Higher-Order Functions (Part II)
FAE with Deferred Substitution

\[(\text{interp} \ \{\text{with} \ {y \ 10} \ \{\text{fun} \ {x} \ \{+ \ y \ x\}\}\})\]

\[\Rightarrow\]

\[(\text{interp} \ \{\text{fun} \ {x} \ \{+ \ y \ x\}\})\]

\[\Rightarrow\]

\[(\text{interp} \ \{\{\text{fun} \ {y} \ \{\text{fun} \ {x} \ \{+ \ y \ x\}\}\} \ 10\})\]

\[\Rightarrow\]

\[(\text{interp} \ \{\text{fun} \ {x} \ \{+ \ y \ x\}\})\]
FAE with Deferred Substitution

\[
(\text{interp } \{\text{with } \{y \ 10\} \ \{\text{fun } \{x\} \ {+ \ y \ x}\}\} \ \\ \{\text{with } \{y \ 7\} \ y\}\})
\]

Argument expression:
\[
(\text{interp } \{\text{with } \{y \ 7\} \ y\})
\]
\[
\Rightarrow \ y = 7
\]
\[
(\text{interp } y) \Rightarrow 7
\]

Function expression:
\[
(\text{interp } \{\text{with } \{y \ 10\} \ \{\text{fun } \{x\} \ {+ \ y \ x}\}\})
\]
\[
\Rightarrow \ y = 10
\]
\[
(\text{interp } \{\text{fun } \{x\} \ {+ \ y \ x}\}) \Rightarrow ?
\]
FAE Values

A function value needs to keep its deferred substitution

```scheme
(define-type FAE-Value
  [numV (n number?)]
  [closureV (param-name symbol?)
    (body FAE?)
    (ds DefSub?)])

(define-type DefSub
  [mtSub]
  [aSub (name symbol?)
    (value FAE-Value?)
    (rest DefSub?)])

(test (interp
  {with {y 10} {fun {x} [+ y x]}})
  (closureV 'x [+ y x]
    (aSub 'y (num 10) (mtSub)))))
```
Continuing Evaluation

Function: \{ \text{fun} \ \{x\} \ \{ + \ y \ x\} \}

Argument: 7

To apply, interpret the function body with the given argument:

(interp \{ + \ y \ x\} )
FAE Interpreter with Deferred Substitution

; interp : FAE? DefSub? -> FAE-Value?
(define (interp a-fae ds)
  (type-case FAE a-fae
    [num (n) (numV n)]
    [add (l r) (num+ (interp l ds) (interp r ds))]
    [sub (l r) (num- (interp l ds) (interp r ds))]
    [id (name) (lookup name ds)]
    [fun (param-name body)
      (closureV param-name body ds)]
    [app (fun-expr arg-expr)
      (local [(define fun-val
                    (interp fun-expr ds))
             (define arg-val
                    (interp arg-expr ds))]
             (interp (closureV-body fun-val)
                    (aSub (closureV-param-name fun-val)
                          arg-val
                          (closureV-ds fun-val))))]))