L6. Substitution

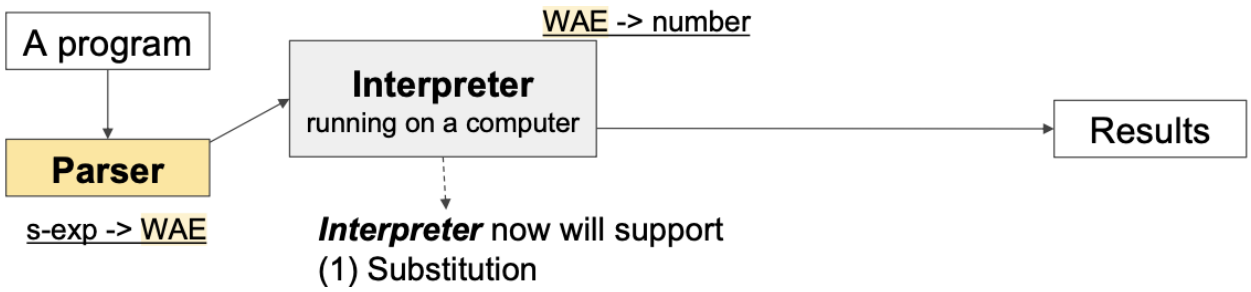
유형	강의
복습 여부	☑
Status	Done

Reminder!

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)

We are extending ‘AE’ to support identifiers!



Why?

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

```
// 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);
```

If we use substitution, we can avoid that redundancy.

```
// 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. **Identifier** : *partialSum*

3. Substitution: To get the result of *totalSum*, *partialSum* needs to be replaced with 55 while computing the total sum.

Identifiers

- Name/identify the value of an expression
- names for the computed constants
- Reuse its name in place of the larger computation to avoid redundancy
- Similar to variables

    - But in variables, the value could change
- 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)

Sample program that uses identifiers

Deal with an expression with an identifier *in our new language*?

1. use an identifier for the repeated expressions
2. We would like to use 'with' keyword to define an identifier for an arithmetic expression and use the identifier for another arithmetic expression.

Q1. How do you write this expression, **{+ {+ 5 5} {+ 5 5}}** with ‘with’?

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

Q2. How do you write this expression, **{+ {- {+ 5 5} 3} {- {+ 5 5} 3}}** with ‘with’?

```
{with {x {+ 5 5}} {+ {- x 3} {- x 3}}}
{with {x {+ 5 5}} {with {y {- x 3}} {+ y y}}}
{with {x 10} {with {y {- x 3}} {+ y y}}}
{with {x 10} {with {y {- 10 3}} {+ y y}}}
{with {y {- 10 3}} {+ y y}}}
{with {y 7} {+ y y}}
{with {y 7} {+ y y}}
{+ 7 7}
14
```

Improve AE to support identifiers

'with' with arithmetic expressions ⇒ WAE in BNF

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

- Identifiers

  <id> ::= x, y, plus, factorial, swap, interp,...

  or (in scheme: a bit different from Racket)

  <id> ::= <initial> <subsequent>* | + | - |

  ...

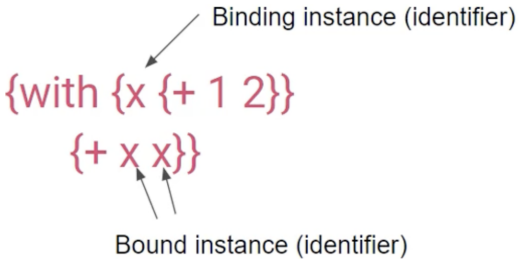
  <initial> ::= <letter> | ! | $ | % | & | * | : | < | = | > | ? | ~ | _ | ^

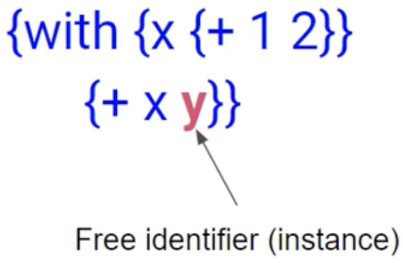
  <subsequent> ::= <initial> | <digit> | . | + | -

  <letter> ::= a | b | ... | z

  <digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

* zero or more occurrences





Practice

Q. What is the result of each expression below?

```
{with {x {+ 1 2}} {+ x x}}
; 6
```

```
{with {x {+ 1 2}} {+ x y}}
; produces error
```

```
{+ {with {x {+ 1 2}}
  {+ x x}}
    {with {x {- 4 3}}
  {+ x x}}}
; 8
```

```
{with {x {+ 1 2}}
  {with {x {- 4 3}}
    {+ x x}}}
; 2
```

```
{with {x {+ 1 2}}
  {with {y {- 4 3}}
    {+ x x}}}
; 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?)])
```

```
; type definition
; Language 'WE' that supports identifier
(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!

```
; [contract] parse : sexp -> WAE
; [purpose] to convert s-expression into WAE
(define (parse sexp)
  (match sexp
    [(? number?) (num sexp)]
    [(list '+ l r) (add (parse l) (parse r))]
    [(list '- l r) (sub (parse l) (parse r))]
    [(list 'with (list i v) e) (with i (parse v) (parse e))]
    [(? symbol?) (id sexp)]
```

```
    [else (error 'parse "bad syntax:~a" sexp)]))

(test (parse '{+ {- 3 4} 7}) (add (sub (num 3) (num 4)) (num 7)))
(test (parse '{with {x 5} {+ 8 2}}) (with 'x (num 5) (add (num 8) (num 2))))
(test (parse '{with {x 5} {+ x x}}) (with 'x (num 5) (add (id 'x) (id 'x))))
```

```
good (parse '(+ (- 3 4) 7)) at line 22
  expected: (add (sub (num 3) (num 4)) (num 7))
  given: (add (sub (num 3) (num 4)) (num 7))

good (parse '(with (x 5) (+ 8 2))) at line 23
  expected: (with 'x (num 5) (add (num 8) (num 2)))
  given: (with 'x (num 5) (add (num 8) (num 2)))

good (parse '(with (x 5) (+ x x))) at line 24
  expected: (with 'x (num 5) (add (id 'x) (id 'x)))
  given: (with 'x (num 5) (add (id 'x) (id 'x)))
```

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***.

```
; i : x
; v : 5
; e : {+ x x}
{with {x 5} {+ x x}}

; Substitution based on Definition 1
```

```
{with {x 5} {+ 5 5}}
; result is 10
```

Q1. How about `{with {x 5} {+ 10 y}}` ?

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

; Substitution based on Definition 1
; No substitutions occur since there is no instances of x in the expression
```

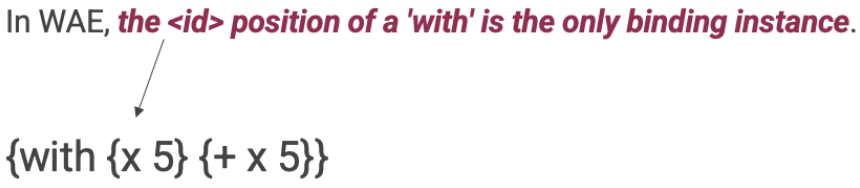
Q2. How about `{with {x 5} {+ x {with {x 3} 10}}}` ?

```
{with {x 5} {+ x {with {x 3} 10}}}

; Substitution based on Definition 1
; {with {x 5} {+ 5 {with {5 3} 10}}}
; Syntactically illegal and our parser will reject this expression!
; Recall WAE in BNF: | {with {<id> <WAE>} <WAE>
```

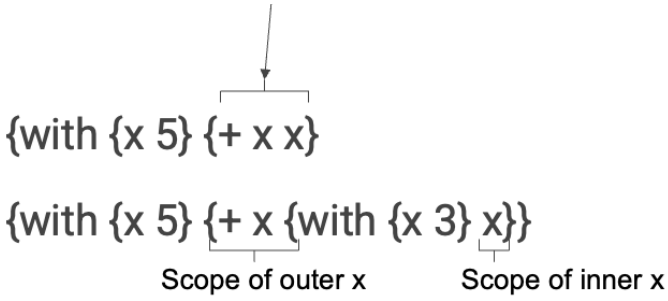
Definition 2 (Binding Instance)

A binding associates an identifier (like a variable) with a specific value or type.



Definition 3 (Scope)

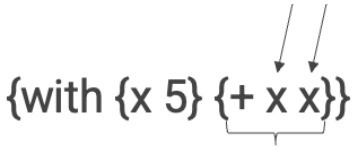
The scope of a binding instance is the region of program text in which instances of the identifier refer to the value bound by the binding instance.



Definition 4 (Bound Instance)

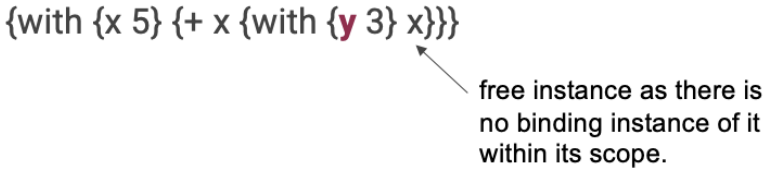
An identifier is bound if it is contained within the scope of a binding instance of its name.

Each bound identifier corresponds to a single binding of that variable.



Definition 5 (Free identifier/Instance)

An identifier not contained in the scope of any binding instance of its name is said to be free.



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***.

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

; Substitution based on Definition 6
```

```
; {with {x 5} {+ 5 {with {x 3} 10}}}
; result is 15
```

Q1. How about `{with {x 5} {+ x {with {x 3} x}}}` ?

```
; i: x
; v: 5
; e: {+ x {with {x 3} x}}
{with {x 5} {+ x {with {x 3} x}}}

; Substitution based on Definition 6
; {with {x 5} {+ 5 {with {x 3} 5}}}
; result is 10
; result should be 8!
```

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***.

```
; i: x
; v: 5
; e: {+ x {with {x 3} x}}
{with {x 5} {+ x {with {x 3} x}}}

; Substitution based on Definition 7
; {with {x 5} {+ 5 {with {x 3} 3}}}
; result is 8
```

Q1. How about `{with {x 5} {+ x {with {y 3} x}}}` ?

```
; i: x
; v: 5
; e: {+ x {with {y 3} x}}
{with {x 5} {+ x {with {y 3} x}}}

; Substitution based on Definition 7
; {with {x 5} {+ 5 {with {y 3} x}}}
; substition cannot be done and produces error
; result should be 10!
```

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***.

```
; i: x
; v: 5
; e: {+ x {with {y 3} x}}
{with {x 5} {+ x {with {y 3} x}}}

; Substitution based on Definition 8
{with {x 5} {+ 5 {with {y 3} 5}}}
; result is 10
```

Final Definition (Substitution)

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***.

```
; i: x
; v: 5
; e: {+ x {with {y 3} x}}
{with {x 5} {+ x {with {y 3} x}}}

; Substitution based on Definition 9
{with {x 5} {+ 5 {with {y 3} 5}}}
; result is 10
```

Q. Find Bound/Binding/Free instances and scopes of binding identifiers

1. {with {x 5} {+ x {with {y x} x}}}

- Bound instances

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

- Binding instances

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

Answer : 10

2. {with {x 5} {+ x {with {x {+ x 1}} x}}}

- Bound instances

```
{with {x 5} {+ x {with {x {+ x 1}} x}}}
```

- Binding instances

```
{with {x 5} {+ x {with {x {+ x 1}} x}}}
```

Answer : 11