

# ITP20005 L6/L7

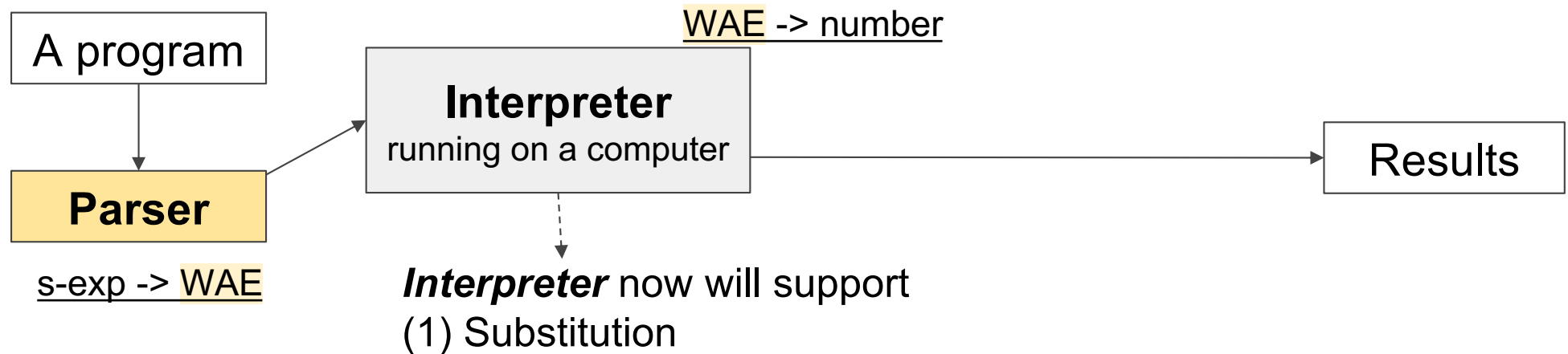
# Substitution

Lecture06  
JC

# Q&A

- What are we doing now? Are we learning Racket of PL?
- A language called **AE**
  - arithmetic (addition and subtraction)  $3 + (5 - 2)$
  - concrete syntax `{+ 3 {- 5 2}}`
  - abstract syntax **AE**  
`(add (num 3) (sub (num 5) (num 2)))`
  - parser `parse: sexp -> AE`  
`(test (parse '{+ 3 {- 5 2}})`  
`(add (num 3) (sub (num 5) (num 2))))`
  - interpreter `interp: AE -> number`  
`(test (interp (parse '{+ 3 {- 5 2}})) 6)`

# Big Picture (modeling languages: substitution)



\* **WAE**: AE that support identifiers

# Agenda

- Identifiers
- Substitution
- Binding
- Scope

# Motivating Example

# Repeated expressions

$\{+ \{+ 5 5\} \{+ 5 5\}\}$

# Repeated expressions

$\{+ \{+ \{+ 5 5\} \{+ 5 5\}\} \{+ \{+ 5 5\} \{+ 5 5\}\}\}$

# Repeated expressions

$\{+ \{+ \{+ 5 5\} \{+ 5 5\}\} \{+ \{+ 5 5\} \{+ 5 5\}\}\}$

When we have any repeated expressions, we might make **a mistake** and evaluating them **wastes computational cycles**.



# See some example in your favorite language

```
// sum from 1 to 10 and repeat it three times to get total sum  
int totalSum = (1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10) + (1 + 2 + 3  
+ 4 + 5 + 6 + 7 + 8 + 9 + 10) + (1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 +  
10);
```

# See some example in your favorite language

```
// sum from 1 to 10 and repeat it three times to get total sum  
int partialSum = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10;  
int totalSum = partialSum + partialSum + partialSum;
```

# See some example in your favorite language

```
// sum from 1 to 10 and repeat it three times to get total sum  
int partialSum = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10;  
int totalSum = partialSum + partialSum + partialSum;
```

1. **Computational benefit:** *partialSum* is calculated once.
2. We call *partialSum* an **identifier**.
3. **Substitution:** To get the result of totalSum, *partialSum* needs to be replaced with 55 while computing the total sum.



# Identifier

to avoid this redundancy

# Identifiers

- Name/identify the value of an expression
- Reuse its name in place of the larger computation
- Similar to variables
  - But variables imply the value of the identifier can change.
- Note that in ***our current language***, we do not initially offer any way of changing the associated value with the identifier.  
⇒ Identifier (works like a **constant** in our current language)

# Topics we cover and schedule (tentative)

- Racket tutorials (L2,3)
- Modeling languages (L4,5)
- Interpreting arithmetic (L5)
- Language principles
  - **Substitution** (L6-7)
  - Function (L8)
  - Deferring Substitution (L9)
  - First-class Functions (L10-L12)
  - Laziness (L13,14)
  - Recursion (L15,16)
  - Mutable data structures (L17,18,19,20)
  - Variables (L21,22)
  - Continuations (L23-26)
- Guest Video Lecture (L27)

# Sample program that uses identifiers

{+ {+ 5 5} {+ 5 5}}

**How do you want to deal with this expression with an identifier in our new language??**

\* Remember that we use an uniform syntax: parenthesized {} prefix in this class.

# Sample program that uses identifiers

{+ {+ 5 5} {+ 5 5}}

;; use an identifier for the repeated expressions, {+ 5 5}

;; We would like to use 'with' keyword to define **an identifier** for **an arithmetic expression** and use the identifier **for another arithmetic expression**.

{with {x {+ 5 5}} {+ x x}}



# A more elaborate example

descend: move to the inner expression to continue calculating.

# A more elaborate example

$\{+ \{- \{+ 5 5\} 3\} \{- \{+ 5 5\} 3\}\}$

$\{\text{with } \{x \{+ 5 5\}\}$

$\{\text{with } \{y \{- x 3\}\}$

$\{+ y y\}\}$

$= \{\text{with } \{x 10\} \{\text{with } \{y \{- x 3\}\} \{+ y y\}\}\}$

$= \{\text{with } \{x 10\} \{\text{with } \{y \{- 10 3\}\} \{+ y y\}\}\}$

$= \{\text{with } \{y \{- 10 3\}\} \{+ y y\}\}$

$= \{\text{with } \{y 7\} \{+ y y\}\}$

$= \{\text{with } \{y 7\} \{+ 7 7\}\}$

$= \{+ 7 7\}$

$= 14$

[+ operator]

[substitution]

[descend\*]

[- operator]

[substitution]

[descend]

[+ operator]

\* descend: move to the inner expression to continue calculating.

# Then, improve AE to support identifiers!

- 'with' with arithmetic expressions  $\Rightarrow$  WAE in BNF

$\langle \text{WAE} \rangle ::= \langle \text{num} \rangle$

|  $\{ + \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

|  $\{ - \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

| ...

| ...

# Then, improve AE to support identifiers!

- 'with' with arithmetic expressions  $\Rightarrow$  WAE in BNF

$\langle \text{WAE} \rangle ::= \langle \text{num} \rangle$

$| \{ + \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ - \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ \text{with } \{ \langle \text{id} \rangle \langle \text{WAE} \rangle \} \langle \text{WAE} \rangle \}$

$| \langle \text{id} \rangle$

- \* We have now two additional rules to AE in BNF:
  - (1) for associating values with an identifier
  - (2) for using the identifier

# Then, improve AE to support identifiers!

- Identifiers

$\langle \text{id} \rangle ::= x, y, \text{plus}, \text{factorial}, \text{swap}, \text{interp}, \dots$

# Then, improve AE to support identifiers!

- Identifiers

$\langle id \rangle ::= x, y, \text{plus}, \text{factorial}, \text{swap}, \text{interp}, \dots$

Or (in scheme: a bit different from Racket)

$\langle id \rangle ::= \langle initial \rangle \langle subsequent \rangle^* | + | - |$

...

$\langle initial \rangle ::= \langle letter \rangle | ! | \$ | \% | \& | * | : | < | = | > | ? |$

$\sim | \_ | ^$

$\langle subsequent \rangle ::= \langle initial \rangle | \langle digit \rangle | . | + | -$

$\langle letter \rangle ::= a | b | \dots | z$

$\langle digit \rangle ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$

\* zero or more occurrences

# Then, improve AE to support identifiers!

- 'with' with arithmetic expressions

$\langle \text{WAE} \rangle ::= \langle \text{num} \rangle$

$| \{ + \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ - \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ \text{with } \{ \langle \text{id} \rangle \langle \text{WAE} \rangle \} \langle \text{WAE} \rangle \}$

$| \langle \text{id} \rangle$

$\{ \text{with } \{ x \{ + 1 2 \} \}$

$\{ + x x \} \}$

# Then, improve AE to support identifiers!

- 'with' with arithmetic expressions

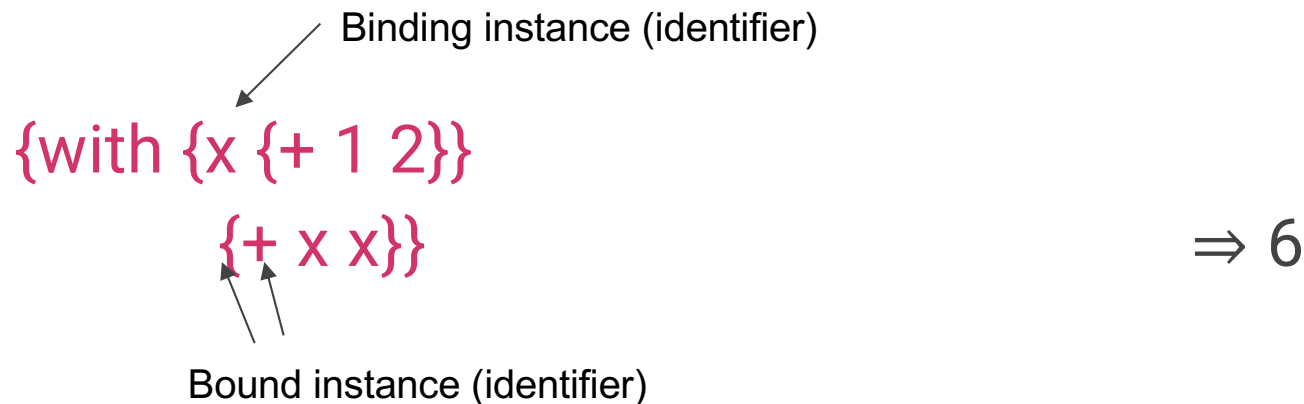
$\langle \text{WAE} \rangle ::= \langle \text{num} \rangle$

$| \{ + \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ - \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ \text{with} \{ \langle \text{id} \rangle \langle \text{WAE} \rangle \} \langle \text{WAE} \rangle \}$

$| \langle \text{id} \rangle$





# Then, improve AE to support identifiers!

- 'with' with arithmetic expressions

$\langle \text{WAE} \rangle ::= \langle \text{num} \rangle$

$| \{ + \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ - \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ \text{with} \{ \langle \text{id} \rangle \langle \text{WAE} \rangle \} \langle \text{WAE} \rangle \}$

$| \langle \text{id} \rangle$

$\{ \text{with} \{ x \{ + 1 2 \} \}$

$\{ + x y \}$

identifier

Free identifier (instance)

$\Rightarrow$  error: free

# Then, improve AE to support identifiers!

- 'with' with arithmetic expressions

$\langle \text{WAE} \rangle ::= \langle \text{num} \rangle$

$| \{ + \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ - \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ \text{with } \{ \langle \text{id} \rangle \langle \text{WAE} \rangle \} \langle \text{WAE} \rangle \}$

$| \langle \text{id} \rangle$

$x \Rightarrow \text{error: free identifier}$

# Then, improve AE to support identifiers!

- 'with' with arithmetic expressions

$\langle \text{WAE} \rangle ::= \langle \text{num} \rangle$

$| \{ + \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ - \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ \text{with} \{ \langle \text{id} \rangle \langle \text{WAE} \rangle \} \langle \text{WAE} \rangle \}$

$| \langle \text{id} \rangle$

$\{ + \{ \text{with} \{ x \{ + 1 2 \} \}$

$\{ + x x \} \}$

$\{ \text{with} \{ x \{ - 4 3 \} \}$

$\{ + x x \} \} \}$

# Then, improve AE to support identifiers!

- 'with' with arithmetic expressions

$\langle \text{WAE} \rangle ::= \langle \text{num} \rangle$

$| \{ + \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ - \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ \text{with} \{ \langle \text{id} \rangle \langle \text{WAE} \rangle \} \langle \text{WAE} \rangle \}$

$| \langle \text{id} \rangle$

$\{ + \{ \text{with} \{ x \{ + 1 2 \} \}$

$\{ + x x \} \}$

$\{ \text{with} \{ x \{ - 4 3 \} \}$

$\{ + x x \} \} \}$

$\Rightarrow 8$

# Then, improve AE to support identifiers!

- 'with' with arithmetic expressions

$\langle \text{WAE} \rangle ::= \langle \text{num} \rangle$

$| \{ + \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ - \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ \text{with} \{ \langle \text{id} \rangle \langle \text{WAE} \rangle \} \langle \text{WAE} \rangle \}$

$| \langle \text{id} \rangle$

$\{ \text{with} \{ x \{ + \ 1 \ 2 \} \}$

$\{ \text{with} \{ x \{ - \ 4 \ 3 \} \}$

$\{ + \ x \ x \} \}$

# Then, improve AE to support identifiers!

- 'with' with arithmetic expressions

$\langle \text{WAE} \rangle ::= \langle \text{num} \rangle$

$| \{ + \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ - \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ \text{with } \{ \langle \text{id} \rangle \langle \text{WAE} \rangle \} \langle \text{WAE} \rangle \}$

$| \langle \text{id} \rangle$

$\{ \text{with } \{ x \{ + 1 2 \} \}$

$\{ \text{with } \{ x \{ - 4 3 \} \}$

$\{ + x x \} \}$

$\Rightarrow 2$

# Then, improve AE to support identifiers!

- 'with' with arithmetic expressions

$\langle \text{WAE} \rangle ::= \langle \text{num} \rangle$

$| \{ + \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ - \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ \text{with } \{ \langle \text{id} \rangle \langle \text{WAE} \rangle \} \langle \text{WAE} \rangle \}$

$| \langle \text{id} \rangle$

$\{ \text{with } \{ x \{ + 1 2 \} \}$

$\{ \text{with } \{ y \{ - 4 3 \} \}$

$\{ + x x \} \}$

# Then, improve AE to support identifiers!

- 'with' with arithmetic expressions

$\langle \text{WAE} \rangle ::= \langle \text{num} \rangle$

$| \{ + \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ - \langle \text{WAE} \rangle \langle \text{WAE} \rangle \}$

$| \{ \text{with } \{ \langle \text{id} \rangle \langle \text{WAE} \rangle \} \langle \text{WAE} \rangle \}$

$| \langle \text{id} \rangle$

$\{ \text{with } \{ x \{ + 1 2 \} \}$

$\{ \text{with } \{ y \{ - 4 3 \} \}$

$\{ + x x \} \}$

$\Rightarrow 6$



# 1. Define type WAE!

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

(define-type WAE

  [num (n number?)]

  [add (lhs WAE?) (rhs WAE?)]

  [sub (lhs WAE?) (rhs WAE?)]

  [with (name symbol?) (named-expr WAE?) (body WAE?)]

  [id (name symbol?)])

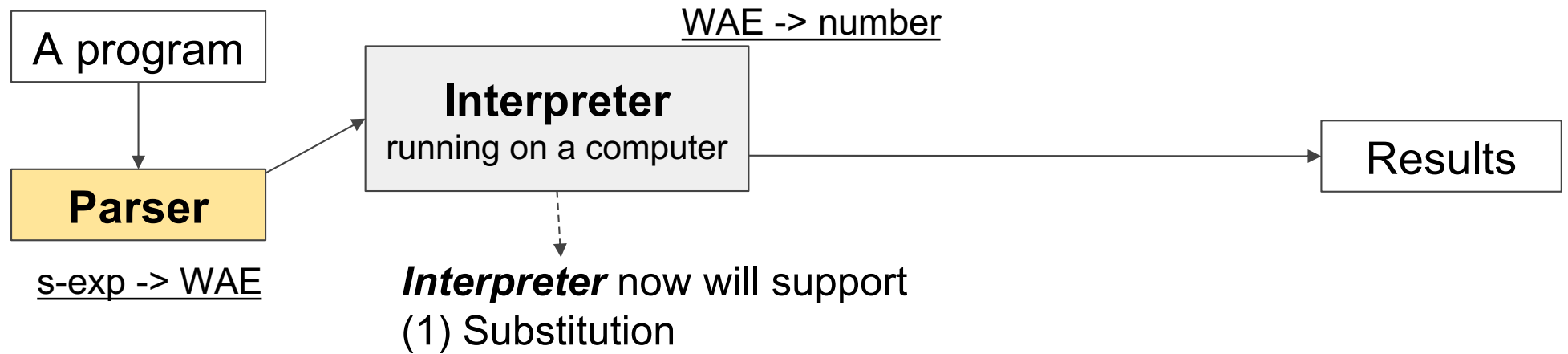
## 2. Implement a parser for WAE!

## 2. Implement a parser for WAE! (by using match)

## 2. Implement a parser for WAE! (by using match)

## 2. Implement a parser for WAE! (by using match)

# Big Picture (modeling languages: substitution)



## 2. Implement a parser for WAE! (by using match)

# Recall this example

$\{+ \{- \{+ 5 5\} 3\} \{- \{+ 5 5\} 3\}\}$

$\{\text{with } \{x \{+ 5 5\}\}$

$\{\text{with } \{y \{- x 3\}\}$

$\{+ y y\}\}$

$= \{\text{with } \{x 10\} \{\text{with } \{y \{- x 3\}\} \{+ y y\}\}$

$= \{\text{with } \{x 10\} \{\text{with } \{y \{- 10 3\}\} \{+ y y\}\}$

$= \{\text{with } \{y \{- 10 3\}\} \{+ y y\}\}$

$= \{\text{with } \{y 7\} \{+ y y\}\}$

$= \{\text{with } \{y 7\} \{+ 7 7\}\}$

$= \{+ 7 7\}$

$= 14$

[+ operator]

[substitution]

[descend\*]

[- operator]


[substitution]

[descend]

[+ operator]

\* descend: move to the inner expression to continue calculating.





Before implementing an WAE interpreter, we need to think how to deal with identifiers in the interpreter.

⇒ Substitution

# Defining Substitution

- Definition 1 (Substitution)

- To substitute identifier  $i$  in  $e$  with the expression  $v$ , replace all identifiers in  $e$  that have the name  $i$ , with the expression  $v$ .

; Our code

;  $i$ :  $x$

;  $v$ : 5

;  $e$ :  $\{+ x x\}$

$\{\text{with } \{x\ 5\} \{+ x x\}\}$

- Substitution based on Definition 1  
?

# Defining Substitution

- Definition 1 (Substitution)

- To substitute identifier  $i$  in  $e$  with the expression  $v$ , replace all identifiers in  $e$  that have the name  $i$ , with the expression  $v$ .

; Our code

;  $i$ :  $x$

;  $v$ : 5

;  $e$ :  $\{+ x x\}$

$\{\text{with } \{x\ 5\} \{+ x x\}\}$

- Substitution based on Definition 1  
 $\{\text{with } \{x\ 5\} \{+ 5 5\}\}$

# Defining Substitution

- How about this code?

`{with {x 5} {+ 10 y}} ;; [substitution]`

to

`{with {x 5} {+ 10 y}}`

\* **No substitutions** occur since there is no instances of x in the expression.

# Defining Substitution

- How about this based on Definition 1?  
 $\{\text{with } \{x \ 5\} \{+ \ x \ \{\text{with } \{x \ 3\} \ 10\}\}\} \ ; \text{ [substitution]}$

to

$\{\text{with } \{x \ 5\} \{+ \ 5 \ \{\text{with } \{5 \ 3\} \ 10\}\}\}$

# Defining Substitution

- How about this based on Definition 1?  
 $\{\text{with } \{x \ 5\} \{+ \ x \ \{\text{with } \{x \ 3\} \ 10\}\}\} \ ; \text{ [substitution]}$

to

$\{\text{with } \{x \ 5\} \{+ \ 5 \ \{\text{with } \{5 \ 3\} \ 10\}\}\}$

**???? Syntactically illegal! Our parser will reject this expression!**  
**Recall WAE in BNF:**  $\mid \{\text{with } \{<\text{id}> \ <\text{WAE}>\} \ <\text{WAE}>$

# Defining Substitution

- So, we need other detailed definitions to make our algorithm for Substitution *precisely*.
  - Definition 2 (Binding Instance)

A binding instance of an identifier is the instance of the identifier that gives it its value. In WAE, *the <id> position of a 'with' is the only binding instance*.

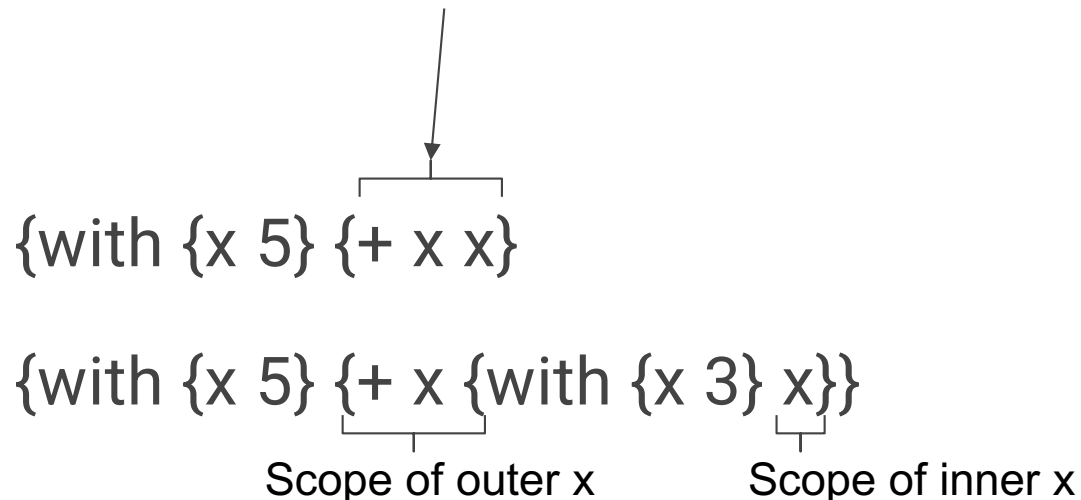
  
{with {x 5} {+ x 5}}

# Defining Substitution

- So, we need other detailed definitions to make our algorithm for Substitution *precisely*.
  - Definition 2 (Binding Instance)

A binding instance of an identifier is the instance of the identifier that gives it its value. In WAE, *the <id> position of a 'with' is the only binding instance*.
  - Definition 3 (Scope)

The scope of a binding instance is the region of program text in which instances of the identifier refer to the value bound by the binding instance.





# Defining Substitution

- So, we need other detailed definitions to make our algorithm for Substitution *precisely*.
  - Definition 3 (Scope)

The scope of a binding instance is the region of program text in which instances of the identifier refer to the value bound by the binding instance.
  - Definition 4 (Bound Instance)

An identifier is bound if it is contained within the scope of a binding instance of its name.

$\{\text{with } \{x \ 5\} \underbrace{\{+ \ x \ x\}}\}$

# Defining Substitution

- So, we need other detailed definitions to make our algorithm for Substitution *precisely*.
  - Definition 5 (Free identifier/Instance)  
An identifier not contained in the scope of any binding instance of its name is said to be free.

{with {x 5} {+ x {with {y 3} x}}}

free instance as there is  
no binding instance of it  
within its scope.

# Defining Substitution

- Definition 6 (Substitution, take 2)
  - To substitute identifier  $i$  in  $e$  with the expression  $v$ , replace all identifiers in  $e$  which are not binding instances that have the name  $i$  with expression  $v$ .

```
; Our code  
;  $i$ : x  
;  $v$ : 5  
;  $e$ : {+ x {with {x 3} 10}}  
      {with {x 5} {+ x {with {x 3} 10}}}
```

- Based on Definition 6  
?

# Defining Substitution

- Definition 6 (Substitution, take 2)
  - To substitute identifier  $i$  in  $e$  with the expression  $v$ , replace all identifiers in  $e$  which are not binding instances that have the name  $i$  with expression  $v$ .

; Our code

;  $i$ :  $x$

;  $v$ : 5

;  $e$ :  $\{+ x \{with \{x \ 3\} \ 10\}\}$

$\{with \{x \ 5\} \{+ x \{with \{x \ 3\} \ 10\}\}\}$

- Based on Definition 6  
 $\{with \{x \ 5\} \{+ 5 \{with \{x \ 3\} \ 10\}\}\}$

# Defining Substitution

- Definition 6 (Substitution, take 2)
  - To substitute identifier  $i$  in  $e$  with the expression  $v$ , replace all identifiers in  $e$  which are not binding instances that have the name  $i$  with expression  $v$ .

How about the following code?

; Our code

;  $i$ :  $x$

;  $v$ : 5

;  $e$ :  $\{+ x \{with \{x \ 3\} x\}\}$

$\{with \{x \ 5\} \{+ x \{with \{x \ 3\} x\}\}\}$  ;; the value of the program is 8

- But....based on Definition 6, the substitution is?

# Defining Substitution

- Definition 6 (Substitution, take 2)
  - To substitute identifier  $i$  in  $e$  with the expression  $v$ , replace all identifiers in  $e$  which are not binding instances that have the name  $i$  with expression  $v$ .

How about the following code?

; Our code

;  $i$ :  $x$

;  $v$ : 5

;  $e$ :  $\{+ x \{with \{x\ 3\} x\}\}$

$\{with \{x\ 5\} \{+ x \{with \{x\ 3\} x\}\}\}$  ;; the value of the program is 8

- But....based on Definition 6, the substitution is  
 $\{with \{x\ 5\} \{+ 5 \{with \{x\ 3\} 5\}\}\}$  ;; the value is 10

????What's wrong with here????

# Defining Substitution

- Definition 7 (Substitution, take 3)
  - To substitute identifier  $i$  in  $e$  with the expression  $v$ , replace all non-binding identifiers in  $e$  having the name  $i$  with the expression  $v$ , **unless the identifier is in a scope different from that introduced by  $i$ .**

; Our code

;  $i$ :  $x$

;  $v$ : 5

;  $e$ :  $\{+ x \{with \{x \ 3\} x\}\}$

$\{with \{x \ 5\} \{+ x \{with \{x \ 3\} x\}\}\}$  ;; the value of the program is 8

- Based on the current Definition 7, the substitution is  $\{with \{x \ 5\} \{+ 5 \{with \{x \ 3\} 3\}\}\}$  ;; the value is 8

# Defining Substitution

- Definition 7 (Substitution, take 3)
  - To substitute identifier  $i$  in  $e$  with the expression  $v$ , replace all non-binding identifiers in  $e$  having the name  $i$  with the expression  $v$ , unless the identifier is in a scope different from that introduced by  $i$ .

How about the following code?

; Our code

;  $i$ :  $x$

;  $v$ : 5

;  $e$ :  $\{+ x \{with \{y \ 3\} x\}\}$

$\{with \{x \ 5\} \{+ x \{with \{y \ 3\} x\}\}\}$  ;; the value of the program is 10

- But....based on Definition 7, the substitution cannot be done (error) because of  $x$  has no value.

$\{with \{x \ 5\} \{+ 5 \{with \{y \ 3\} x\}\}\}$  ;; error



# Defining Substitution (final)

- Definition 8 (Substitution, take 4)
  - To substitute identifier  $i$  in  $e$  with the expression  $v$ , replace all bound instances and replace all non-binding identifiers in  $e$  having the name  $i$  with the expression  $v$ , except within nested scopes of  $i$ .

; Our code

;  $i$ :  $x$

;  $v$ : 5

;  $e$ :  $\{+ x \{with \{y \ 3\} x\}\}$

$\{with \{x \ 5\} \{+ x \{with \{y \ 3\} x\}\}\}$  ;; the value of the program is 10

- Based on Definition 8...  
 $\{with \{x \ 5\} \{+ 5 \{with \{y \ 3\} 5\}\}\}$  ;; the value is 10.

# Defining Substitution (final)

- Definition 9 (Substitution, take 5)
  - To substitute identifier  $i$  in  $e$  with the expression  $v$ , replace all bound instances of  $i$  and **replace all free instances of  $i$  in  $e$  with  $v$ .**

; Our code

;  $i$ :  $x$

;  $v$ : 5

;  $e$ :  $\{+ x \{with \{y \ 3\} x\}\}$

$\{with \{x \ 5\} \{+ x \{with \{y \ 3\} x\}\}\}$  ;; the value of the program is 10

- Based on Definition 9...  
 $\{with \{x \ 5\} \{+ 5 \{with \{y \ 3\} 5\}\}\}$  ;; the value is 10.

JC

## **TODO**

Read Chapter 3. Substitution

[jcnam@handong.edu](mailto:jcnam@handong.edu)  
<https://lifove.github.io>

\* Slides are from Prof. Sukeyoung Ryu's PL class in 2018 Spring  
or created by JC based on the main text book.

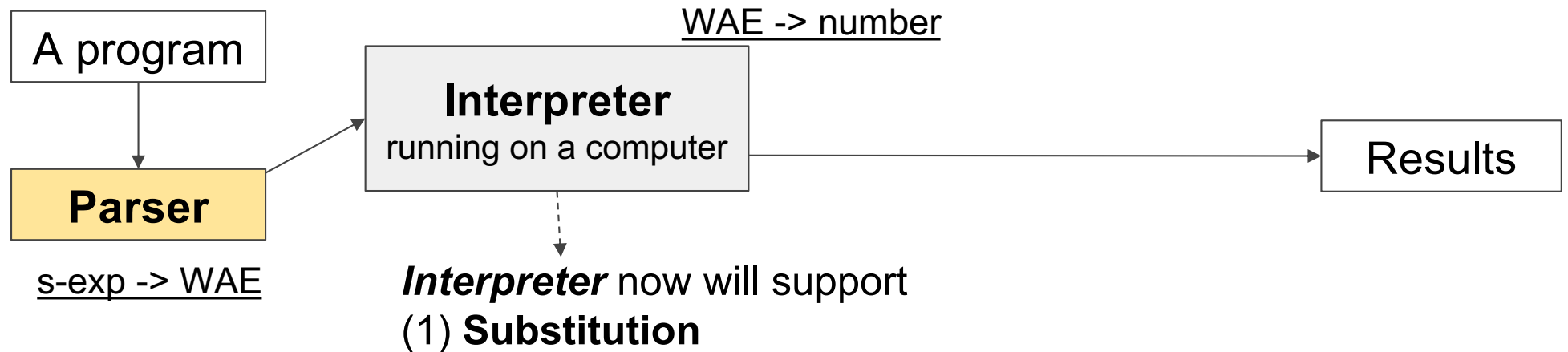
# ITP20005 L7

# Substitution (2)

Lecture07

JC

# Big Picture (modeling languages: substitution)



# See this example...

- Find Bound/Binding/Free instances and scopes of binding identifiers

$\{\text{with } \{x\ 5\} \{+ x \{\text{with } \{y\ x\} x\}\}\}$

$\{\text{with } \{x\ 5\} \{+ x \{\text{with } \{x\ \{+ x\ 1\}\} x\}\}\}$

# Defining Substitution (final)

- Definition 9 (Substitution, take 5)
  - To substitute identifier *i* in *e* with the expression *v*, replace all bound instances of *i* and **replace all free instances of *i* in *e* with *v*.**

; Our code

; *i*: x

; *v*: 5

; *e*: {+ x {with {y x} x}}

{with {x 5} {+ x {with {y x} x}}} ;; the value of the program is 10

- Based on Definition 9...  
{with {x 5} {+ 5 {with {y 5} 5}}} ;; the value is 10.

# 1. Define type WAE!

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

(define-type WAE

[num (n number?)]

[add (lhs WAE?) (rhs WAE?)]

[sub (lhs WAE?) (rhs WAE?)]

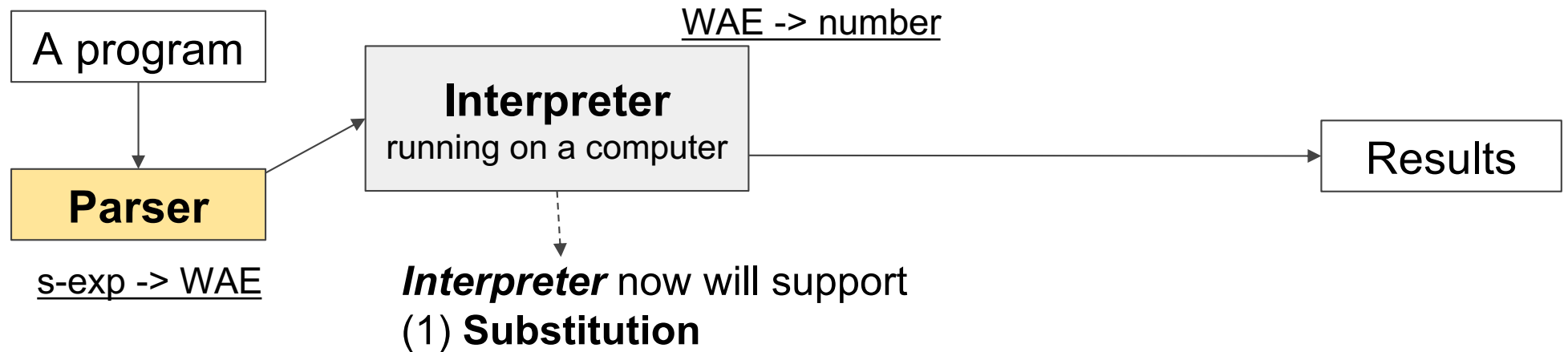
[with (name symbol?) (named-expr WAE?) (body WAE?)]

[id (name symbol?)])



## 2. Implement a parser for WAE! (by using match)

# Big Picture (modeling languages: substitution)



# WAE and its interpretation

(with 'x (num 5) (add (id 'x) (id 'x)))

[Substitution]

(with 'x (num 5) (add (num 5) (num 5)))

[Descend]

(add (num 5) (num 5))

[add operator]

10

[Result]

⇒ We need an interpreter function as well as a substitution function.

\* descend: move to the inner expression to continue calculating.

### 3. Implement substitution for WAE Interpreter

$e$        $i$        $v$   
 ↓      ↓      ↓

```

; [contract] subst: WAE symbol number -> WAE
; (here, symbol is an identifier and number is the value for the identifier)
; [purpose] to substitute second argument with third argument in first argument,
; as per the rules of substitution; the resulting expression contains
; no free instances of the second argument
(define (subst wae i val)
  ...)
  
```

```

; [tests]
  
```

```

; {with {x 10} 5}           ⇒    10 for x in 5           ⇒ 5
  
```

```

(test (subst (num 5) 'x 10) (num 5))
  
```

```

; {with {x 10} {+ 1 x}}     ⇒    10 for x in {+ 1 x}     ⇒ {+ 1 10}
  
```

```

(test (subst (add (num 1) (id 'x)) 'x 10) (add (num 1) (num 10)))
  
```

```

; {with {x 10} x}           ⇒    10 for x in x           ⇒ 10
  
```

```

(test (subst (id 'x) 'x 10) (num 10))
  
```

```

; {with {x 10} y}           ⇒    10 for x in y           ⇒ y (no
substitution)
  
```

### 3. Implement substitution for WAE Interpreter

```
; [contract] subst: WAE symbol number -> WAE  
; ...
```

```
; [tests]
```

```
; {with {x 10} 5}           ⇒    10 for x in 5           ⇒ 5
```

```
(test (subst (num 5) 'x 10) (num 5))
```

```
; {with {x 10} {+ 1 x}}     ⇒    10 for x in {+ 1 x}     ⇒ {+ 1 10}
```

```
(test (subst (add (num 1) (id 'x)) 'x 10) (add (num 1) (num 10)))
```

```
; {with {x 10} x}           ⇒    10 for x in x           ⇒ 10
```

```
(test (subst (id 'x) 'x 10) (num 10))
```

```
; {with {x 10} y}           ⇒    10 for x in y           ⇒ y (no  
substitution)
```

```
(test (subst (id 'y) 'x 10) (id 'y))
```

```
; {with {y 10} {- x 1}}     ⇒    10 for y in {- x 1}     => {- x 1}
```

```
(no substitution)
```

```
(test (subst (sub (id 'x) (num 1)) 'y 10) (sub (id 'x) (num 1)))
```

### 3. Implement substitution for WAE Interpreter

```
; [contract] subst: WAE symbol number -> WAE
```

```
; [...]
```

```
(define (subst wae id val)
```

```
  (type-case WAE wae
```

```
    [num      (n)                wae]
    [add      (l r)              ...]
    [sub      (l r)              ...]
    [with     (i v e)            ...]
    [id       (s)                ...]))
```

```
; [tests]
```

```
; {with {x 10} 5}           ⇒    10 for x in 5           ⇒ 5
```

```
(test (subst (num 5) 'x 10) (num 5))
```

```
; {with {x 10} {+ 1 x}}     ⇒    10 for x in {+ 1 x} ⇒ {+ 1 10}
```

```
(test (subst (add (num 1) (id 'x)) 'x 10) (add (num 1) (num 10)))
```

```
; {with {x 10} x}           ⇒    10 for x in x           ⇒ 10
```

```
(test (subst (id 'x) 'x 10) (num 10))
```

```
; {with {x 10} y}           ⇒    10 for x in y           ⇒ y (no substitution)
```

```
(test (subst (id 'y) 'x 10) (id 'y))
```

```
; {with {y 10} {- x 1}}     ⇒    10 for y in {- x 1}     ⇒ {- x 1} (no substitution)
```

```
(test (subst (sub (id 'x) (num 1)) 'y 10) (sub (id 'x) (num 1)))
```

### 3. Implement substitution for WAE Interpreter

; [contract] subst: WAE symbol number -> WAE

(define (subst wae idtf val)

(type-case WAE wae

[num (n) wae]

[add (l r) (add (subst l idtf val) (subst r idtf val))]

[sub (l r) (sub (subst l idtf val) (subst r

idtf val))]

[with (i v e) ...]

[id (s) ...]))

; [tests]

; {with {x 10} {+ 1 x}} ⇒ 10 for x in {+ 1 x} ⇒ {+ 1 10}

(test (subst (add (num 1) (id 'x)) 'x 10) (add (num 1) (num 10)))

; {with {x 10} x} ⇒ 10 for x in x ⇒ 10

(test (subst (id 'x) 'x 10) (num 10))

; {with {x 10} y} ⇒ 10 for x in y

⇒ y (no substitution)

(test (subst (id 'y) 'x 10) (id 'y))

; {with {y 10} {- x 1}} ⇒ 10 for y in {- x 1}

⇒ {- x 1} (no substitution)

(test (subst (sub (id 'x) (num 1)) 'y 10) (sub (id 'x) (num 1)))

# 3. Implement substitution for WAE Interpreter

```
; [contract] subst: WAE symbol number -> WAE
```

```
(define (subst wae idtf val)
```

```
  (type-case WAE wae
```

```
    [num      (n)                wae]
```

```
    [add      (l r)              (add (subst l idtf val) (subst r idtf
```

```
val)))]
```

```
    [sub      (l r)              (sub (subst l idtf val) (subst r
```

```
idtf val)))]
```

```
    [with     (i v e) ...]
```

```
    [id       (s)                (if (symbol=? s idtf) (num val)
```

```
wae))]))
```

```
; [tests]
```

```
; {with {x 10} {+ 1 x}}      => 10 for x in {+ 1 x} => {+ 1 10}
```

```
(test (subst (add (num 1) (id 'x)) 'x 10) (add (num 1) (num 10))))
```

```
; {with {x 10} x}           => 10 for x in x           => 10
```

```
(test (subst (id 'x) 'x 10) (num 10))
```

```
; {with {x 10} y}           => 10 for x in y
```

```
=> y (no substitution)
```

```
(test (subst (id 'y) 'x 10) (id 'y))
```



### 3. Implement substitution for WAE Interpreter

; [contract] subst: WAE symbol number -> WAE

(define (subst wae idtf val)

(type-case WAE wae

[num (n) wae]

[add (l r) (add (subst l idtf val) (subst r idtf

val))]

[sub (l r) (sub (subst l idtf val) (subst r

idtf val))]

[with (i v e) ...]

[id (s) (if (symbol=? s idtf) (num val)

wae)))]

; {with {x 10} {...{with {y 17} x}}} ⇒ 10 for x in {with {y 17} x} ⇒ {with {y 17} 10}

(test (subst (with 'y (num 17) (id 'x)) 'x 10) (with 'y (num 17) (num 10)))

; {with {x 10} {...{with {y x} y}}} ⇒ 10 for x in {with {y x} y} ⇒ {with {y 10} y}

(test (subst (with 'y (id 'x) (id 'y)) 'x 10) (with 'y (num 10) (id 'y)))

; {with {x 10} {...{with {x y} x}}} ⇒ 10 for x in {with {x y} x} ⇒ {with {x y} x}

(test (subst (with 'x (id 'y) (id 'x)) 'x 10) (with 'x (id 'y) (id 'x)))

### 3. Implement substitution for WAE Interpreter

```
; [contract] subst: WAE symbol number -> WAE
(define (subst wae idtf val)
```

```
(type
```

```
val)))
```

```
idtf val)))
```

One subst function call conducts substitutions for one binding identifier!

```
st r idtf
```

```
val) (subst r
```

```
[with (rve) ...]
[id (s)
```

```
(if (symbol=? s idtf) (num val)
```

```
wae)))))
```

```
; {with {x 10} {...{with {y 17} x}}} ⇒ 10 for x in {with {y 17} x} ⇒ {with {y 17} 10}
```

```
(test (subst (with 'y (num 17) (id 'x)) 'x 10) (with 'y (num 17) (num 10)))
```

```
; {with {x 10} {...{with {y x} y}}} ⇒ 10 for x in {with {y x} y} ⇒ {with {y 10} y}
```

```
(test (subst (with 'y (id 'x) (id 'y)) 'x 10) (with 'y (num 10) (id 'y)))
```

```
; {with {x 10} {...{with {x y} x}}} ⇒ 10 for x in {with {x y} x} ⇒ {with {x y} x}
```

```
(test (subst (with 'x (id 'y) (id 'x)) 'x 10) (with 'x (id 'y) (id 'x)))
```

### 3. Implement substitution for WAE Interpreter

; [contract] subst: WAE symbol number -> WAE

(define (subst wae idtf val)

(type-case WAE wae

[num (n) wae]

[add (l r) (add (subst l idtf val) (subst r idtf

val)))]

[sub (l r) (sub (subst l idtf val) (subst r

idtf val)))]

[with (i v e) (with i (subst v idtf val) (if (symbol=? i idtf) e

(subst e idtf val)))]

[id (s) (if (symbol=? s idtf) (num val)

wae)))]

; {with {x 10} {...{with {y 17} x}} ⇒ 10 for x in {with {y 17} x} ⇒ {with {y 17} 10}

(test (subst (with 'y (num 17) (id 'x)) 'x 10) (with 'y (num 17) (num 10)))

; {with {x 10} {...{with {y x} y}} ⇒ 10 for x in {with {y x} y} ⇒ {with {y 10} y}

(test (subst (with 'v (id 'x) (id 'v)) 'x 10) (with 'v (num 10) (id 'v)))

### 3. Implement substitution for WAE Interpreter

; [contract] subst: WAE symbol number -> WAE

(define (subst wae idtf val)

(type-case WAE wae

[num (n) wae]

[add (l r) (add (subst l idtf val) (subst r idtf  
val)))]

[sub (l r) (sub (subst l idtf val) (subst r  
idtf val))]

[with (i v e) (with i (subst v idtf val) (if (symbol=? i idtf) e

(subst e idtf val))))]

[id (s) (if (symbol=? s idtf) (num val)  
wae))]))

## 4. Implement WAE interpreter

; interp: WAE -> number

(define (interp wae)

(type-case WAE wae

[num (n) n]

[add (l r) (+ (interp l) (interp r))]

[sub (l r) (- (interp l) (interp r))]))

...

... ))

## 4. Implement WAE interpreter

; interp: WAE -> number

(define (interp wae)

(type-case WAE wae

[num (n) n]

[add (l r) (+ (interp l) (interp r))]

[sub (l r) (- (interp l) (interp r))]))

[with (i v e) ...]

[id (s) ...]))

## 4. Implement WAE interpreter

; interp: WAE -> number

(define (interp wae)

(type-case WAE wae

[num (n) n]

[add (l r) (+ (interp l) (interp r))]

[sub (l r) (- (interp l) (interp r))]))

[with (i v e) ... ]

[id (s) (error 'interp "free identifier")]))

## 4. Implement WAE interpreter

; interp: WAE -> number

(define (interp wae)

(type-case WAE wae

[num (n) n]

[add (l r) (+ (interp l) (interp r))]

[sub (l r) (- (interp l) (interp r))])

[with (i v e) ... (interp v) ... (interp e) ...]

[id (s) (error 'interp "free identifier")])

(test (interp (with 'x (num 5) (add (id 'x) (id 'x)))) 10)



## 4. Implement WAE interpreter

; interp: WAE -> number

(define (interp wae)

(type-case WAE wae

[num (n) n]

[add (l r) (+ (interp l) (interp r))]

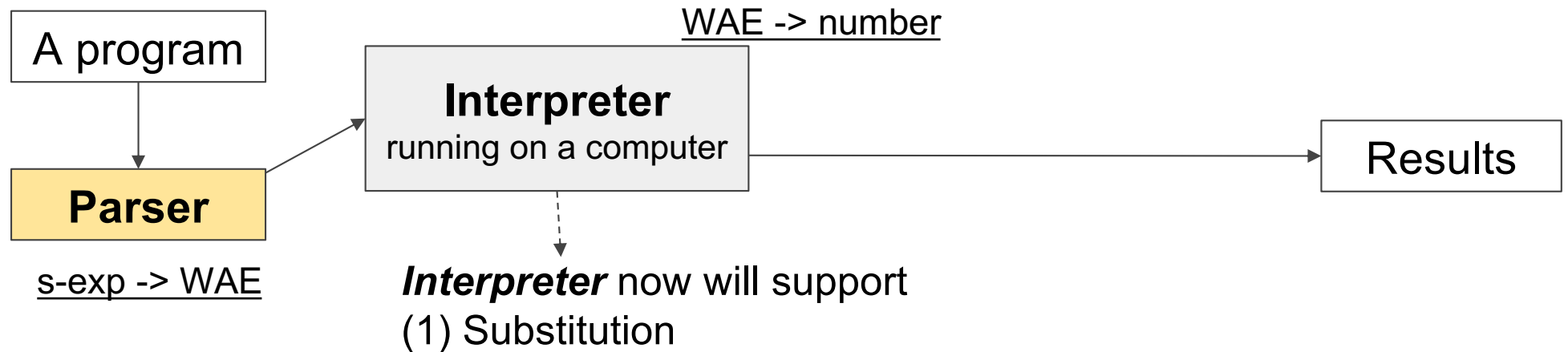
[sub (l r) (- (interp l) (interp r))]

[with (i v e) (interp (subst e i (interp v)))]

[id (s) (error 'interp "free identifier")]))

(test (interp (with 'x (num 5) (add (id 'x) (id 'x)))) 10)

# Big Picture (modeling languages: substitution)



# Topics we cover and schedule (tentative)

- Racket tutorials (L2,3)
- Modeling languages (L4,5)
- Interpreting arithmetic (L5)
- Language principles
  - Substitution (L6-7)
  - Function (L8)
  - Deferring Substitution (L9)
  - First-class Functions (L10-L12)
  - Laziness (L13,14)
  - Recursion (L15,16)
  - Mutable data structures (L17,18,19,20)
  - Variables (L21,22)
  - Continuations (L23-26)
- Guest Video Lecture (L27)

## TODO

Read Chapter 4. An Introduction to Functions

2nd edition:

<http://cs.brown.edu/courses/cs173/2012/book/adding-functions.html>

JC

[jcnam@handong.edu](mailto:jcnam@handong.edu)  
<https://lifove.github.io>

\* Slides are from Prof. Sukeyoung Ryu's PL class in 2018 Spring  
or created by JC based on the main text book.