Cost of Substitution

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

$\Rightarrow$

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

$\Rightarrow$

```
(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}}}}})    x = 1
```

⟹

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

⟹ ... ⟹

```
(interp y)    y = 2    x = 1
```

Deferring Substitution with the Same Identifier

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

$\Rightarrow$

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

$\Rightarrow$

```
(interp x )
```
x = 2   x = 1

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)]))
```
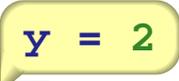
Function Calls

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

`(interp {with {y 2}`
`        {f 10}}  )`
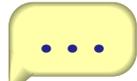
$\Longrightarrow$

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

$\Longrightarrow$

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

Function Calls

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

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

$\Longrightarrow$

               `y = 2`

`(interp {f 10})`

$\Longrightarrow$

               `x = 10`

`(interp {+ 1 x})`

Interpreting function body starts with only one substitution

F1WAE Interpreter with Deferred Substitutions

```
; interp : F1WAE list-of-FunDef DefrdSub -> num
(define (interp a-f1wae fundefs ds)
  (type-case F1WAE a-f1wae
    ...
    [app (name arg-expr)
         ...]))
```

F1WAE Interpreter with Deferred Substitutions

```
; interp : F1WAE list-of-FunDef DefrdSub -> num
(define (interp a-f1wae fundefs ds)
  (type-case F1WAE a-f1wae
    ...
    [app (name arg-expr)
         (local [(define a-fundef
                    (lookup-fundef name fundefs))]
           (interp (fundef-body a-fundef)
                   fundefs
                   ...
                   (interp arg-expr fundefs ds)
                   ...))]))
```

F1WAE Interpreter with Deferred Substitutions

```
; interp : F1WAE list-of-FunDef DefrdSub -> num
(define (interp a-f1wae fundefs ds)
  (type-case F1WAE a-f1wae
    ...
    [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))))]))
```