Technische Universität München
Fakultät für Informatik

A. Berlea
berlea@in.tum.de
Raum: MI 02.07.053

Programmiersprachen

Wintersemester 2006/2007

1. Übungsblatt *2. November 2005*

Aufgabe 1:

Schreibe ein SML-Programm, mit dem man arithmetische Ausdrücke auswerten kann.

a) Definiere zunächst einen Typ `simple_expr`, der *konstante* Ausdrücke beschreibt, d.h. diese Ausdrücke können die Grundrechenarten („+“, „−“, „∗“ und „/“) und `int` Zahlen enthalten. Außerdem benötigt wird eine Funktion

```
simple_eval: simple_expr -> int,
```

die deine Ausdrücke auswertet.

b) Definiere nun einen Typ `expr` mit einer entsprechenden Funktion `eval`, der zusätzlich auch Variablen enthalten kann. Man allerdings für die Aufgabe davon ausgehen, dass nur die Variablen x, y und z vorkommen können.

Bedenke, dass man nun zum Auswerten eine *Variablen-Umgebung* ρ benötigt, in der die aktuellen Werte der drei Variablen nachgeschlagen werden können. (Tip: Man kann ρ als Funktion an `eval` übergeben!)

Aufgabe 2:

Erweitere das Programm aus Aufgabe 1 um Statements. Gehe dabei wie folgt vor:

a) Definiere zunächst eine Funktion `update`, die als Parameter eine *Variablen-Umgebung* ρ, eine Variable v und einen Integer-Wert x erhält. Zurückliefern soll diese Funktion eine *Variablen-Umgebung* ρ', die wie folgt definiert ist:

$$\rho'(v') = \begin{cases} x & \text{falls } v' = v \\ \rho(v') & \text{sonst} \end{cases}$$

b) Definiere ähnlich zur Aufgabe 1 einen Type `bool_expr` für boolsche Ausdrücke und eine Funktion `eval_bool` zur Auswertung solcher Ausdrücke. Unterstütze dabei die Operatoren $\wedge$ (`And`) und $\neq$ (`Not`) sowie die Relationen $=$ (`Eq`) und $<$ (`Lt`). Beachte, dass boolsche Ausdrücke Variablen enthalten können. Die Funktion `eval_bool` erhält also als Parameter eine *Variablen-Umgebung* und einen boolschen Ausdruck.

c) Definiere einen Typ `stmt` für Java-Statements. Beschränken dich dabei auf Zuweisungen (`Assign`), bedingte Verzweigung (`If`), `while`-Schleifen (`While`) sowie auf Blöcke (`Block`). Ein Block ist dabei eine Folge von Statements, die in Java mit `{` eingeleitet und mit `}` abgeschlossen wird.

d) Definiere schließlich eine Funktion `run`, die Java-Statements ausführt. Diese erhält als Parameter ein Statement sowie eine *Variablen-Umgebung* und liefert eine *Variablen-Umgebung* zurück. Der Rückgabewert entspricht der *Variablen-Umgebung* nach Ausführung des Java-Statements.

e) Teste den Java-Interpreter anhand des folgenden Statements:

```
while (x < 10000)
{
  x = x + 1;
  y = y + x;
}
```