Deferred Substitutions

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 $O(\mathbf{n}^2)$ time!

Deferring Substitution

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

⇒

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

⇒

```
(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}}           )
```

⟹

```
(interp  {with {x 2}        [x = 1]
            x}              )
```
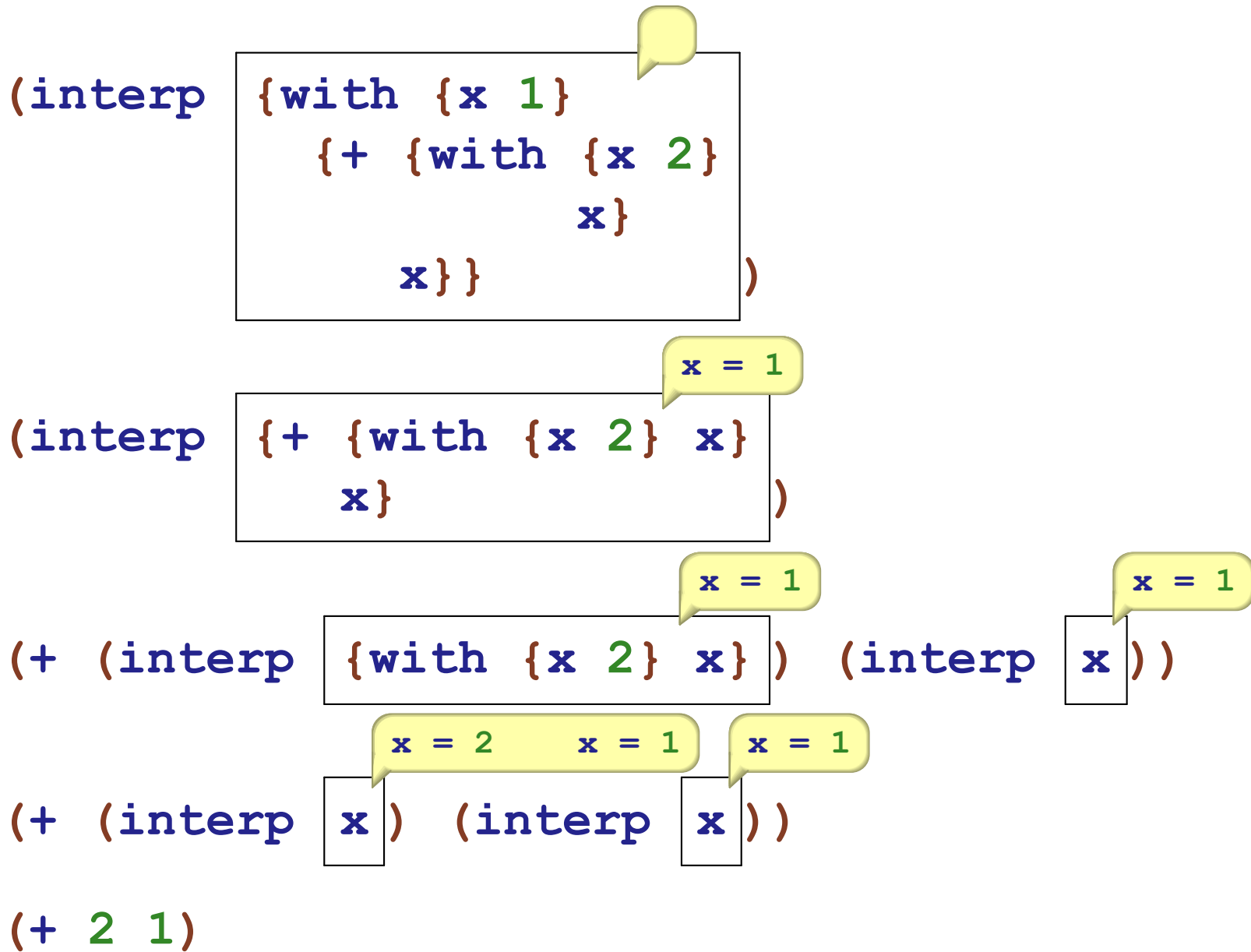
⟹

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

Always add to start, then always check from start

Deferring Substitution with the Same Identifier

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

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

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

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

```
(+ 2 1)
```

Representing Deferred Substitution

Change

```
; interp : WAE? -> number?
```

to

```
; interp : WAE? DefSub? -> number?
```

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

Interp with DefSub

```
(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? DefSub? -> number?
(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 (name named-expr body)
          ...]
    [id (name) ...]))
```

WAE Interpreter with Deferred Substitutions

```
; interp : WAE? DefSub? -> number?
(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 (name named-expr body)
          ...]
    [id (name) (lookup name ds)]))
```

WAE Interpreter with Deferred Substitutions

```
; lookup : symbol? DefSub? -> number?
(define (lookup name ds)
   (type-case DefSub ds
     [mtSub () (error 'lookup "free identifier")]
     [aSub (n num rest)
           (if (symbol=? n name)
               num
                (lookup name rest))])))
```

WAE Interpreter with Deferred Substitutions

```
; interp : WAE? DefSub? -> number?
(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 (name named-expr body)
          ...]
    [id (name) (lookup name ds)]))
```

WAE Interpreter with Deferred Substitutions

```
; interp : WAE? DefSub? -> number?
(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 (name named-expr body)
            ... (interp named-expr ds) ...]
    [id (name) (lookup name ds)]))
```

WAE Interpreter with Deferred Substitutions

```
; interp : WAE? DefSub? -> number?
(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 (name named-expr body)

          ...

          (aSub name (interp named-expr ds) ds)

          ...]
    [id (name) (lookup name ds)]))
```

WAE Interpreter with Deferred Substitutions

```
; interp : WAE? DefSub? -> number?
(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 (name named-expr body)
          (interp
           body
            (aSub name (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}}` `)`

$\Rightarrow$

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

y = 2

$\Rightarrow$

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

x = 10

Interpreting the function body starts with only one substitution

Function Calls

What goes wrong if you extend the old substitution?

```
{deffun {f x} {+ y x}}

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

⇒

(interp  {f 10}  )   y = 2

⇒

(interp  {+ y x}  )   x = 10  y = 2

⇒ 12 wrong!
```

Function Calls

What goes wrong if you extend the old substitution?

```
{deffun {f x} {+ y x}}

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

$\Rightarrow$

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

$\Rightarrow$

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

$\Rightarrow$ free identifier (as it should)

F1WAE Interpreter with Deferred Substitutions

```
; interp : F1WAE? (listof FunDef?) DefSub? -> number?
(define (interp a-f1wae fundefs ds)
  (type-case F1WAE a-f1wae
    ...
    [app (fun-name arg)
         ...]))
```

F1WAE Interpreter with Deferred Substitutions

```
; interp : F1WAE? (listof FunDef?) DefSub? -> number?
(define (interp a-f1wae fundefs ds)
  (type-case F1WAE a-f1wae
    ...
    [app (fun-name arg)
         (local [(define a-fundef
                   (lookup-fundef fun-name fundefs))]
            (interp (fundef-body a-fundef)
                    fundefs
                    ...
                    (interp arg fundefs ds)
                    ...))]))
```

F1WAE Interpreter with Deferred Substitutions

```
; interp : F1WAE? (listof FunDef?) DefSub? -> number?
(define (interp a-f1wae fundefs ds)
  (type-case F1WAE a-f1wae
    ...
    [app (fun-name arg)
         (local [(define a-fundef
                   (lookup-fundef fun-name fundefs))]
           (interp (fundef-body a-fundef)
                   fundefs
                   (aSub (fundef-param-name a-fundef)
                         (interp arg fundefs ds)
                         (mtSub))))]))
```

Timing tests

```
(define (mk-sums n)
  (cond
    [(zero? n) 1]
    [else
     (define varn (string->symbol (format "x~a" n)))
     `{+ ,varn ,(mk-sums (- n 1))}]))

(define (mk-withs n body)
  (cond
    [(zero? n) body]
    [else
     (define varn (string->symbol (format "x~a" n)))
     `{with {,varn 1}
            ,(mk-withs (- n 1) body)}]))
```

Timing tests

```
(define (mk-exp n) (mk-withs n (mk-sums n)))

(test (mk-exp 2)
      `{with {x2 1}
             {with {x1 1}
                   {+ x2 {+ x1 1}}}})

(define (run n)
  (define (parse (mk-exp n)))
  (time (interp expr '() (mtSub))))
```

Timing tests

With the substitution-based interpreter, expect the difference between adjacent timings to be growing linearly. With the deferred-substitution-based one, you will also see linear growth, but if you make the environment use a more efficient data structure, that'll go away

(you may need to make the numbers bigger or smaller to see what is going on here)

```
(collect-garbage) (collect-garbage)
(collect-garbage) (collect-garbage)
(run 100) (run 110) (run 120)
(run 130) (run 140) (run 160)
```

Note: always run your timing tests with `racket` at the command line, not in DrRacket.