

Continuations **Continuation Passing Style** Idea: Use functions to represent the control Writing procedures such that all flow of a program procedure calls take a continuation to Method: Each procedure takes a function as which to give (pass) the result, and an extra argument to which to pass its return no result, is called continuation result; outer procedure "returns" no result passing style (CPS) Function receiving the result called a continuation Continuation acts as "accumulator" for work still to be done 9/18/17 9/18/17 7 8 **Continuation Passing Style** Why CPS? A compilation technique to implement non- Makes order of evaluation explicitly clear local control flow, especially useful in Allocates variables (to become registers) for each step of computation interpreters. Essentially converts functional programs into imperative ones A formalization of non-local control flow in Major step for compiling to assembly or byte denotational semantics code Tail recursion easily identified Possible intermediate state in compiling Strict forward recursion converted to tail recursion functional code • At the expense of building large closures in heap

9/18/17

Other Uses for Continuations

- CPS designed to preserve order of evaluation
- Continuations used to express order of evaluation
- Can be used to change order of evaluation
- Implements:
 - Exceptions and exception handling
 - Co-routines
 - (pseudo, aka green) threads

9/18/17

11

9

Example

9/18/17

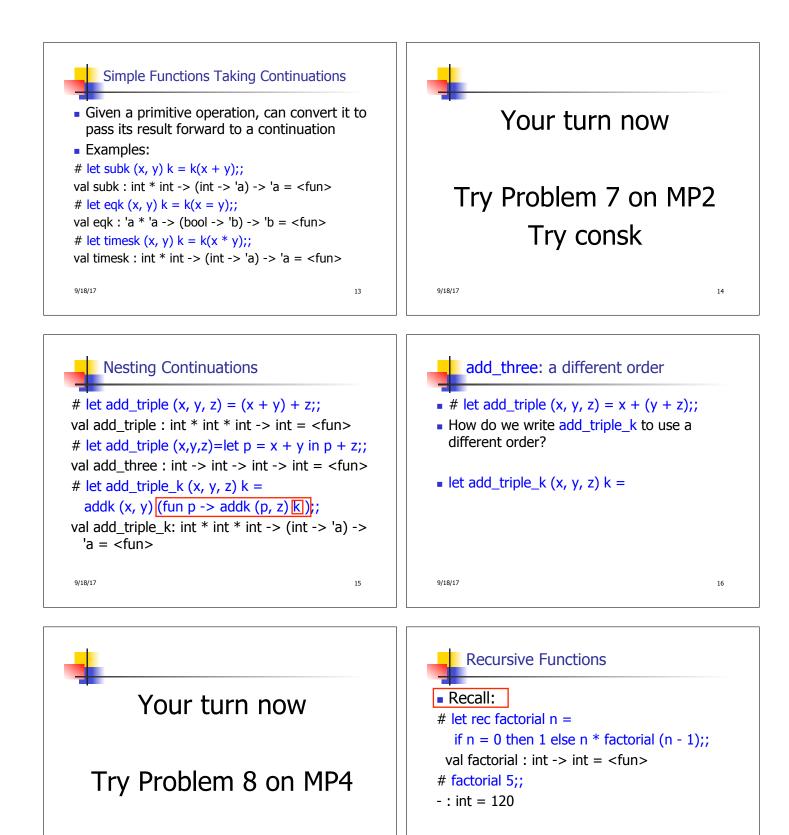
Simple reporting continuation: # let report x = (print_int x; print_newline());; val report : int -> unit = <fun>

10

12

```
Simple function using a continuation:
# let addk (a, b) k = k (a + b);;
val addk : int * int -> (int -> 'a) -> 'a = <fun>
# addk (22, 20) report;;
2
- : unit = ()
```

```
9/18/17
```

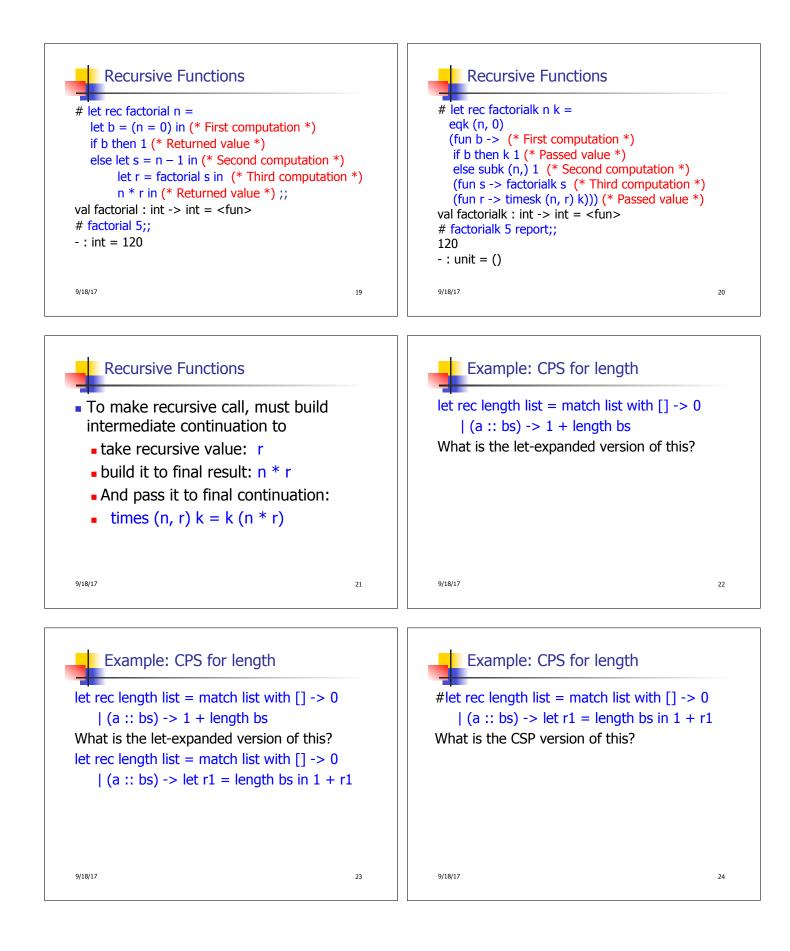


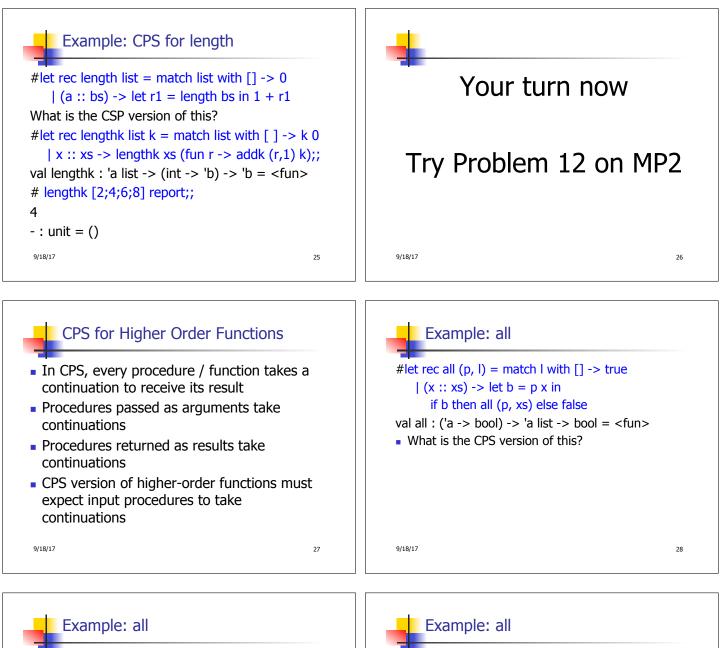
17

9/18/17

9/18/17

18





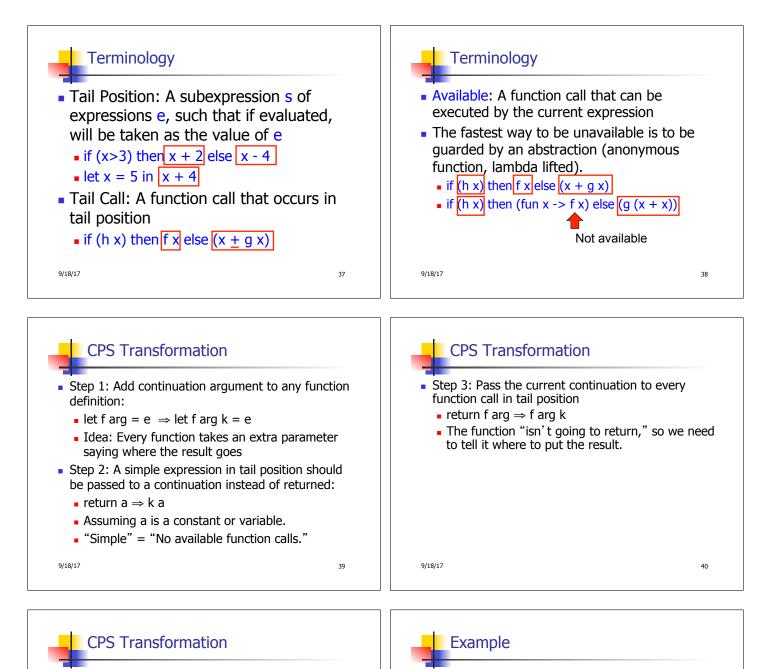
#let rec all (p, l) = match l with [] -> true
 | (x :: xs) -> let b = p x in
 if b then all (p, xs) else false
val all : ('a -> bool) -> 'a list -> bool = <fun>
 What is the CPS version of this?
#let rec allk (pk, l) k =

9/18/17

29

Example: all
let rec all (p, l) = match l with [] -> true
| (x :: xs) -> let b = p x in
if b then all (p, xs) else false
val all : ('a -> bool) -> 'a list -> bool = <fun>
What is the CPS version of this?
let rec allk (pk, l) k = match l with [] -> true





- 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 $f(g arg) \Rightarrow g arg (fun r-> f r k)$

9/18/17

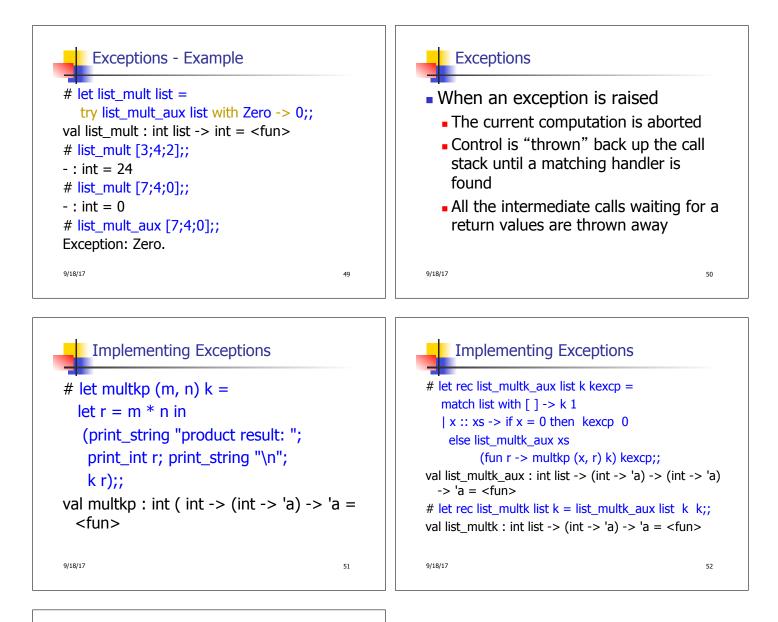
41

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 *)

9/18/17

42





53

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 = ()

9/18/17
```