

i116: Basic of Programming

8. Assignment calculator: assignments & parse trees

Kazuhiro Ogata, Canh Minh Do

Roadmap

- Assignments
- Parse trees for assignment calculator

Assignments

- The syntax of an assignment (treated as a statement but not an expression) used for the assignment calculator being developed in this course is as follows:
aVariable := anExpression ;
- anExpression may have variables.
For example, $x := x * x + 4 * x + 4 ;$
- If we know the value of variable x , we can calculate the expression $x * x + 4 * x + 4$.
For example, when x is 2, the expression is $2 * 2 + 4 * 2 + 4$, which can be easily calculated.
- But, how do we know the value of variable x ?

Assignments

- Even though we happen to know that the value of variable x is 2, it does not seem reasonable to replace each occurrence of variable x with 2.

$2 := 2 * 2 + 4 * 2 + 4 ;$

- The left-hand side occurrence of variable x should not be replaced with 2.

$x := 2 * 2 + 4 * 2 + 4 ;$

- If we calculate the right-hand side expression, the assignment becomes the following:

$x := 16 ;$

Assignments

$x := 16 ;$

- How should we deal with it?
- So far, we have learned several data structures, such as built-in tuples and lists.
- One data structure makes it possible to keep variable x and know (or maintain) the value of x .
- It suffices to make an **association** between variable x and the value of x .
- We can use **dictionaries** to do so.

Assignments

Suppose that we are given the following dictionary:

$\{\dots, x:2, \dots\}$

Let *env* be the dictionary, where *env* stands for **environment**.

We can ask *env* to know the value of variable *x*.

env[*x*]

Assignments

Therefore, we can calculate the expression

$$x * x + 4 * x + 4$$

basically as follows:

$$env[x] * env[x] + 4 * env[x] + 4$$

Assignments

Then, the assignment

$$x := x * x + 4 * x + 4 ;$$

becomes

$$x := env[x] * env[x] + 4 * env[x] + 4 ;$$

and then becomes

$$x := 16 ;$$

Assignments

How should we handle it?

$x := 16 ;$

env should be updated as follows:

$env[x] = 16$

Then, *env* becomes as follows:


$\{..., x:16, ...\}$

Assignments

→ $x := 2 ;$
 $x := x * x ;$
 $x := x * x ;$
 $x := x * x ;$
 $y := 2 ;$
 $y := y * y ;$
 $x := x * y ;$

{ }

Assignments

$x := 2 ;$
 $x := x * x ;$
 $x := x * x ;$
 $x := x * x ;$
 $y := 2 ;$
 $y := y * y ;$
 $x := x * y ;$

$\{x:2\}$

Assignments

$x := 2 ;$

$x := x * x ;$

 $x := x * x ;$

$x := x * x ;$

$y := 2 ;$

$y := y * y ;$

$x := x * y ;$

$\{x:4\}$

Assignments

$x := 2 ;$

$x := x * x ;$

$x := x * x ;$

 $x := x * x ;$

$y := 2 ;$

$y := y * y ;$

$x := x * y ;$

$\{x:16\}$

Assignments

$x := 2 ;$

$x := x * x ;$

$x := x * x ;$

$x := x * x ;$

→ $y := 2 ;$

$y := y * y ;$

$x := x * y ;$

$\{x:256\}$

Assignments

$x := 2 ;$

$x := x * x ;$

$x := x * x ;$

$x := x * x ;$

$y := 2 ;$

→ $y := y * y ;$

$x := x * y ;$

$\{x:256, y:2\}$

Assignments

$x := 2 ;$

$x := x * x ;$

$x := x * x ;$

$x := x * x ;$

$y := 2 ;$

$y := y * y ;$

 $x := x * y ;$

$\{x:256, y:4\}$

Assignments

```
x := 2 ;  
x := x * x ;  
x := x * x ;  
x := x * x ;  
y := 2 ;  
y := y * y ;  
x := x * y ;
```

{x:1024, y:4}



Parse trees for assignment calculator

- Because variables have been introduced, we need to have parse trees for variables.

```
class VarParseTree(ExpParseTree):  
    name = ""  
    def __init__(self, n):  
        self.name = n  
    def __str__(self):  
        return self.name
```

Var: 'x'

A string is used to represent a variable or a variable name.

Parse trees for assignment calculator

- We keep the other classes for expression parse trees.

```
class NumParseTree(ExpParseTree):  
    ...
```

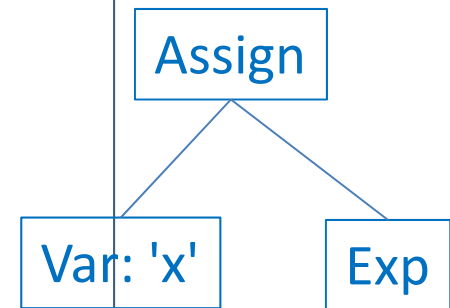
```
class UmiParseTree(ExpParseTree):  
    exp = ExpParseTree()  
    ...
```

```
class AddParseTree(ExpParseTree):  
    exp1 = ExpParseTree()  
    exp2 = ExpParseTree()  
    ...
```

Parse trees for assignment calculator

- Because assignments or assignment statements have been introduced, we need to have parse tree for them.

```
class AssignParseTree(StmParseTree):  
    var = ExpParseTree()  
    exp = ExpParseTree()  
    def __init__(self, v, e):  
        self.var = v  
        self.exp = e  
    def __str__(self):  
        return '(' + self.var.name + ' := ' + str(self.exp) + ' ;' + ')'
```



Parse trees for assignment calculator

```
class StmParseTree(object):  
    def __str__(self):  
        pass
```

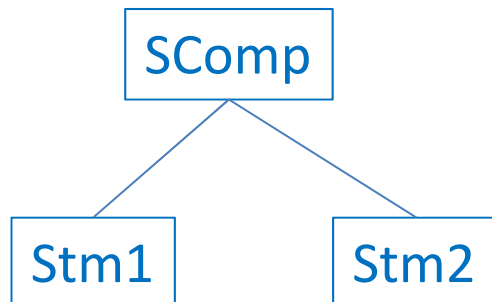
Note that as written already we do not need to have this class, but this class is used like an interface of Java.

Parse trees for assignment calculator

- How should we combine two or more assignments?
- Actually we do not have any means to do so.
- It is necessary to have a language construct that makes it possible to combine two or more statements.
- It is called **sequential compositions** or **sequential composition statements**.

Parse trees for assignment calculator

```
class SCompParseTree(StmParseTree):  
    stm1 = StmParseTree()  
    stm2 = StmParseTree()  
    def __init__(self, s1, s2):  
        self.stm1 = s1  
        self.stm2 = s2  
    def __str__(self):  
        return '(' + str(self.stm1) + " " + str(self.stm2) + ')'
```

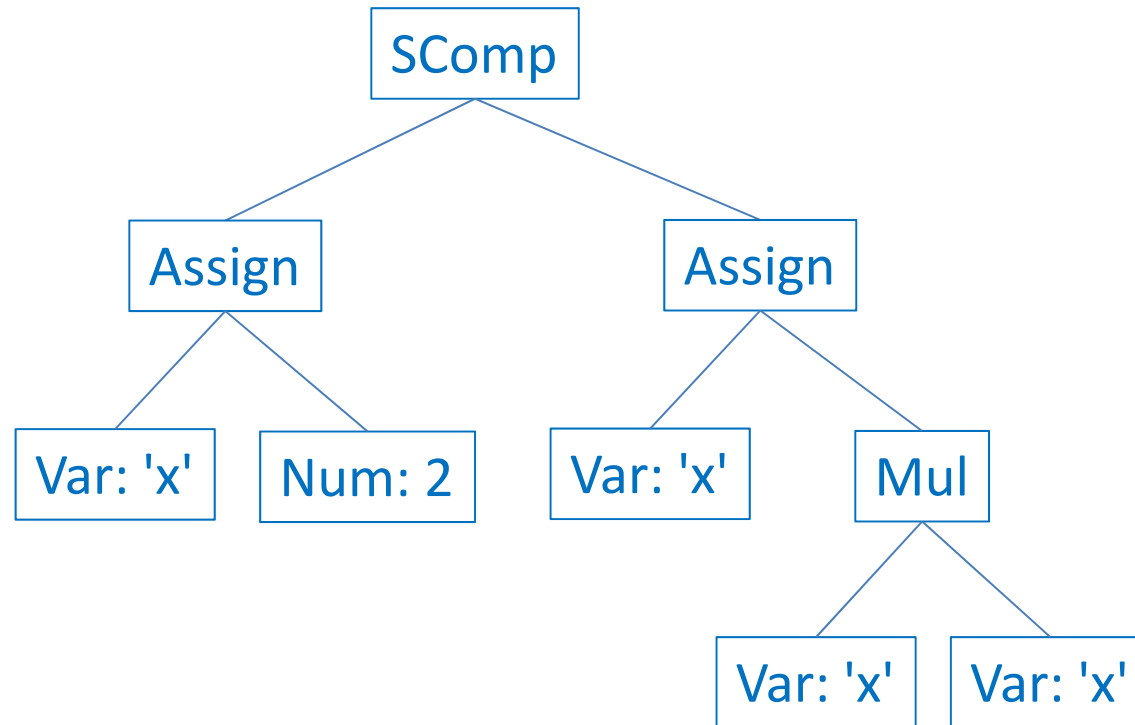


$x := 2 ;$
 $x := x * x ;$

Parse trees for assignment calculator

$x := 2 ;$

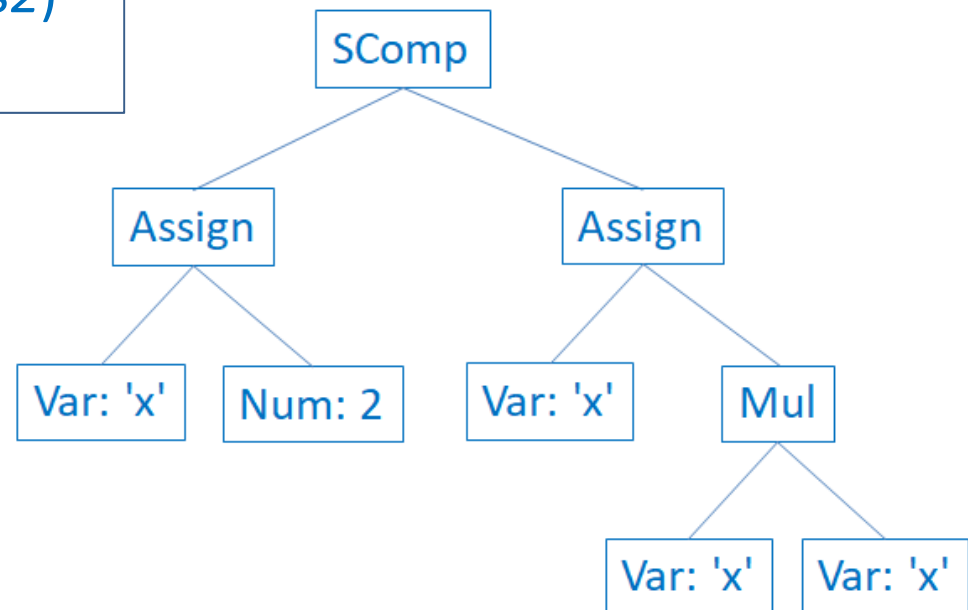
$x := x * x ;$



Parse trees for assignment calculator

```
varX = VarParseTree('x')  
two = NumParseTree(2)  
as1 = AssignParseTree(varX,two)  
e1 = MulParseTree(varX,varX)  
as2 = AssignParseTree(varX,e1)  
pgm1 = SCompParseTree(as1,as2)  
print(pgm1)
```

$x := 2 ;$
 $x := x * x ;$



Parse trees for assignment calculator

```

varX = VarParseTree('x')
varY = VarParseTree('y')
two = NumParseTree(2)
as1 = AssignParseTree(varX,two)
as2 = AssignParseTree(varY,two)
e1 = MulParseTree(varX,varX)
e2 = MulParseTree(varY,varY)
as3 = AssignParseTree(varX,e1)
as4 = AssignParseTree(varY,e2)
e3 = MulParseTree(varX,varY)
as5 = AssignParseTree(varX,e3)
pgm1 = SCompParseTree(as1,as3)
pgm2 = SCompParseTree(pgm1,as3)
pgm3 = SCompParseTree(pgm2,as3)
pgm4 = SCompParseTree(pgm3,as2)
pgm5 = SCompParseTree(pgm4,as4)
pgm6 = SCompParseTree(pgm5,as5)
print(pgm6)

```

$$x := 2 ;$$

$$x := x * x ;$$

$$x := x * x ;$$

$$x := x * x ;$$

$$y := 2 ;$$

$$y := y * y ;$$

$$x := x * y ;$$

Parse trees for assignment calculator

$x := 2 ;$
 $x := x * x ;$
 $x := x * x ;$
 $x := x * x ;$
 $y := 2 ;$
 $y := y * y ;$
 $x := x * y ;$

