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}}} ⇒ 2
{withcc done
 {{withcc esc
    {done {+ 1 {withcc k
                 {esc k}}}}}
  3}}                        ⇒ 4
```

KCFAE Values

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

Implementing withcc

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

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

Implementing Continuations

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

  ...)
```

Implementing Continuations

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

Implementing Continuations

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

Implementing Continuations

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

Implementing Continuations

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

Implementing Continuations

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

Implementing Continuations

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

Implementing Continuations

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

Implementing Continuations

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