

Administrative stuff

Use piazza; don't email me directly.

AE

{- 20 {+ 10 10}}

{- 20 {+ 17 17}}

{- 20 {+ 3 3}}

WAE

```
{with {x 10}  
  {- 20 {+ x x}}}
```

```
{with {x 17}  
  {- 20 {+ x x}}}
```

```
{with {x 3}  
  {- 20 {+ x x}}}
```

FIWAE

```
{defun {f x}                                {f 10}  
  {- 20 {+ {+ x x}                          {f 17}  
        {+ x x}}}}  
  
                                           {f 3}
```

FIWAE

```
{defun {f x}                                {f 10}
  {- 20 {twice
        {twice x}}}}                       {f 17}

{defun {twice y}                             {f 3}
  {+ y y}}
```

FIWAE

```
{deffun {f x}                                {f 10}  
  {- 20 {twice  
        {twice x}}}}                        {f 17}
```

```
{deffun {twice y}                            {f 3}  
  {+ y y}}
```

```
; interp : FIWAE list-of-FunDef -> num
```

FIWAE

```
{deffun {f x}                                {f 10}  
  {- 20 {twice  
        {twice x}}}}                        {f 17}
```

```
{deffun {twice y}                             {f 3}  
  {+ y y}}
```

<FunDef> ::= {deffun {<id> <id>} <F1WAE>}

FIWAE


```
{deffun {f x}                                {f 10}
  {- 20 {twice
        {twice x}}}}                        {f 17}
```

```
{deffun {twice y}                            {f 3}
  {+ y y}}
```

```
<FunDef> ::= {deffun {<id> <id>} <F1WAE>}
<F1WAE>  ::= ...
          | {<id> <F1WAE>}
```


FIWAE Grammar

<FunDef> ::= {deffun {<id> <id>} <F1WAE>} 

<F1WAE> ::= <num>
| {+ <F1WAE> <F1WAE>}
| {- <F1WAE> <F1WAE>}
| {with {<id> <F1WAE>} <F1WAE>}
| <id>
| {<id> <F1WAE>} 

FIWAE Datatypes

```
(define-type FunDef
  [fundef (fun-name symbol?)
          (arg-name symbol?)
          (body F1WAE?)])
```

```
(define-type F1WAE
  [num (n number?)]
  [add (lhs F1WAE?)
        (rhs F1WAE?)]
  ...
  [app (fun-name symbol?)
        (arg F1WAE?)])
```

FIWAE Interp

```
(define (interp a-wae fundefs)
  (type-case F1WAE a-wae
    [num (n) n]
    [add (l r) (+ (interp l fundefs)
                  (interp r fundefs))]
    ...
    [app (name arg-expr)
         ...]))
```

FIWAE Interp

```
(define (interp a-wae fundefs)
  (type-case F1WAE a-wae
    [num (n) n]
    [add (l r) (+ (interp l fundefs)
                  (interp r fundefs))]
    ...
    [app (name arg-expr)
         ...]))
```

```
(test (interp (add (num 1) (num 1))
              empty)
      2)
```

FIWAE Interp

```
(define (interp a-wae fundefs)
  (type-case F1WAE a-wae
    [num (n) n]
    [add (l r) (+ (interp l fundefs)
                  (interp r fundefs))]
    ...
    [app (name arg-expr)
         ...]))
```

```
(test (interp (add (num 1) (num 1))
              (list
                (fundef 'f 'x
                        (add (id 'x) (num 3))))))
2)
```

FIWAE Interp

```
(define (interp a-wae fundefs)
  (type-case F1WAE a-wae
    [num (n) n]
    [add (l r) (+ (interp l fundefs)
                  (interp r fundefs))]
    ...
    [app (name arg-expr)
         ...]))
```

```
(test (interp (app 'f (num 1))
              (list
               (fundef 'f 'x
                       (add (id 'x) (num 3))))))
4)
```

FIWAE Interp

```
(define (interp a-wae fundefs)
  (type-case F1WAE a-wae
    [num (n) n]
    [add (l r) (+ (interp l fundefs)
                  (interp r fundefs))]
    ...
    [app (name arg-expr)
         ...]))

(test (interp (app 'f (num 10))
             (list
              (fundef 'f 'x
                      (sub (num 20)
                           (app 'twice (id 'x))))
              (fundef 'twice 'y
                      (add (id 'y) (id 'y))))
      0))
```

FIWAE Interp

```
(define (interp a-wae fundefs)
  (type-case F1WAE a-wae
    [num (n) n]
    [add (l r) (+ (interp l fundefs)
                  (interp r fundefs))]
    ...
    [app (name arg-expr)
         ... (interp arg-expr fundefs) ... ]))
```


FIWAE Interp

```
(define (interp a-wae fundefs)
  (type-case F1WAE a-wae
    [num (n) n]
    [add (l r) (+ (interp l fundefs)
                  (interp r fundefs))]
    ...
    [app (name arg-expr)
         ... (lookup-fundef name fundefs)
         ... (interp arg-expr fundefs) ... ]))
```

```
; lookup-fundef : symbol list-of-FunDef -> FunDef
```

FIWAE Interp

```
(define (interp a-wae fundefs)
  (type-case F1WAE a-wae
    [num (n) n]
    [add (l r) (+ (interp l fundefs)
                  (interp r fundefs))]
    ...
    [app (name arg-expr)
         (local [(define a-fundef
                   (lookup-fundef name fundefs))]
                 (interp (subst (fundef-body a-fundef)
                                (fundef-arg-name a-fundef)
                                (interp arg-expr fundefs))
                          fundefs))]))))
```

Lookup

```
; lookup-fundef : symbol list-of-FunDef -> FunDef  
(define (lookup-fundef name fundefs)  
  ...)
```

Lookup

```
; lookup-fundef : symbol list-of-FunDef -> FunDef
(define (lookup-fundef name fundefs)
  (cond
    [(empty? fundefs)
     ...]
    [else
     ... (first fundefs)
     ... (lookup-fundef name (rest fundefs))
     ...])))
```

Lookup

```
; lookup-fundef : symbol list-of-FunDef -> FunDef
(define (lookup-fundef name fundefs)
  (cond
    [(empty? fundefs)
     (error 'interp "unknown function")]
    [else
     (if (symbol=? name (fundef-fun-name
                        (first fundefs)))
         (first fundefs)
         (lookup-fundef name (rest fundefs))))]))
```

Subst

```
; subst : F1WAE symbol num -> F1WAE
(define (subst a-flwae sub-id val)
  (type-case F1WAE a-flwae
    ...
    [app (fun-name arg)
         (app fun-name (subst arg sub-id val))])))
```