# Semantische Analyse

Lukas Ruppert

February 18, 2016

**Abstract**

Die Semantische Analyse überprüft die gegebene abstrakte Syntax auf semantische Fehler und typisiert zudem jeden einzelnen Ausdruck und jedes Statement.

# Contents

# 1 Hauptfunktionen der SemantikCheck.hs

## 1.1 typecheckExpr - Analyse und Typisierung eines Ausdrucks

```
typecheckExpr :: Expr -> [(Type, String)] -> [Class] -> Expr
```

typecheckExpr typisiert einen einzelnen Ausdruck unter Angabe der sichtbaren lokalen Variablen, sowie der Liste aller Klassen. Als Rückgabe erhält man stets einen Ausdruck der Form TypedExpr(rekursiv typisierter Ausdruck, Typ des Ausdrucks)

## 1.2 typecheckStmt - Analyse und Typisierung eines Statements

```
typecheckStmt :: Stmt -> [(Type, String)] -> [Class] -> Stmt
```

typecheckStmt typisiert ein einzelnes Statement unter Angabe der sichtbaren lokalen Variablen, sowie der Liste aller Klassen. Als Rückgabe erhält man stets ein Statement der Form TypedStmt(rekursiv typisiertes Statement, Typ des Statements)

## 1.3 typecheckStmtExpr - Analyse und Typisierung eines Statement-Ausdrucks

```
typecheckStmtExpr :: StmtExpr -> [(Type, String)] -> [Class] -> StmtExpr
```

typecheckStmtExpr typisiert einen einzelnen Statement-Ausdruck unter Angabe der sichtbaren lokalen Variablen, sowie der Liste aller Klassen. Als Rückgabe erhält man stets einen Statement-Ausdruck der Form TypedStmtExpr(rekursiv typisierter Statement-Ausdruck, Typ des Statement-Ausdrucks)

## 1.4 typecheckFieldDecls - Typprüfung von Feld-Deklarationen

```
typecheckFieldDecls :: [FieldDecl] -> [Class] -> Bool
```

typecheckFieldDecls überprüft für jede Feld-Deklaration unter Angabe der Liste aller Klassen, ob die angegebenen Typen existieren.

## 1.5 typecheckMethod - Analyse und Typisierung einer Methode

```
typecheckMethod :: MethodDecl -> Type -> [Class] -> MethodDecl
```

typecheckMethod typisiert alle Statements einer gegebenen Methode unter Angabe des Klassennamens, aus dem die Methode stammt, sowie der Liste aller Klassen. Zusätzlich zur Typisierung der Methoden-Statements wird auch überprüft, ob die Argument-Typen definiert sind und ob der angegebene Rückgabetyp mit dem aus den Statements ausgewerteten Typ übereinstimmt. Als Ergebnis erhält man die Methode mit den typisierten Statements zurück.

## 1.6 typecheckClass - Analyse und Typisierung einer Klasse

```
typecheckClass :: Class -> [Class] -> Class
```

typecheckClass analysiert und typisiert eine Klasse unter Angabe der Liste aller Klassen. Zunächst wird sichergestellt, dass alle angegebenen Superklassen existieren und dass nicht rekursiv abgeleitet wird. Dann werden die Typen aller Feld-Deklarationen mittels typecheckFieldDecls überprüft. Zum Schluss werden alle Methoden mittels typecheckMethod analysiert und typisiert. Als Ergebnis erhält man die Klasse mit den typisierten Methoden.

## 1.7 typecheckPrg - Analyse und Typisierung eines Programms

```
typecheckPrg :: Prg -> Prg
```

typecheckPrg analysiert und typisiert die Liste aller Klassen und gibt als Ergebnis die Liste aller typisierten Klassen zurück.

# 2 Hilfsfunktionen der SemantikCheck.hs

## 2.1 typecheckListOfExpr - Analyse und Typisierung einer Liste von Ausdrücken

```
typecheckListOfExpr :: [Expr] -> [(Type, String)] -> [Class] -> [Expr]
```

typecheckListOfExpr ruft unter Angabe der lokalen Variablen und der Liste aller Klassen typecheckExpr für eine Liste von Ausdrücken auf.

## 2.2 typeListUpperBound - Obermenge einer Liste von Typen

```
typeListUpperBound :: [Type] -> [Class] -> Type
```

typeListUpperBound liefert unter Angabe der Liste aller Klassen die Obermenge einer Liste von Typen.

## 2.3 typeUpperBound - Obermenge zweier Typem

```
typeUpperBound :: Type -> Type -> [Class] -> Type
```

typeUpperBound liefert unter Angabe der Liste aller Klassen die Obermenge zweier Typen.

## 2.4 getTypeOfBinary - Typ eines Ausdrucks mit zwei Argumenten

```
getTypeOfBinary :: String -> Type -> Type -> [Class] -> Type
```

getTypeOfBinary liefert unter Angabe der Liste aller Klassen den Ergebnistyp einer gegebenen Operation, angewandt auf zwei gegebene Typen.

## 2.5 getMaybeClass - Suchen einer Klasse

```
getMaybeClass :: Type -> [Class] -> Maybe Class
```

getMaybeClass liefert zu einem gegebenen Typ unter Angabe der Liste aller Klassen entweder die gesuchte Klasse oder Nichts, falls die gesuchte Klasse nicht definiert ist.

## 2.6 getMaybeFieldDecl - Suchen einer Feld-Deklaration

```
getMaybeFieldDecl :: String -> [FieldDecl] -> Maybe FieldDecl
```

getMaybeFieldDecl liefert zu einem gegebenen Feld-Namen unter Angabe der Liste aller Felder der Klasse entweder das gesuchte Feld oder Nichts, falls das gesuchte Feld nicht definiert ist.

## 2.7 getMaybeMethodDecl - Suchen einer Methode

```
getMaybeMethodDecl :: String -> [MethodDecl] -> Maybe MethodDecl
```

getMaybeMethodDecl liefert zu einem gegenenen Methodennamen unter Angabe der Liste aller Methoden der Klasse entweder die gesuchte Methode oder Nichts, falls die gesuchte Methode nicht definiert ist.

## 2.8 getMaybeLocalVar - Suchen einer Lokalen Variablen

```
getMaybeLocalVar :: String -> [(Type, String)] -> Maybe (Type, String)
```

getMaybeLocalVar liefert zum gegebenen Namen einer lokalen Variablen unter Angabe aller lokalen Variablen entweder die gesuchte lokale Variable oder Nichts, falls die gesuchte lokale Variable nicht exisitert.

## 2.9 fromJustOrError - Erhalten eines Wertes oder Ausgabe einer Fehlermeldung

```
fromJustOrError :: Maybe a -> String -> a
```

fromJustOrError liefert entweder den in einem Maybe gekapselten Wert oder gibt die angegebene Fehlermeldung aus.

## 2.10 getTypeFromLocalVar - Typ einer Lokalen Variablen

```
getTypeFromLocalVar :: (Type, String) -> Type
```

getTypeFromLocalVar liefert den Typ einer gegebenen lokalen Variablen.

## 2.11 getTypeFromFieldDecl - Typ einer Feld-Deklaration

```
getTypeFromFieldDecl :: FieldDecl -> Type
```

getTypeFromFieldDecl liefert den Typ einer gegebenen Feld-Deklaration.

## 2.12 getTypeFromClass - Typ einer Klasse

```
getTypeFromClass :: Class -> Type
```

getTypeFromClass liefert den Typ einer gegebenen Klasse.

## 2.13 getClasses - Suchen einer Liste von Klassen

```
getClasses :: [Type] -> [Class] -> [Class]
```

getClasses liefert zu einer Liste von Typen unter Angabe der Liste aller Klassen entweder die Liste aller gesuchten Klassen oder gibt eine Fehlermeldung aus, falls eine gesuchte Klasse nicht definiert ist.

## 2.14 getClassesIncludingSuperClasses - Suchen einer Liste von Klassen inklusive Superklassen

```
getClassesIncludingSuperClasses :: [Type] -> [Class] -> [Class]
```

getClassesIncludingSuperClasses liefert zu einer Liste von Typen unter Angabe aller Klassen entweder die Liste aller gesuchten Klassen sowie ihre Superklassen oder gibt eine Fehlermeldung aus, falls eine gesuchte Klasse oder Superklasse nicht definiert ist.

## 2.15 getFieldDeclsFromClass - Feld-Deklarationen einer Klasse

```
getFieldDeclsFromClass :: Class -> [Class] -> [FieldDecl]
```

getFieldDeclsFromClass liefert zu einer gegebenen Klasse unter Angabe der Liste aller Klassen alle Feld-Deklarationen der Klasse sowie ihrer Superklassen.

## 2.16 getMethodDeclsFromClass - Methoden einer Klasse

```
getMethodDeclsFromClass :: Class -> [Class] -> [MethodDecl]
```

getFieldDeclsFromClass liefert zu einer gegebenen Klasse unter Angabe der Liste aller Klassen alle Methoden der Klasse sowie ihrer Superklassen.

## 2.17 getConstructorFromClass - Konstruktor einer Klasse

```
getConstructorFromClass :: Class -> MethodDecl
```

getConstructorFromClass liefert zu einer gegebenen Klasse entweder den dort definierten Konstruktor oder den Standardkonstruktor der Klasse.

## 2.18 getSuperClassTypesFromClass - Liste der Superklassen

```
getSuperClassTypesFromClass :: Class -> [Type]
```

getSuperClassTypesFromClass liefert die Liste aller in der Klasse angegeben Superklassen. (In Java maximal eine.)

## 2.19 expandListOfSuperClasses - Superklassen von Superklassen

```
expandListOfSuperClasses :: [Type] -> [Class] -> [Type]
```

expandListOfSuperClasses liefert zu einer Liste von Superklassen unter Angabe der Liste aller Klassen die Liste der Superklassen inklusive aller ihrer Superklassen.

## 2.20 expandListOfSuperClassesHelper - Hilfsfunktion für expandListOfSuperClasses

```
expandListOfSuperClassesHelper:: [Type] -> [Type] -> [Class] -> [Type]
```

expandListOfSuperClassesHelper erhält zusätzlich zu den in expandListOfSuperClasses gegebenen Argumenten eine zweite Typliste, welche die bereits enthaltenen Klassen auflistet, um so rekursive Ableitungen zu erkennen.

## 2.21 getTypeFromMethodDecl - Rückgabetyp einer Methode

```
getTypeFromMethodDecl :: MethodDecl -> Type
```

getTypeFromMethodDecl liefert den angegebenen Rückgabetyp einer gegebenen Methode.

## 2.22 getArgsFromMethodDecl - Parameter einer Methode

```
getArgsFromMethodDecl :: MethodDecl -> [(Type, String)]
```

getArgsFromMethodDecl liefert die Parameter einer angegebenen Methode.

## 2.23 isSubtypeOf - Subtypenrelation

```
isSubtypeOf :: Type -> Type -> [Class] -> Bool
```

isSubtypeOf bestimmt unter Angabe der Liste aller Klassen, ob der erste gegebene Typ ein Subtyp des zweiten gegebenen Typs ist.

## 2.24 getTypeNameFromType - Typ als String

```
getTypeNameFromType :: Type -> String
```

getTypeNameFromType liefert zu einem gegebenen Typ den Typnamen als String.

## 2.25 typeExists - Existenz eines Typs

```
typeExists :: Type -> [Class] -> Bool
```

typeExists überprüft unter Angabe der Liste aller Klassen, ob ein gegebener Typ exisitert.