle x\,, y \rangle \cdot 1 \;=\; y$	`snd (x,y) = y`	—
tuple: any element selection	$\mathit{tup} \cdot 0\,,\; \mathit{tup} \cdot 1\,,\; \cdots$	—	—
size of a tuple	$\#\,\mathit{tup}$	—	—
finite lists of length n	$\mathcal{L}_n(A)$	—	—
finite lists of any length	$\mathcal{L}_*(A)$	`[a]`	—
infinite lists (streams)	$\mathcal{L}_\infty(A)$	—	—
length of finite list s	$\#s$	`length s`	—
element i of (in)finite list s	$s \cdot i$	`s !! i`	—
cons	$a \triangleright s$	`a : s`	—
snoc	$s \triangleleft a$	—	—
concatenation	$s +\!\!+\, t$	`s ++ t`	—
take	$s \lceil n$	`take n s`	—
drop	$s \lfloor n$	`drop n s`	—
map	$f \bullet s$	`map f s`	$f\,(s)$
where-clause	E whr *decls* end	`let` *decls* `in` E	—