

# Homework 4: Semantic Analysis

Instructor : Kiayias

Out Date:

11/06/2007

Due Date:

11/16/2007

---

## Objectives

The purpose of this homework is to design a semantic analyzer for C-. The semantic analysis will be performed over the abstract syntax tree that will be constructed as in the previous homeworks. Each node of the abstract syntax tree in this case will offer a number of methods that will enable the traversal of the tree for the purpose of decorating it with scope information and synthesizing the type information. Each extension of the basic node class will provide the implementation of two methods `buildscopes` and `analyze` that will facilitate the semantic analysis. The analysis will occur in two stages, the scope building stage where the scope information will be propagating downwards and the type synthesis stage where the type information will be propagating upwards.

## Handout

To obtain your assignment, go to the class website in the handouts section and download `hw4-package.zip`. Similar to previous homeworks, this archive is a template project that you can import into Eclipse. It contains support code and basic structure. It is up to you to “fill in” the blanks. For your convenience I have included the C--grammar. The project now includes a new package called `sem` that contains a number of classes all of which extend the following abstract class:

```
package sem;

import parser.SrcLoc;
import ast.ASTtype;

public abstract class Descriptor {
    public boolean isemptyflag;
    public boolean isinheritedflag;
    public boolean isEmpty() { return isemptyflag; }
    public boolean isClass() { return false; }
    public boolean isMethod() { return false; }
    public boolean isVariable() { return false; }
    public boolean isInherited() { return isinheritedflag; }
    public ASTtype type;
    public String name;
    public SrcLoc location;
    public int sizeof() { return 0; }
}
```

These classes are inserted into a symbol table that is maintained by a class `SEMscope`. All these classes are provided.

Inside the package `ast` you will find all abstract syntax tree class that are referenced in `parser.cup`. In most classes the instantiation of the method `buildscopes` and `analyze` is missing; you will have to provide these methods.

In the zip file you will also find the files **test1.cmm**, **test2.cmm** written in C--. Try to catch as many semantic errors as possible. In particular the semantic errors of **test2.cmm** are the following:

Pass 1

```
[SEM]: Method named 'method' in location 20:9 was already defined in location 8:9
[SEM]: Name 'test' in location 37:6 was already bound in current scope in location 26:6
[SEM]: Method named 'method' in location 41:7 was already defined in location 24:7
[SEM]: Extending a non-existent class at location 60:33
[SEM]: Bad class constructor at location 73:8
```

Pass 2

```
[SEM]: Condition of if statement is not boolean at location 12:5
[SEM]: Condition of while loop is not boolean at location 13:7
[SEM]: Error spawning array at location 14:15 expected integer bounds.
[SEM]: Type mismatch at location 16:9 expected int operand.
[SEM]: Return statement at location 17:2 does not return the correct type for enclosing method
[SEM]: Identifier 'c' undeclared at location 34:5
[SEM]: Type mismatch at location 53:4 expected int but received boolean
[SEM]: actuals do not match formals in location 55:26
[SEM]: Given item is not a member of given class at location 56:27
[SEM]: Type mismatch at location 70:4 expected boolean but received int
[SEM]: No such constructor at location 88:26
[SEM]: Class constructor absent at location 90:25
[SEM]: Type mismatch at location 95:9 expected class type but received class type
[SEM]: Unknown type at location 97:13
[SEM]: actuals do not match formals in location 102:13
```

**Remark 1.** Feel free to toss away the grammar I give you and substitute it for the one you have built from homework #3 incorporating the `traverse` pretty printing functionality into the system. Only minor modifications to your code will be required to do this.

**Remark 2.** There are other semantic errors that are not demonstrated by **test2.cmm**. Depending on the difficulty of catching the error, if you find one and you successfully report as well as provide a C--program exhibiting the error, you can receive **extra credit**.

## Hand-in

Your solution should be an archive of the whole project (that should compile correctly and perform the semantic analysis as good as you can) and should include the following:

1. All the classes of the package **ast** appropriately completed.

You submit your homework by midnight on the due date as a zip archive exported from eclipse. The e-mail address to submit is `c244fa07@cse.uconn.edu`.

Have fun!

## test2.cmm

```
1  /* CSE 244 Fall 2006
2  * This is a syntactically correct C-- program
3  * that is full of semantic errors.
4  * Catch them all!
5  */
6
7  class firstclass {
8      boolean method(boolean baz, int j)
9      {
10         int[] bar;
11         boolean test;
12         if (32) { }
13         while(bar) { bar[1] = 1; }
14         bar = new int[1..test];
15         bar[1] = (j + 1)*2;
16         j = bar[test];
17         return 10;
18     }
19
20     boolean method(boolean bazbaz, int i) {}
21
22     int a;
23
24     int[] method()
25     {
26         int test;
27         int t;
28         int b;
29
30         for
31             (t=1; t<10; t++)
32             { a = 1;
33               b = 2;
34               c = 3;
35             }
36
37         int test;
38         test = 1;
39     }
40     boolean method(boolean baz) {}
41     int[] method() {}
42 };
43
44 class secondclass{
45     int a;
46     boolean b;
47     secondclass() { }
48     void method2()
49     {
50         int a;
51         boolean b;
52         firstclass firstinstance;
53         a = firstinstance.method(b,a);
54         b = firstinstance.method(firstinstance.method(b));
```

```

55         b = firstinstance.method(a);
56         a = firstinstance.method2(b,a);
57     }
58 };
59
60 class thirdclass extends myclass {
61     int b;
62 };
63
64 class fourthclass extends secondclass{
65     int c;
66     boolean a;
67     void method3()
68     {
69         a = false;
70         b = 1;
71         method2();
72     }
73     fourthv()
74     {
75         int a;
76     }
77     fourthclass(int b)
78     {
79         b = 1;
80     }
81 };
82
83 class fifthclass{
84     int method4(int a, int b)
85     {
86         fourthclass class1;
87         class1 = new fourthclass(a);
88         class2 = new fourthclass(a,b);
89         firstclass class2;
90         class2 = new firstclass();
91
92         secondclass class3;
93         class3 = new secondclass();
94
95         class3 = class1;
96
97         sixthclass class4;
98     }
99
100     void method5(firstclass foo) {
101         secondclass s;
102         method5(s);
103     }
104 };

```