

KCFAE Grammar

```
<KCFAE> ::= <num>
          | {+ <KCFAE> <KCFAE>}
          | {- <KCFAE> <KCFAE>}
          | <id>
          | {fun {<id>} <KCFAE>}
          | {<KCFAE> <KCFAE>}
          | {if0 <KCFAE> <KCFAE> <KCFAE>}
          | {withcc <id> <KCFAE>}
```

NEW

```
{withcc k {+ 1 {k 2}}}
```

\Rightarrow 2

```
{withcc done
```

```
  {{withcc esc
```

```
    {done {+ 1 {withcc k
```

```
      {esc k}}}}}
```

```
  3}}
```

\Rightarrow 4

KCFAE Values

```
(define-type KCFAE-Value
  [numV (n number)]
  [closureV (param symbol?)
            (body KCFAE?)
            (ds DefrdSub?)]
  [contV (proc procedure?)])
```

Implementing withcc

```
; interp : KCFAE DefrdSub -> KCFAE-Value
(define (interp a-fae ds)
  (type-case KCFAE a-fae
    ...
    [withcc (id body-expr)
      (let/cc k
        (interp body-expr
          (aSub id
            (contV k)
            ds))))]))
```

This will work, but it's too meta-circular to tell us anything

Implementing Continuations

```
; interp : KCFAE DefrdSub (KCFAE-Value -> alpha) -> alpha
(define (interp a-fae ds k)
  ...
  ...)
```

Implementing Continuations

```
; interp : KCFAE DefrdSub (KCFAE-Value -> alpha) -> alpha
(define (interp a-fae ds k)
  ...
  [num (n) (k (numV n))]
  ...)
```

Implementing Continuations

```
; interp : KCFAE DefrdSub (KCFAE-Value -> alpha) -> alpha
(define (interp a-fae ds k)
  ...
  [add (l r)
       (interp l ds
               (lambda (v1)
                 (interp r ds
                         (lambda (v2)
                           (k (num+ v1 v2)))))))]
  ...)
```

Implementing Continuations

```
; interp : KCFAE DefrdSub (KCFAE-Value -> alpha) -> alpha
(define (interp a-fae ds k)
  ...
  [sub (l r)
    (interp l ds
      (lambda (v1)
        (interp r ds
          (lambda (v2)
            (k (num- v1 v2))))))]
  ...)
```

Implementing Continuations

```
; interp : KCFAE DefrdSub (KCFAE-Value -> alpha) -> alpha
(define (interp a-fae ds k)
  ...
  [id (name) (k (lookup name ds))]
  ...)
```

Implementing Continuations

```
; interp : KCFAE DefrdSub (KCFAE-Value -> alpha) -> alpha
(define (interp a-fae ds k)
  ...
  [fun (param body-expr)
    (k (closureV param body-expr ds))]
  ...)
```

Implementing Continuations

```
; interp : KCFAE DefrdSub (KCFAE-Value -> alpha) -> alpha
(define (interp a-fae ds k)
  ...
  [app (fun-expr arg-expr)
    (interp fun-expr ds
      (lambda (fun-val)
        (interp arg-expr ds
          (lambda (arg-val)
            (type-case KCFAE fun-val
              [closureV (param body-expr ds)
                (interp body-expr
                  (aSub param
                    arg-val
                    ds)
                  k)]
              [contV (k)
                (k arg-val)]
              [else (error ...)])))]))
  ...)
```

Implementing Continuations

```
; interp : KCFAE DefrdSub (KCFAE-Value -> alpha) -> alpha
(define (interp a-fae ds k)
  ...
  [if0 (test-expr then-expr else-expr)
    (interp test-expr ds
      (lambda (v)
        (if (numzero? v)
            (interp then-expr ds k)
            (interp else-expr ds k))))])
  ...)
```

Implementing Continuations

```
; interp : KCFAE DefrdSub (KCFAE-Value -> alpha) -> alpha
(define (interp a-fae ds k)
  ...
  [withcc (id body-expr)
    (interp body-expr
             (aSub id
                  (contV k)
                  ds)
             k)]
  ...)
```