

## Plan for Today

### What is Semantic Analysis?

- does a syntactically correct program make sense?
- compile time or statically

### Types

- representation
- interpretation
- possible operations

### Type expressions

- atomic types and constructed types

### MiniJava types and type rules

- representation and interpretation will be discussed while doing IRT generation
- what operations are possible on what types?

CS453 Lecture

Semantic Analysis

1

## Dynamic determination of which method is called

```
class A { public void foo() {} }
class B : public A {
    public void foo() { print "I am B"; }
}
class C : public B {
    public void foo() { print "I am C"; }
}

A a;
if (test) {
    a = new B();
} else {
    a = new C();
}
a.foo();
```

CS453 Lecture

Semantic Analysis

2

## Type implementation in the MiniJava compiler

```
public class Type {
    public static final Type ARRAY = new Type();

    public static final Type BOOL = new Type();

    public static final Type INT = new Type();

    // class type map (key: class name, value: type)
    private static final HashMap<String, Type> classTypes
        = new HashMap<String, Type>();
}
```

### Only one instance of the type object per atomic type and class type

- to determine if types are equal just compare references
- does the Type class know about inheritance?

CS453 Lecture

Semantic Analysis

3

## Implementing type checking for MiniJava (Slide 1)

### Visitor over AST will check for type errors at each AST node

	Syntax	AST production AST node
Errors	id = Exp ;	statement = {assign} id exp [LINENUM, POSNUM] Undeclared variable VARNAME [LINENUM, POSNUM] Invalid expression type assigned to variable VARNAME
	id [Exp] = Exp ; statement = {array_assign} id [index]:exp exp	[LINENUM, POSNUM] Undeclared variable VARNAME [LINENUM, POSNUM] Array reference to non-array type [LINENUM, POSNUM] Invalid index expression type for array reference [LINENUM, POSNUM] Invalid expression type assigned into array
	Exp op Exp    exp = {op} [l_exp]:exp [r_exp]:exp	[LINENUM, POSNUM] Invalid left operand type for operator OP [LINENUM, POSNUM] Invalid right operand type for operator OP

CS453 Lecture

Semantic Analysis

4

## Implementing type checking for MiniJava (Slide 2)

Syntax	AST production	AST node
! Exp	exp = {not} exp	[LINENUM,POSNUM] Invalid operand type for operator !
new int [ Exp ]	exp = {new_array} exp	[LINENUM,POSNUM] Invalid operand type for new array operator
Exp [ Exp ]	exp = {array} exp [index]:exp	[LINENUM,POSNUM] Array reference to non-array type [LINENUM,POSNUM] Invalid index expression type for array reference
Exp . length	exp = {length} exp	[LINENUM,POSNUM] Operator length called on non-array type

CS453 Lecture

Semantic Analysis

5

## Implementing type checking for MiniJava (Slide 3)

Syntax	AST production	AST node
new id ( )	exp = {new} id	[LINENUM,POSNUM] Class CLASSNAME does not exist [LINENUM,POSNUM] Undeclared class type in new operator
Exp . id ( ExpList )	exp = {call} exp id [args]:exp*	[LINENUM,POSNUM] Invalid type returned from method METHODNAME [LINENUM,POSNUM] Class CLASSNAME does not exist [LINENUM,POSNUM] Receiver of method call must be a class type [LINENUM,POSNUM] Method METHODNAME does not exist in class type CLASSNAME [LINENUM,POSNUM] Method METHODNAME requires exactly NUM arguments [LINENUM,POSNUM] Invalid argument type for method METHODNAME

CS453 Lecture

Semantic Analysis

6

## Implementation Plan

### Test-driven approach

- Write test cases for
  - one AST node at a time
  - one type check at a time
  - one possible type at a time (start with atomic types)
- Set up a regression testing script
  - capture your compiler output on test case to a temp file
  - compare output to a handwritten output for test case
- Implement
  - one AST node at a time
  - one type check at a time
  - one possible type at a time (start with atomic types)

### Advantages

- turn in your program at any point to get partial credit
- separate two most difficult pieces: understanding MiniJava typing and implementing the typecheck with the provided data structures

CS453 Lecture

Semantic Analysis

7