

Praktikum Compilerbau

Sitzung 5 – Semantische Analyse

Prof. Dr.-Ing. Gregor Snelting
Andreas Zwinkau

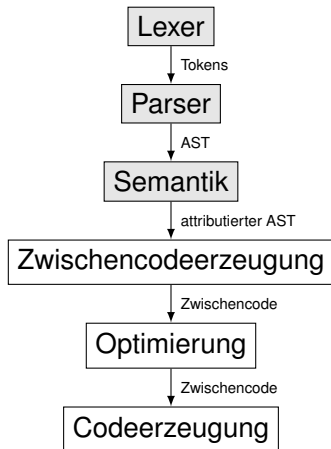
IPD Snelting, Lehrstuhl für Programmierparadigmen



- Abgabe Grammatiken
- Abgabe Attributierte Grammatiken
- Was waren die Probleme?
- Hat soweit alles geklappt?

Probleme:

- Keine ausführbarer Compiler (shell skript)
- `class` \Rightarrow `Class`?!
- Stack trace bei `LexerException: File not found!`
- `NullPointerException`
 - `StatementNode.java:61`
 - `LogicalOrExpressionNode.java:44`
- Terminierung bei viele Ausdrücken?
`OutOfMemoryError`
- Fehlermeldungen (`error`) statt stack traces!



Die semantische Analyse überprüft die Wohlgeformtheit von Programmen. Sie prüft dabei die Dinge die mit der lexikalischen und syntaktischen Analyse nicht möglich waren.

In MiniJava:

- *Namensanalyse*
 - Welche Dinge werden definiert.
 - Welche Definition wird von einer Benutzung referenziert.
- *Typanalyse*
 - Welchen Typ hat ein Ausdruck?
 - Welchen Typ haben benutzerdefinierte Dinge wie Methoden, Felder und Klassen?
 - Passen die Typen zu den Operationen im Programm?

Aufgaben der Namensanalyse – Definition/Benutzung

```
int a = 42;  
int b = a + 3;  
System.out.println(a);
```

- Liste der Definitionen muss leicht zugänglich sein.
- Bei Codeerzeugung: Zuordnung von „Namen“/Definitionen zu Ressourcen.
- Zuordnung Benutzung zu Definition nötig für Codeerzeugung, Analysen und Typprüfung.
- Fehlermeldung bei mehreren Definitionen gleichen Namens (im selben Gültigkeitsbereich).
- Fehlermeldung bei Verwendung eines Namens ohne Definition.

Namensanalyse – Schachtelung, Gültigkeitsbereiche

```
class A {  
  public int foo;  
  public int getFoo() { return foo; }  
  public void setFoo(int foo) {  
    this.foo = foo;  
  }  
  public void bar() {  
    if (true) {  
      int foo = getFoo() + 42;  
      System.out.println(foo);  
    }  
    System.out.println(foo);  
  }  
}
```

Bereiche (Scopes):

- (Pakete)^a
- Klassen
- Methoden, Felder, (innere Klassen)
- Parameter
- Deklaration in Block-Anweisungen, (in for-Anweisung)

^anicht in MiniJava

Namensanalyse – Definition nach Benutzung

```
class A {  
    public void printFoo() {  
        System.out.println(foo);  
        bar();  
    }  
    public int foo;  
    public void bar() { }  
}
```

- In einigen Bereichen (Pakete, Klassen, Methoden, Felder) dürfen Bezeichner vor ihrer Definition benutzt werden.
- Dies ist der Hauptgrund warum semantische Analyse in Java/MiniJava einen eigenen Durchlauf benötigt

Namensanalyse – Disjunkte Namensräume

```
public class foo {  
    public foo foo();  
    public foo foo(int x) {  
        foo = foo(0);  
        foo: while (true) {  
            int foo = 42;  
            while (true) {  
                foo(foo).foo = (foo) null;  
                break foo;  
            }  
        }  
        return this;  
    }  
}
```

Unterscheidung Namensräume
anhand des syntaktischen
Kontexts:

- Typen
- Methoden
- Felder
- Marken

Namensanalyse – Referenzierung fremder Namensräume

```
class B {  
    public int foo;  
}  
class A {  
    public B b;  
    public int getFoo() {  
        return b.foo;  
    }  
}
```

Namensanalyse – Pakete

```
class A {  
    public void foo() {  
        System.out.println(42);  
    }  
}  
  
class B {  
    public B System;  
    public B out;  
    public void println(int x) {  
    }  
    public void foo() {  
        System.out.println(42);  
    }  
}
```

- Gemeinsame Basisklasse (`Entity`) für Dinge die mit Namen referenziert werden können.
- Stringtabelle für effizientes Vergleichen benutzen.
- Namenstabelle für effizientes Finden benutzen.

Stringtabelle

```
class Symbol {  
    String getString();  
}
```

```
public interface StringTable {  
    Symbol findOrInsert(String string);  
}
```

Namenstabelle

```
public interface NameTable {  
    void enterScope();  
    void leaveScope();  
  
    void enterDefinition(Symbol symbol, Definition definition);  
    Definition getCurrentDefinition(Symbol symbol);  
    boolean definitionInCurrentScope(Symbol symbol);  
}
```

```
public class SemanticAnalysis {  
    /* shortcuts for NameTable */  
    Definition currentMethod;  
    Definition currentType;  
    Definition currentVariable;  
}
```

Beachte: Zugriff auf fremde Namensräume nicht über Namenstabelle.

- Wieviele Durchläufe über den AST sind nötig?
- Besucher-Muster oder einfacher rekursiver Abstieg?

- Was wird typisiert? (Ausdrücke, Methoden, Felder)
- Was für Typen gibt es?
 - Atomare Typen: Primitive (**int**, **boolean**, **void?**), Referenz auf X
 - Zusammengesetzte Typen: Array von X, Methodentyp?, Klassentyp?

Ausdrücke Typisieren

- In MiniJava: Jede Operation/jedes Literal hat festen Typ.
- In Java: Operationen und überladene Methoden werden erst durch Typ der Parameter eindeutig bestimmt.
(Gibt es das auch in MiniJava?)

Beispiele:

- **INTEGER_LITERAL** : *int*
- **IDENTIFIER** Typ hängt von Definition ab
- **+** : $int \times int \rightarrow int$
- **&&** : $boolean \times boolean \rightarrow boolean$
- **<** : $int \times int \rightarrow boolean$
- **new X()** : *Reference(X)*

Typen der Operanden müssen zu Operationen passen.

- Bedingungen `if(x)`, `while(x)`: $\text{Typ}(x) == \text{Boolean}$
- Methodenaufrufe $m(p_1, \dots, p_n)$: m hat n Parameter und jeder Ausdruck p_i hat den Typ des entsprechenden Parameters
- Return Statements `return x`: $\text{Typ}(x)$ passt zur Methode.

- Zuweisungen nur an bestimmte Ausdrücke erlaubt (sogenannte l-values). Welche sind das?
- this-Zeiger in static Methoden nicht erlaubt
- Auf jedem Pfad von Anfang zum Ende der Methode gibt es eine **return**-Anweisung. Ausnahme Rückgabebetyp **void**.
- **void** nur als Rückgabebetyp für Methoden zulässig.
- Was ist mit **null**?
- Nicht in MiniJava:
 - Vor jeder Benutzung einer lokalen Variablen muss es eine Definition geben.
 - Es darf kein unerreichbarer Code existieren

Wette

Schickt uns bis zum 20.11. euren Compiler, der die Optionen `--print-ast` und `--check` so wie auf den Übungsblättern angegeben unterstützt. Wir werden in jedem Compiler einen Bug finden!

Feedback! Fragen? Probleme?

- Nächstes Mal: Laptop mitbringen (Firm wird installiert)
- Anmerkungen?
- Probleme?
- Fragen?