COMP0020

Imperative Style. vs. functional style

λ calculus elegant.

Miranda Tuple. ("Ted", 23, False)
:: ([Char], num, bool)
("increment", inc) :: ([char], num → num).

lazy evaluation. → g(x y) = x.
g(1, 1/0)
↓
never evaluated.

partial functions.
f(x y) = x/y. y≠0.

polymorphic functions don't evaluate some of the input.

three :: * → num.
three x → 3

for different types, use *, **, ***

Non-exhaustive patterns

f True = False.
x = f False ✗ False not defined.

Patterns destroy laziness.
not lazy. f. (x, 0) = x.

notlazy f (x, y) = x.

notlazy (5, 1/0)

↳ evaluated to cmp to 0

Duplicate param = equality test.

bothequal (*, *) → Bool.

(x, x) = T

(x, y) = F

Recursive function creates loops.

f :: num → [char]

f 0 = ""

f n = "x" ++ (f (n-1)), if n > 0

= "x" ++ (f (n+1)), otherwise.

Type Synonyms

str == [char].

Coord == (num, num, num)

Lists.

xl :: [num].

xl = x : [lst] (lst, x)

or xl = [lst, x].

Iterative list.

[1..] = [1, 2, 3 ___].

list comprehension → [n * n | n ∈ [1, 2, 3]]

= [1, 4, 9]

λ calculus

α. reduction. $\lambda x.E \to \lambda y.E[y/x]$

all free occurences $x \to y$.

β reduction $(\lambda x.E)M = E[M/x]$

η reduction $\lambda x.(E\ x) \to E$.

δ rules + , × (3+4) = 7.

Examples.

$\lambda x.(x+3)\ 5. \to$ β reduction

$(5+3) \to$ δ rules

8.

$\lambda y.((\lambda x.(x+y))\ 5))\ 3 \to$

$\lambda y.(5+y))\ 3. \to$

$5+3 \to$

8

$(\lambda x.((x\ 5) + (x\ 4)))\ \ (\lambda x.(x+3)) \rightarrow$

$((\lambda x(x+3)\ 5) + ((\lambda x(x+3)\ 4))$.

$(5+3) + (4+3)$.

$= 15$.

Mark Scan GC.

Triggerd at malloc(). when free is low.

Program pause at GC.

Mark Gargabage. Trace all live pointers

Scan. Garbage node available for re-use

We can compact to reduce. fragment. sometimes

every block has a "mark" bit