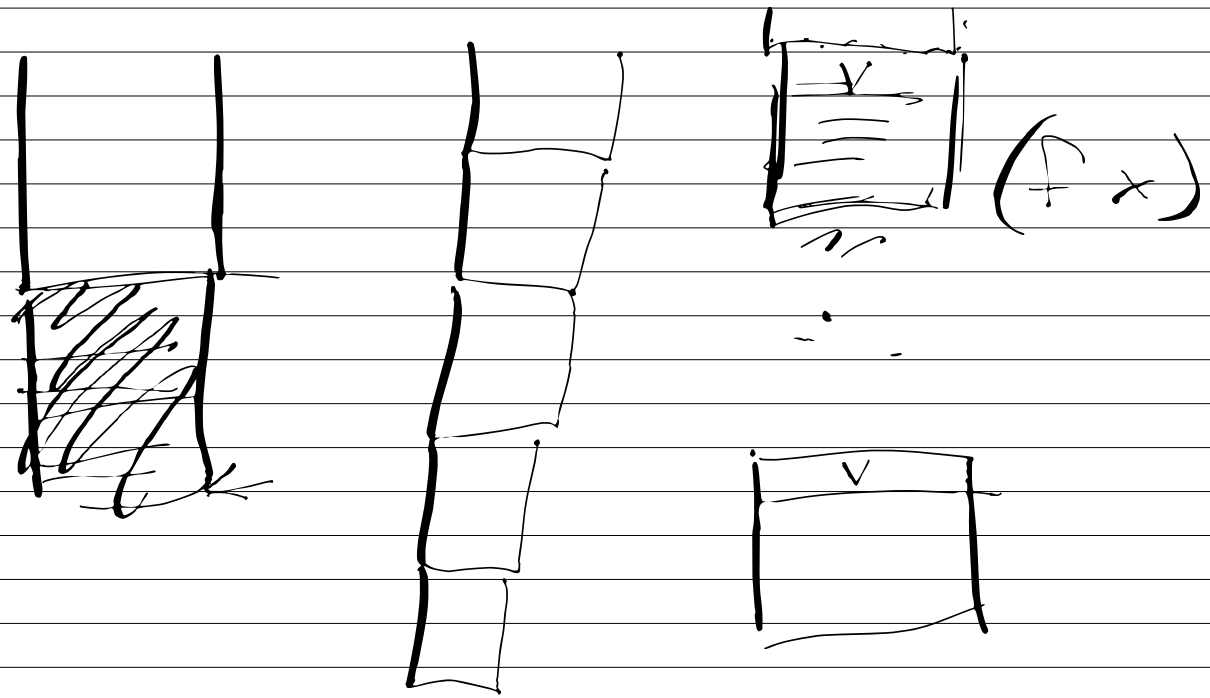
Continuation-passing style "CPS"

1 simple or serious?

2 serious, in tail pos?

"tail call"

→ Properly implemented tail call optimization



DCL Digital Command Language

Continuation?

[(foo x y) (foo (valof x env)
(valof y env))]

(f
(g
(h (+ 5

-1

(abort -1)

(error "sadness" y) "sadness"

```
(Δ !
 (λ (n)
  (if (zero? n)
      1
      (* n (! (sub1 n))))))
```

```
(Δ !
 (λ (n k)
  (if (zero? n)
   (k 1)
   (! (sub1 n)
    (λ (v)
     (k (* n v)))))))
```

1 every λ gets an additional argument, k

2 simple expr in tail pos, apply k

3 serious calls in non-tail pos.
move into tail pos. & build k

```
(Δ fib
 (λ(n)
  (if (< n 2)
      n
   (+ (fib (sub1 n))
      (fib (sub1 (sub1 n)))))))
```

```
(Δ fib-cps
  (λ(n k)
    (if (< n 2)
        (k n)
        (fib-cps (sub1 n)
          (λ(v)
            (fib-cps (sub1 (sub1 n))
              (λ(w)
                (k (+ v w)))))))))
```

```
(define takef-cps
  (λ (f-cps ls k)
    (cond
      ((empty? ls) (k '()))
      (else (f-cps (car ls)
                   (λ (b)
                     (cond
                       (b (takef-cps f-cps (cdr ls)
                                     (λ (v)
                                       (k (cons (car ls) v)))))
                       (else
                         (takef-cps f-cps (cdr ls) k)))))))))
```

```
(define takef
  (λ (f ls)
    (cond
      ((empty? ls) '())
      ((f (car ls))
       (cons (car ls) (takef f (cdr ls))))
      (else (takef f (cdr ls))))))
```

What do we get for our trouble?

tricky bits

let

match

Cond

vs.

nested calls.

```
(Δ c-+                    >((c+ 5) 4)
  (λ(m)                    9
    (λ(n)
      (+ m n))))
```

```
(Δ c+-cps
  (λ(m k)
    (k (λ(n k)
         (k (+ m n))))))
```

```
(c+-cps 5 (λ(c-cps) (c-cps 4 (empty-k)
                             )))
```