

# Cost of Substitution

```
(interp {with {x 1}
          {with {y 2}
            {+ 100 {+ 99 {+ 98 ... {+ y x}}}}}} )
```

⇒

```
(interp {with {y 2}
          {+ 100 {+ 99 {+ 98 ... {+ y 1}}}} } )
```

⇒

```
(interp {+ 100 {+ 99 {+ 98 ... {+ 2 1}}}} )
```

With **n** variables, evaluation will take  $O(n^2)$  time!

# Deferring Substitution

```
(interp {with {x 1}
          {with {y 2}
            {+ 100 {+ 99 {+ 98 ... {+ y x}}}}}} )
```

⇒

```
(interp {with {y 2}
          {+ 100 {+ 99 {+ 98 ... {+ y x}}}} } )
```

⇒

```
(interp {+ 100 {+ 99 {+ 98 ... {+ y x}}}} )
```

⇒ ... ⇒

```
(interp y )
```

# Deferring Substitution with the Same Identifier

`(interp {with {x 1}  
          {with {x 2}  
              x}})`

⇒

`(interp {with {x 2}  
          x})`

⇒

`(interp x)`

Always add to start, then always check from start

# Representing Deferred Substitution

Change

```
; interp : WAE -> num
```

to

```
; interp : WAE DefrdSub -> num
```

```
(define-type DefrdSub  
  [mtSub]  
  [aSub (name symbol?)  
    (value number?)  
    (rest DefrdSub?) ] )
```

# Interp with DefrdSub

```
(interp {with {x 1}
         {with {y 2}
           {+ 100 {+ 99 {+ 98 ... {+ y x}}}}}}
 (mtSub))
```

```
⇒ (interp {with {y 2}
           {+ 100 {+ 99 {+ 98 ... {+ y x}}}}}}
 (aSub 'x 1 (mtSub)))
```

```
⇒ (interp {+ 100 {+ 99 {+ 98 ... {+ y x}}}}
 (aSub 'y 2 (aSub 'x 1 (mtSub))))
```

⇒ ...

```
⇒ (interp y (aSub 'y 2 (aSub 'x 1 (mtSub))))
```

# WAE Interpreter with Deferred Substitutions

```
; interp : WAE DefrdSub -> num  
(define (interp a-wae ds)  
  (type-case WAE a-wae  
    [num (n) n]  
    [add (l r) (+ (interp l ds) (interp r ds))]  
    [sub (l r) (- (interp l ds) (interp r ds))]  
    [with (bound-id named-expr body-expr)  
      ...]  
    [id (name) ...]))
```

# WAE Interpreter with Deferred Substitutions

```
; interp : WAE DefrdSub -> num  
(define (interp a-wae ds)  
  (type-case WAE a-wae  
    [num (n) n]  
    [add (l r) (+ (interp l ds) (interp r ds))]  
    [sub (l r) (- (interp l ds) (interp r ds))]  
    [with (bound-id named-expr body-expr)  
      ...]  
    [id (name) (lookup name ds)]))
```

# WAE Interpreter with Deferred Substitutions

```
; lookup : symbol DefrdSub -> num
(define (lookup name ds)
  (type-case DefrdSub ds
    [mtSub () (error 'lookup "free variable")]
    [aSub (sub-name num rest-ds)
     (if (symbol=? sub-name name)
         num
         (lookup name rest-ds))]))
```



# WAE Interpreter with Deferred Substitutions

```
; interp : WAE DefrdSub -> num  
(define (interp a-wae ds)  
  (type-case WAE a-wae  
    [num (n) n]  
    [add (l r) (+ (interp l ds) (interp r ds))]  
    [sub (l r) (- (interp l ds) (interp r ds))]  
    [with (bound-id named-expr body-expr)  
      ...]  
    [id (name) (lookup name ds)]))
```

# WAE Interpreter with Deferred Substitutions

```
; interp : WAE DefrdSub -> num  
(define (interp a-wae ds)  
  (type-case WAE a-wae  
    [num (n) n]  
    [add (l r) (+ (interp l ds) (interp r ds))]  
    [sub (l r) (- (interp l ds) (interp r ds))]  
    [with (bound-id named-expr body-expr)  
      ... (interp named-expr ds) ...]  
    [id (name) (lookup name ds)]))
```

# WAE Interpreter with Deferred Substitutions

```
; interp : WAE DefrdSub -> num  
(define (interp a-wae ds)  
  (type-case WAE a-wae  
    [num (n) n]  
    [add (l r) (+ (interp l ds) (interp r ds))]  
    [sub (l r) (- (interp l ds) (interp r ds))]  
    [with (bound-id named-expr body-expr)  
      ...  
      (aSub bound-id (interp named-expr ds) ds)  
      ...]  
    [id (name) (lookup name ds)]))
```

# WAE Interpreter with Deferred Substitutions

```
; interp : WAE DefrdSub -> num  
(define (interp a-wae ds)  
  (type-case WAE a-wae  
    [num (n) n]  
    [add (l r) (+ (interp l ds) (interp r ds))]  
    [sub (l r) (- (interp l ds) (interp r ds))]  
    [with (bound-id named-expr body-expr)  
      (interp  
        body-expr  
        (aSub bound-id (interp named-expr ds) ds))]  
    [id (name) (lookup name ds)]))
```

# Function Calls

```
{deffun {f x} {+ 1 x}}
```

```
(interp {with {y 2} {f 10}})
```

⇒

```
(interp {f 10})
```

y = 2

⇒

```
(interp {+ 1 x})
```

...

# Function Calls

```
{deffun {f x} {+ 1 x}}
```

```
(interp {with {y 2}  
          {f 10}})
```

⇒

```
(interp {f 10})
```

y = 2

⇒

```
(interp {+ 1 x})
```

x = 10

Interpreting function body starts with only one substitution

# FlWAE Interpreter with Deferred Substitutions

```
; interp : FlWAE list-of-FunDef DefrdSub -> num
(define (interp a-flwae fundefs ds)
  (type-case FlWAE a-flwae
    ...
    [app (name arg-expr)
         ... ]))
```

# FlWAE Interpreter with Deferred Substitutions

```
; interp : FlWAE list-of-FunDef DefrdSub -> num  
(define (interp a-flwae fundefs ds)  
  (type-case FlWAE a-flwae  
    ...  
    [app (name arg-expr)  
      (local [(define a-fundef  
                (lookup-fundef name fundefs))]  
        (interp (fundef-body a-fundef)  
                fundefs  
                ...  
                (interp arg-expr fundefs ds)  
                ...))])])
```



# FIWAE Interpreter with Deferred Substitutions

```
; interp : FlWAE list-of-FunDef DefrdSub -> num
(define (interp a-flwae fundefs ds)
  (type-case FlWAE a-flwae
    ...
    [app (name arg-expr)
      (local [(define a-fundef
              (lookup-fundef name fundefs))]
        (interp (fundef-body a-fundef)
              fundefs
              (aSub (fundef-arg-name a-fundef)
                    (interp arg-expr fundefs ds)
                    (mtSub))))))]))
```