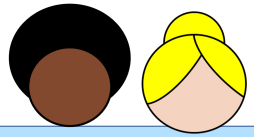
Why in the World is Learning Continuation Passing Style (CPS) So Hard?

Stephen Payne

PROBLEM

- Students learn Continuation passing-style (CPS) in CSSE 304 Programming Language Concepts
- Notoriously difficult concept every year
- **GOAL: Determine what makes CPS so difficult for novices to learn, and whether CPS-tailored feedback helps to reach a correct solution more efficiently**



CPS makes NO SENSE!

QUALITIES OF CPS CODE

Example CPS Code

```
(define member-cps (lambda (item ls k)) 1
    (cond
        [(null? ls) (apply-k k #f)] 2
        [(eq? (car ls) item) (apply-k k #t)]
    3   [else (member-cps item (cdr ls) k)])))
```

1. All substantial procedures (as defined by the problem) take a continuation as an argument
   - **Define continuation:** whatever work is left to do after evaluating expression
   - Ex: If given `(+ 4 (* 6 3))`, then `(+ 4 value-of-mult)` is then **continuation** of `(* 6 3)`
2. Continuation is applied once result is calculated via procedure `apply-k` (CPS code is typically recursive)
3. ALL calls to substantial procedures are in **tail-position**
   - **Define *tail-position:*** *the result of the caller function reduces to that of the callee*

METHODOLOGY

- Create a tool with **tail-position feedback** regarding **name of non-tail, substantial procedure calls**

IN-TAIL FORM? **NO!**

Your code contains these substantial procedure calls in NON-tail position:
- `factorial-cps`
- `K`

- Feedback provided every time students run code
- Students will use the feedback to **adjust their code** so it's in proper CPS form and thus receive **higher *final scores***
- Feedback should **reduce *time* needed to solve problem** as we predicted functionality and form were correlated

Tool Available at: https://rose-easy-cps.web.app/

CLASS EXERCISE

- Subjects asked to **transform** 4 non-CPS procedures to CPS form
- Students split into 2 groups, **feedback** & **non-feedback**
- **ONLY Feedback** students saw tail-position feedback when running code
- All students saw feedback when formally submitting code

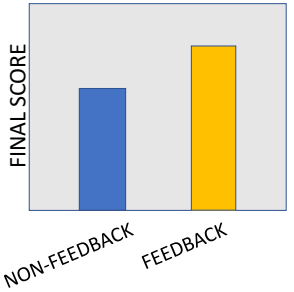
CODE SUBMITTED: GREAT JOB?

Your code contained no NON-tail position substantial procedure calls.

PROBLEM SET

`Factorial-CPS`	`Member?-CPS`
`Set?-CPS`	`Insert-Correctly-CPS`

RESULTS



- Students in given **feedback** had **higher *final scores*** than those in **non-feedback** group
- Students in given **feedback** took **less *time*** than those in **non-feedback** group
- Average ***score*** effect size (*d = 0.315*) means **feedback was moderately helpful,** roughly as much **as an average teacher** (*via Hattie's Ranking*)
- Feedback had negligible effect on ***time*** (*d = 0.222*)

COMMON ERRORS

- *Attempting to use a CPS procedure's return value outside of its continuation*
  - *Obvious indicator is continuation* `(Lambda (v) v)`

Example

```
(if (member?-cps (car ls) (cdr ls)
        (make-k  (lambda (v) v)))
```

- ***Applying continuations*** *(via apply-k)to non-primitive expressions*
  - *Obvious indicator: using return value of* `apply-k` *in caller function*

Example

```
(* n (apply-k (- n 1)))
```