

4.23

Alyssa's:

```

(def (analyze-sequence exps)
  (def (execute-sequence procs env)
    (cond ((null? (cdr procs))
           ((car procs) env))
          (else
           ((car procs) env)
           (execute-sequence (cdr procs) env))))
  (let ((procs (map analyze exps)))
    (if (null? procs)
        (error "Empty seq: ANALYZE")
        (λ (env)
          (execute-sequence procs env)))))

```

Text's:

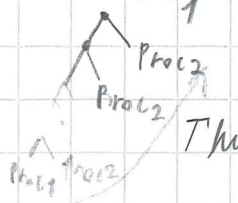
```

(def (analyze-sequence exps)
  (def (sequentially proc1 proc2)
    (λ (env) (proc1 env) (proc2 env)))
  (def (loop first-proc rest-procs)
    (if (null? rest-procs)
        first-proc
        (loop (sequentially first-proc (car rest-procs))
              (cdr rest-procs))))
  (let ((procs (map analyze exps)))
    (if (null? procs) (error "Empty seq: ANALYZE")
        (loop (car procs) (cdr procs)))))

```

The text version "builds" a sequence of sequences to directly execute some runtime. whereas Alyssa's defers the realization for runtime.

↑
through execute-sequence



Thus lots of runtime ^{cond} clause overhead.

For sequences of one, the text version simply removes the sequencing overhead, whereas Alyssa's keeps the cond overhead.