Comprehensions

Dun Dun

Daaaaaaaaaaaa

Comprehensions in Practice

Python: [i for i in range(20) if i%2 == 0]

Haskell: [(i,j) | i <- [1,2], j <- [1..4]]

Ceylon: {for (o in orgs) for (e in o.emps) e.name}

Scala: for (i <- range(from, to) if i % 2 == 0) yield i

C#: from element in array orderby element descending where element > 2 select element

SQL: SELECT MAX(TEMP_F), MIN(TEMP_F), AVG(RAIN_I), ID FROM STATS GROUP BY ID

for comprehensions

for (v in list) expr(v)

list : List(T) List(T')

v : T

expr : T → T'

for compilation

list.map(λv. expr(v))

if comprehensions

for (x in xs) if (x>5) x+1

xs.filter(λx. x>5).map(λx. x+1)

xs.map(λx. if (x>5) [x+1] else []).flatten()

nested for comprehensions

for (x in 1..10) for (y in 1..10) (x,y)

for (x in 1..10) 1..10.map(λy. (x,y)) [concat]

1..10.map(λx.).flatten()

How to make composable/consistent?

Alternatively if (cond) expr ⇒ if (cond) Some(expr) else None

for(var in list) expr ⇒ list.map(λvar. expr)

for (x in xs) if (x>3) x+1 ⇒ xs.map(λx. if(x>3) Some(x+1) else None) : List<option<ℤ>> ⇓ filterOption List<ℤ>

Judgements

context ↓ expression ↓ type

$\Gamma \vdash e : \tau$ (in context Γ, e is an expression of type τ)

$\Gamma \vdash c : \tau \,|\, e$ (in context Γ, c is a computation resulting in es of τs)

Derivation Rules

$\Gamma \vdash l : List(\tau) \rightsquigarrow L : [\![\Gamma]\!] \to List([\![\tau]\!])$

$\Gamma, v{:}\tau \vdash c : \tau' \,|\, e \rightsquigarrow f : [\![\Gamma]\!] \times [\![\tau]\!] \to \Sigma e \in \mathbb{N}.([\![\tau']\!](e))$

$list, \tau \to \tau' \rightsquigarrow t : \forall n.\, List([\![\tau]\!](n)) \to [\![\tau']\!](n)$

$\Gamma \vdash for(v \text{ in } l)\ c : \tau' \,|\, \tau' \rightsquigarrow \lambda G.\, t(L(G).map(\lambda x. f(G, x))) : [\![\Gamma]\!] \to [\![\tau']\!]([\![\tau]\!])$

$List, list \to list \rightsquigarrow flatten$

if Derivation Rule

$\Gamma \vdash cond : \mathbb{B}$

$\Gamma \vdash c : \tau \,|\, e$

$option, \tau \to \tau'$

$\Gamma \vdash if(cond)\ c : \tau \,|\, \tau'$

$option, option \to option$

$list, option \to list$

$option, list \to list$

Boring Rule

$\Gamma \vdash e : \tau \rightsquigarrow f : [\![\Gamma]\!] \to [\![\tau]\!]$

$\Gamma \vdash e : \tau \,|\, Il \rightsquigarrow f : [\![\Gamma]\!] \to [\![Il]\!]([\![\tau]\!])$

$Il ; c \to c \rightsquigarrow id$

$c ; Il \to c \rightsquigarrow id$

Example

$\vdots$

$x : \mathbb{Z} \vdash x+1 : \mathbb{Z}$

$\vdots$ $\quad$ $\vdots$

$x : \mathbb{Z} \vdash x > 3 : \mathbb{B}$ $\quad$ $x : \mathbb{Z} \vdash x+1 : \mathbb{Z} \,|\, option$ $\quad$ $option, \tau \to option$

$xs : List(\mathbb{Z}) \vdash xs : List(\mathbb{Z})$ $\quad$ $\vdots$ $\quad$ $List, option \to List$

$xs : List(\mathbb{Z}) \vdash for(x \text{ in } xs)\ if(x>3)\ x+1 : \mathbb{Z} \,|\, List$

Semantics

$[\![\Gamma \vdash e : \tau]\!] : [\![\Gamma]\!] \to [\![\tau]\!]$

$[\![\Gamma \vdash e : \tau / \varepsilon]\!] : [\![\Gamma]\!] \to [\![\varepsilon]\!]([\![\tau]\!])$

$[\![\varepsilon, \varepsilon' \to \varepsilon'']\!] : \forall \tau.\ [\![\varepsilon]\!]([\![\varepsilon']\!](\tau)) \to [\![\varepsilon'']\!](\tau)$