Programmierung

09. Übungsblatt

Zeitraum: 22. – 26. Juni 2015

Übung 1 (AGS 13.1 ⋆)

Gegeben sei folgendes C_0-Programm *Max*:

```
#include <stdio.h>

int main() {
  int a, b, max;
  scanf("%i", &a);
  scanf("%i", &b);
  if (a > b) max = a;
  else max = b;
  printf("%d", max);
  return 0;
}
```

(a) Berechnen Sie schrittweise das baumstrukturierte Programm $bMax_0 = trans(Max)$ mit Hilfe der in der Vorlesung angegebenen Übersetzungsfunktionen.

(b) Wandeln Sie $bMax_0$ in ein Programm Max_0 mit linearisierten Adressen um und berechnen Sie $\mathcal{P}[\![Max_0]\!](5:7)$. Dokumentieren Sie den Zustand der AM_0 nach Ausführung jedes Befehls.

Übung 2 (AGS 13.8)

(a) Geben Sie für das folgende C_0-Programm die bereits linearisierte Übersetzung an. Zwischenschritte der Übersetzung brauchen Sie nicht anzugeben. Schreiben Sie je Zeile nur einen Befehl.

```
#include <stdio.h>

int main() {
  int a, b, sum;
  sum = 0;
  scanf("%i", &a);
  if (a < 0)
    a = 0;
  while (a > 0) {
    scanf("%i", &b);
    sum = sum + b;
    a = a - 1;
  }
  printf("%d", sum);
  return 0;
}
```

(b) Folgender Ausschnitt aus einem AM_0-Programm sei gegeben:

9:	`...`	15:	`JMC 23;`	21:	`JMP 16;`
10:	`LOAD 1;`	16:	`LIT 0;`	22:	`JMP 24;`
11:	`LOAD 2;`	17:	`LOAD 1;`	23:	`WRITE 1;`
12:	`LOAD 3;`	18:	`GT;`	24:	`...`
13:	`ADD;`	19:	`JMC 22;`		
14:	`LE;`	20:	`WRITE 2;`		

Lassen Sie dieses Programm auf der AM_0 mit der Anfangskonfiguration

$$(13, 0:1:1, [1/1, 2/1, 3/0], \varepsilon, \varepsilon)$$

schrittweise ablaufen bis der Befehlszähler größer bzw. gleich 24 ist.

(c) Geben Sie für den unter (b) gegebenen Ausschnitt aus einem AM_0-Programm die zugehörigen C_0-Statements an, deren Übersetzung (bis auf eine eventuelle Verschiebung der Befehlsadressen) zu dieser AM_0-Befehlsfolge führt. Vergeben Sie dabei für den Speicherplatz i die Variable `xi`.

Übung 3

Gegeben seien die folgenden Definitionen.

```haskell
data Tree a = Node a (Tree a) (Tree a) | Leaf a

map :: (a->b) -> [a] -> [b]
map f []     = []
map f (x:xs) = f x : map f xs

tmap :: (a->b) -> Tree a -> Tree b
tmap f (Leaf x)     = Leaf (f x)
tmap f (Node x l r) = Node (f x) (tmap f l) (tmap f r)

inorder :: Tree a -> [a]
inorder (Leaf x)      = [x]
inorder (Node x l r ) = inorder l ++ [x] ++ inorder r
```

Beweisen Sie durch strukturelle Induktion, dass für alle Typen `a` und `b`, alle Bäume `t :: Tree a` und alle Funktionen `f :: a -> b` die Eigenschaft

```
map f (inorder t) = inorder (tmap f t)
```

gilt. Quantifizieren Sie alle Variablen, geben Sie in jedem Schritt die verwendete definierende Gleichung, Hilfsaussage, bzw. die Induktionshypothese an. Sie dürfen die folgende Eigenschaft (M) als bewiesen voraussetzen: für alle Typen `a` und `b`, alle Funktionen `f :: a -> b` und Listen `xs , ys :: [a]` gilt:

```
map f xs ++ map f ys = map f (xs ++ ys)
```

Zusatzaufgabe 1 (AGS 13.10 (a))
Geben Sie für das folgende C_0-Programm die Übersetzung in ein linearisiertes AM_0-Programm an. Zwischenschritte der Übersetzung brauchen Sie nicht anzugeben. Schreiben Sie je Zeile nur einen Befehl auf.

```
#include <stdio.h>

int main() {
  int x, y, a;
  scanf("%i", &y);
  scanf("%i", &a);
  x = 0;
  while (x < a) {
    x = x + 1;
    y = y * y;
  }
  printf("%d", y);
  return 0;
}
```

Zusatzaufgabe 2 (AGS 12.4.4 ⋆)

(a) Gegeben sei folgender λ-Term:

$$(x\ (\lambda yz.x\ z)(\lambda xy.z\ y\ x))\ ((\lambda yx.x\ y\ z)\ (\lambda y.x\ z))$$

Reduzieren Sie diesen Term bis seine Normalform erreicht ist. Schreiben Sie – bevor Sie einen Ableitungsschritt ausführen – für die relevanten (Teil-)Ausdrücke die Mengen der freien bzw. der gebundenen Vorkommen von Variablen auf.

(b) Überprüfen Sie, ob sich die folgende Applikation auf $\langle true\rangle$ reduzieren lässt:

$$\langle iszero\rangle\langle false\rangle$$

Benutzen Sie die folgenden Beziehungen:

$$\langle iszero\rangle = (\lambda k.k(\langle true\rangle\langle false\rangle)\langle true\rangle)$$
$$\langle true\rangle = (\lambda xy.x)\,, \quad \langle false\rangle = (\lambda xy.y)$$

(c) Gegeben sei:

$$\begin{aligned}\langle G\rangle = (&\lambda gxy.\langle ite\rangle(\langle iszero\rangle(\langle pred\rangle y))\\ &(\langle add\rangle x\langle 3\rangle)\\ &(g\ (\langle succ\rangle x)\ (\langle pred\rangle y)))\end{aligned}$$

Berechnen Sie $\langle Y\rangle\langle G\rangle\langle 4\rangle\langle 2\rangle$. Führen Sie im Rechenprozess zweckmäßige Abkürzungen der λ–Terme ein.

(d) Eine Funktion $f : \mathbb{N}^+ \times \mathbb{N}^+ \to \mathbb{N}^+$ sei wie folgt definiert:

$$\begin{aligned} f(x,y) &= x\cdot y && \text{für } y = 1\\ f(x,y) &= f(3\cdot x, y-1)\cdot(x+y) && \text{für } y \geq 2\end{aligned}$$

Geben Sie zur Funktion f den zugehörigen λ-Term $\langle f\rangle$ an.