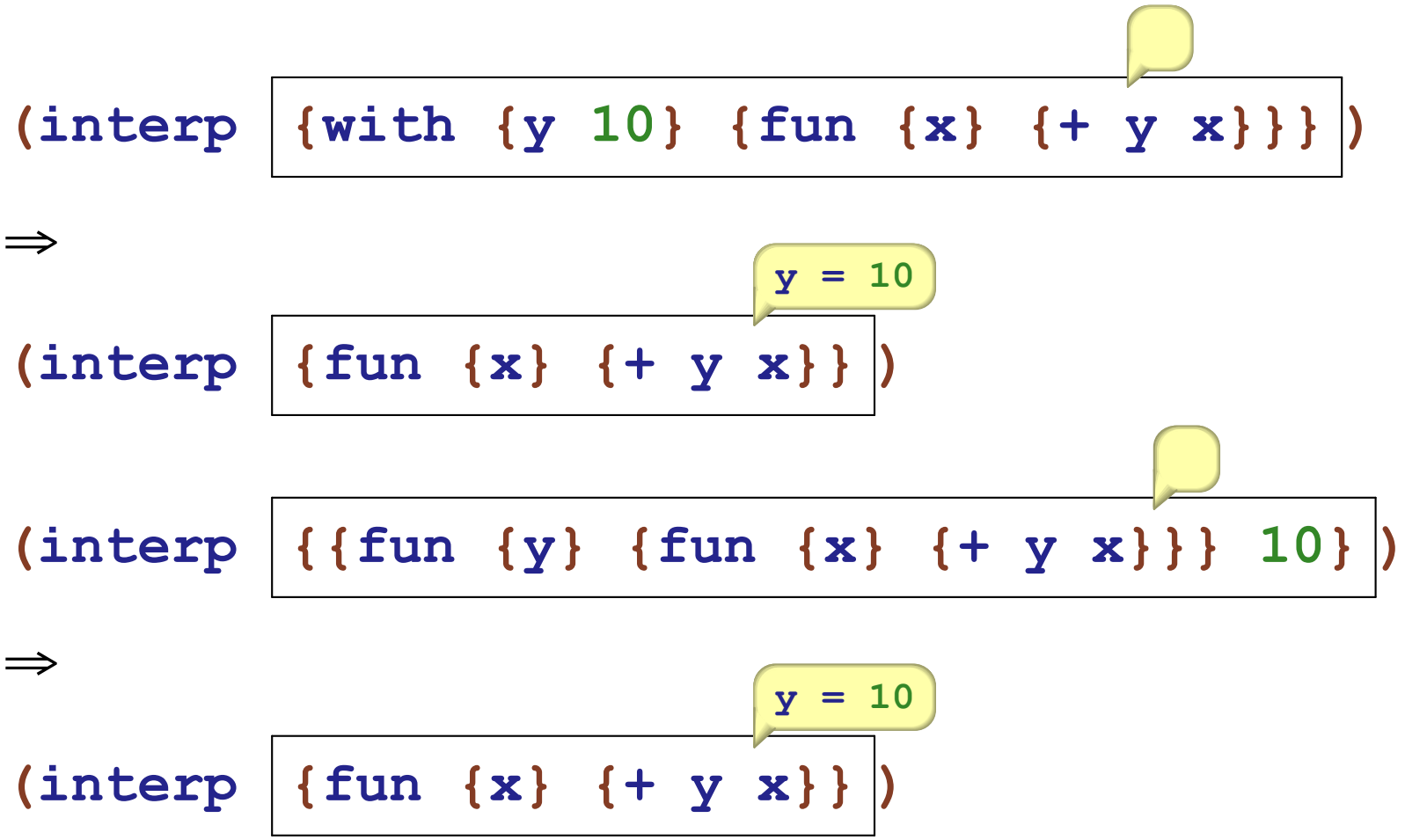
Higher-Order Functions (Part II)

FAE with Deferred Substitution



FAE with Deferred Substitution

```
(interp {{with {y 10} {fun {x} {+ y x}}}
         {with {y 7} y}})
```

Argument expression:

```
(interp {with {y 7} y})
```

⇒

y = 7

```
(interp y) ⇒ 7
```

Function expression:

```
(interp {with {y 10} {fun {x} {+ y x}}})
```

⇒

y = 10

```
(interp {fun {x} {+ y x}}) ⇒ ?
```

FAE Values

A function value needs to keep its deferred substitution

```
(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: `{fun {x} {+ y x}}` (`y = 10`)

Argument: `7`

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

`(interp {+ y x})` (`x = 7`, `y = 10`)

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))))])))
```