Christian-Albrechts-Universität zu Kiel
AG Programmiersprachen und Übersetzerkonstruktion
Prof. Dr. M. Hanus

Christian-Albrechts-Platz 4
24118 Kiel
Tel.: 0431 / 880-7271

5. Übung zur Vorlesung „Prinzipien von Programmiersprachen“

Wintersemester 2008/2009

Abgabe: 9. Dezember 2008 in der Vorlesung

Aufgabe 17 (Präsenzaufgabe)

In der Vorlesung wurde das Konzept der Module vorgestellt. Geben Sie Inferenzregeln für die Deklaration und die Einbindung von Modulen an. Hierbei habe eine Moduldeklaration die Form:

```
module m (i1,...,in) {
  d1;...;dm;
}
```

Dabei bezeichnen d_1,...,d_m die Deklarationen des Moduls (Variablen-, Prozedur- und Funktionsdeklarationen) und i_1,...,i_n sind die Identifikatoren der Deklarationen, die die Schnittstelle des Moduls bilden. Nur diese Deklarationen sind von außerhalb des Moduls zugreifbar.
Die Einbindung eines Moduls erfolgt mit der Deklaration `import` m. Diese Deklaration fügt die Deklarationen aus der Schnittstelle zu der Umgebung des Programms hinzu.

Aufgabe 18

In der Vorlesung wurde die Relation $\overset{R}{\vdash}$ zur Herleitung des Wertes eines Ausdrucks definiert. Diese Relation wurde um Funktionsaufrufe erweitert. Dabei wurde jedoch vorausgesetzt, dass der Speicher M durch die Funktionsaufrufe nicht verändert wird, d.h. Funktionen dürfen keine Seiteneffekte haben.

Geben Sie nun eine Ableitungsrelation $\langle E, M\rangle \overset{RM}{\vdash} e : v, M'$ an, bei der e ein Ausdruck, v ein Wert und M' der möglicherweise veränderte Speicher ist. Die Umgebung E soll während der Auswertung nicht verändert werden. Für einen Ausdruck e sind folgende Fälle zu betrachten:

$$
\begin{array}{rcll}
e & = & x & \texttt{\% Variable} \\
 & \mid & c & \texttt{\% Konstante} \\
 & \mid & e_1 \otimes e_2 & \texttt{\% } \otimes \in \{\texttt{+},\texttt{-},\texttt{*},\texttt{/}\} \\
 & \mid & \texttt{if (}B\texttt{) then } e_1 \texttt{ else } e_2 & \\
 & \mid & f\texttt{(}e_1,\ldots,e_n\texttt{)} &
\end{array}
$$

Wie ist die Regel für die Zuweisung zu modifizieren?

Aufgabe 19
Implementieren Sie eine Klasse `Facloop`. Diese Klasse soll in einer Schleife immer wieder folgende Schritte durchführen: einen String einlesen, diesen String in eine ganze Zahl umwandeln, die Fakultät dieser Zahl berechnen und ausgeben. Falls der Benutzer `quit` eingibt, so soll die Schleife verlassen werden. Das Programm soll folgende Struktur besitzen:

```
import java.io.*;
public class Facloop {
  public static void main( String[] args ) throws IOException {
    BufferedReader in
      = new BufferedReader( new InputStreamReader( System.in ) );

    while ( true ) { ... }
  }

  public static int factorial( int x ) {
    if ( x < 0 ) throw new IllegalArgumentException
                          ( "Argument darf nicht negativ sein!" );
    ...
  }
}
```

Das Objekt `in` besitzt dann eine Methode `readLine()`, die einen String von der Eingabe liest. Der Methodenaufruf `Integer.parseInt( eingabe )` wandelt den String `eingabe` in eine ganze Zahl um. Falls bei der Umwandlung in eine ganze Zahl oder der Berechnung der Fakultät eine Ausnahme auftritt, so soll diese durch eine `try/catch`-Konstruktion abgefangen werden und eine generische Fehlermeldung (`Ungueltige Eingabe: ...`) ausgegeben werden.

Aufgabe 19
Eine Methodendeklaration sei in der Umgebung in der folgenden Form eingetragen:

$$m : method(x_1 : \tau_1, \ldots, x_n : \tau_n, \tau, S)$$

Hierbei sind $x_i : \tau_i$ die Parameter, τ der Typ des Rückgabewertes und S der Rumpf der Methode. Die Deklarationsumgebung wird für die Methodendeklaration nicht benötigt, da sie bereits in der Klassendeklaration abgelegt wurde.
Geben Sie eine Ableitung von $\langle E, M\rangle$ `y = x.m();` $\langle E, M'\rangle$ an, wobei

- $E = E';\ x : (ref2, C);\ y : (ref3, \texttt{double})$
- $E' = C : class\{\texttt{double}\ x,\ \texttt{double}\ y,\ m : method(\emptyset, \texttt{double}, S), \emptyset\}$;
- $S = m$ `= x*x+y*y;`
- $M = \{ref2 \mapsto ref4,\ ref3 \mapsto 0.0,\ ref4 \mapsto 3.0,\ ref5 \mapsto 5.5\}$.

Wie sieht M' nach der Abarbeitung von `y = x.m();` aus? Die Herleitungen für $\overset{lookup}{\vdash}$ und $\overset{L}{\vdash}$ können weggelassen werden.