

At the expense of building large closures in heap

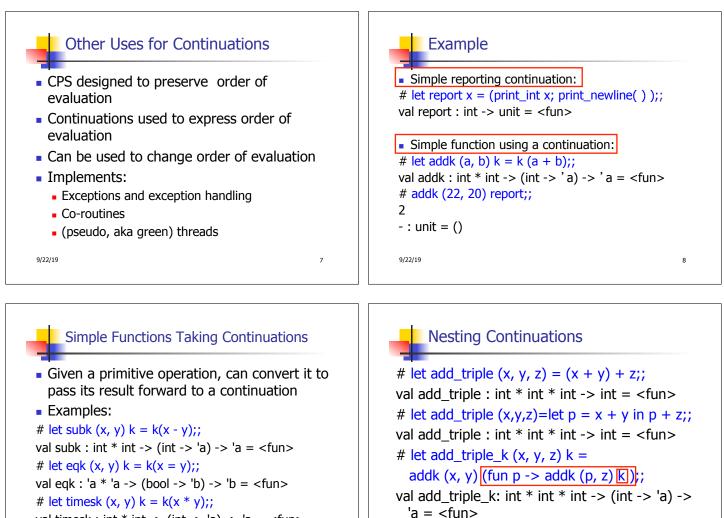
6

9/22/19

5

9/22/19

functional code



9/22/19

val timesk : int * int -> (int -> 'a) -> 'a = <fun>

9/22/19

add_three: a different order

- # let add_triple (x, y, z) = x + (y + z);;
 How do we write add_triple_k to use a
- different order?
- let add_triple_k (x, y, z) k =

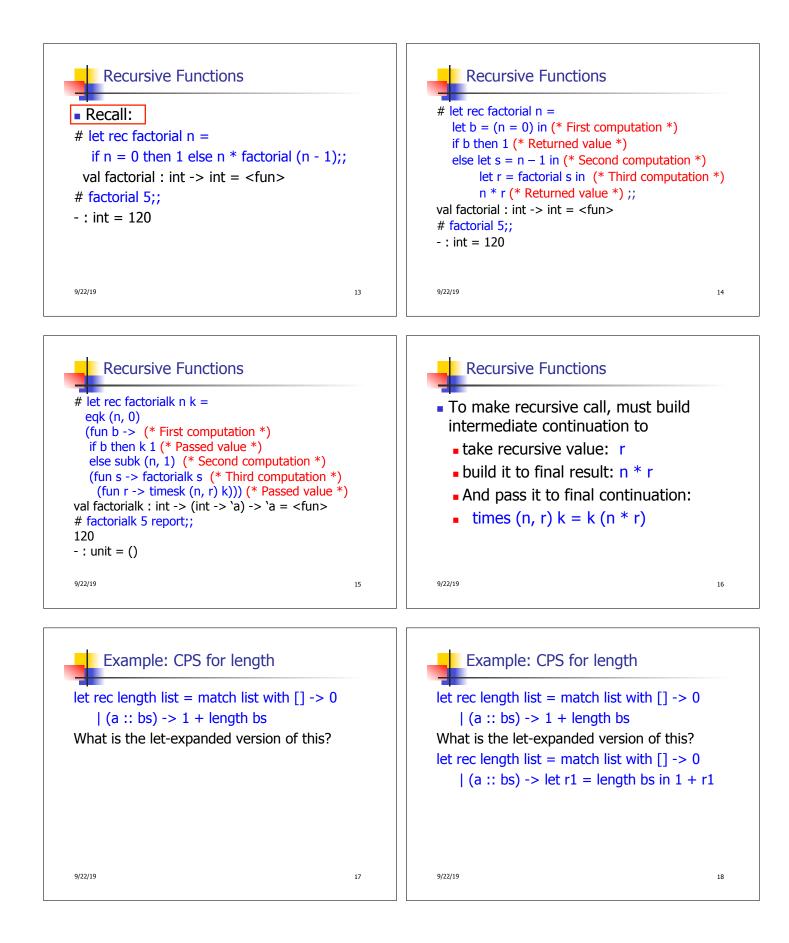


11

9

add_three: a different order # let add_triple (x, y, z) = x + (y + z);; How do we write add_triple_k to use a different order? let add_triple_k (x, y, z) k = addk (y,z) (fun r -> addk(x,r) k)

10





#let rec allk (pk, l) k =

9/22/19

23

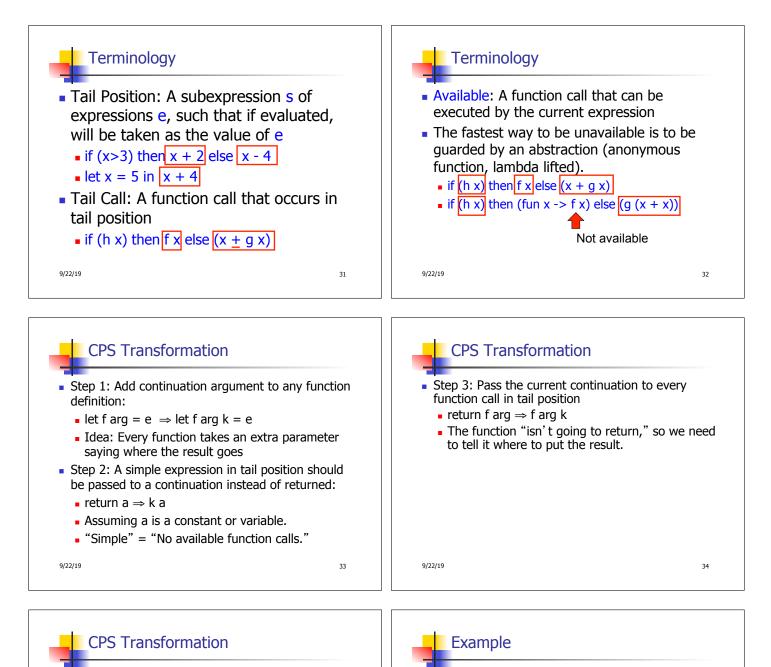
9/22/19

#let rec allk (pk, l) k = match l with [] ->

true

24





- Step 4: Each function call not in tail position needs to be converted to take a new continuation (containing the old continuation as appropriate)
 - return op (f arg) \Rightarrow f arg (fun r -> k(op r))
 - op represents a primitive operation
 - return $f(g arg) \Rightarrow g arg (fun r-> f r k)$

9/22/19

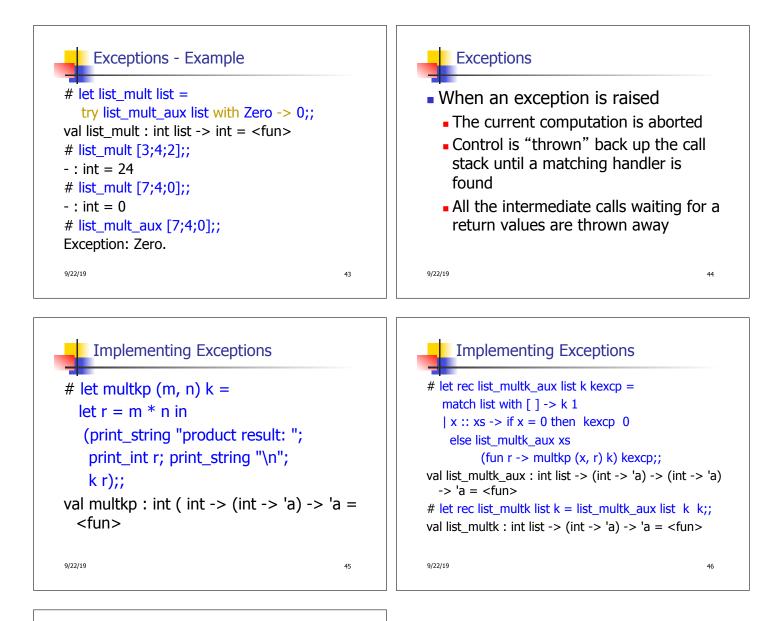
35

After: **Before:** let rec add listk lst k = let rec add list lst = (* rule 1 *) match lst with match lst with []->0 |[] -> k 0 (* rule 2 *) | 0 ::: xs -> add_list xs | 0 ::: xs -> add_listk xs k (* rule 3 *) | x :: xs -> (+) x | x :: xs -> add_listk xs (add_list xs);; (fun r -> k ((+) x r));; (* rule 4 *)

36

9/22/19





47

Implementing Exceptions

```
# list_multk [3;4;2] report;;
product result: 2
product result: 8
product result: 24
24
- : unit = ()
# list_multk [7;4;0] report;;
0
- : unit = ()

9/22/19
```