Aufgabenblatt zur 9. Übung

Zeitraum: 06.06. bis 10.06.2011

Bitte beachten: Ab Mittwoch, dem 08.06.11, wird der Übungsbetrieb **mittwochs in der 1. DS** wegen extrem geringer Teilnehmerzahlen eingestellt.

1. Aufgabe: (Klausuraufgabe 02.2011)

(a) Gegeben sei folgendes Fragment eines C_1-Programms.

```
#include <stdio.h>

int x, y;

void g(...) {
...
}

void f(int a, int* b) {
int c;
if (x < y)
   g(c, &x);
f(c, b);
}

void main() {
...
}
```

Übersetzen Sie die Sequenz der Statements im Rumpf von `f` in entsprechenden AM_1-Code mit baumstrukturierten Adressen (mittels *stseqtrans*). Zwischenschritte brauchen Sie keine anzugeben. Geben Sie zunächst die dazu benötigte Symboltabelle $tab_{f+lDecl}$ an.

$tab_{f+lDecl} =$

AM_1-Code:

(b) Gegeben sei folgender AM_1-Code:

1:	`INIT 1;`	6:	`STORE(lokal,1);`	11:	`RET 2;`	16:	`LOADA(global,1);`
2:	`CALL 12;`	7:	`LOAD(lokal,1);`	12:	`INIT 1;`	17:	`PUSH;`
3:	`JMP 0;`	8:	`LOAD(lokal,-3);`	13:	`READ(lokal,1);`	18:	`CALL 4;`
4:	`INIT 1;`	9:	`ADD;`	14:	`LOAD(lokal,1);`	19:	`WRITE(global,1);`
5:	`LIT 1;`	10:	`STOREI(-2);`	15:	`PUSH;`	20:	`RET 0;`

Führen Sie das folgende Ablaufprotokoll der AM_1 weiter, indem Sie sie schrittweise ablaufen lassen, bis sie stoppt oder die Zeilen der Tabelle erschöpft sind.

BZ	DK	LK	REF	Inp	Out
15	7	$(0) : (3 : 0 : 7)$	3	ε	ε

2. Aufgabe: (AGS 13.9*)

(a) Übersetzen Sie die beiden nachfolgenden C_1-Statements, die sich im Rumpf der Funktion g befinden, in entsprechenden baumstrukturierten AM_1-Code. Nehmen Sie an, `while` sei das dritte Statement in g. Zwischenschritte brauchen Sie keine anzugeben.
Die aktuelle Symboltabelle ist:

$tab_{g+lDecl} = [f/(\text{proc}, 1), g/(\text{proc}, 2), a/(\text{var}, \text{global}, 1), b/(\text{var}, \text{global}, 2), c/(\text{var}, \text{lokal}, 1),$
$y/(\text{var}, \text{lokal}, -3), z/(\text{var-ref}, -2)].$

```
...
while (a >= 0){ b=*z; g(b,z);}
f(&c);
...
```

(b) Gegeben sei folgender AM_1-Code:

1:	`INIT 2;`	6:	`STORE(global,2);`	11:	`READ(lokal,1);`	16:	`PUSH;`
2:	`CALL 10;`	7:	`LIT 7;`	12:	`LOAD(lokal,1);`	17:	`CALL 4;`
3:	`JMP 0;`	8:	`STORE(lokal,1);`	13:	`STORE(global,1);`	18:	`WRITE(lokal,2);`
4:	`INIT 1;`	9:	`RET 1;`	14:	`READ(lokal,2);`	19:	`RET 0;`
5:	`LOADI(-2);`	10:	`INIT 2;`	15:	`LOADA(lokal,2);`		

Betrachten Sie nun die AM_1, die sich bereits im Zustand (Konfiguration)

$$\sigma = (14,\ \varepsilon,\ 7 : 0 : 3 : 0 : 7 : 0,\ 4,\ 10,\ \varepsilon)$$

befindet. Lassen Sie die AM_1, beginnend mit σ, solange ablaufen, bis die Maschine stoppt. Dokumentieren Sie den Zustand der AM_1 nach Ausführung jedes Befehls.

3. Aufgabe: (AGS 14.14*)

Gegeben sei der folgende Anweisungsteil *while* eines C_0–Programms:

```
while (z > 0) {
  z = z - 1;
  y = y + 3;
}
```

Beweisen Sie die Verifikationsformel $\{(x \geq 0) \wedge (z = x) \wedge (y = 0)\}$ *while* $\{(y = 3x)\}$ mit Hilfe des Hoare–Kalküls, indem Sie die Beweisschritte angeben und den Namen der jeweils verwendeten Regel hinzuschreiben.
Ermitteln Sie zunächst eine geeignete Schleifeninvariante.

Zusatzaufgabe: (AGS 13.11*)

(a) Gegeben sei folgendes Fragment eines C_1-Programms.

```
#include <stdio.h>

int a, b;

void g(...)
{...}

void f(int *y)
{ int x;
  while (a == 0) { a=a-x; g(b, &x);}
  f(y);
}

void main()
{...}
```

Übersetzen Sie die Sequenz der Statements im Rumpf von `f` in entsprechenden baumstrukturierten AM_1-Code (mittels *stseqtrans*). Zwischenschritte brauchen Sie keine anzugeben.

Geben Sie zunächst die dazu benötigte Symboltabelle an.

(b) Gegeben sei folgender AM_1-Code:

1:	`INIT 2;`	6:	`STORE(global,2);`	11:	`READ(lokal,2);`	16:	`PUSH;`
2:	`CALL 10;`	7:	`LOAD(lokal,-3);`	12:	`READ(lokal,1);`	17:	`CALL 4;`
3:	`JMP 0;`	8:	`STORE(global,1);`	13:	`LOAD(lokal,2);`	18:	`WRITE(global,2);`
4:	`INIT 0;`	9:	`RET 2;`	14:	`PUSH;`	19:	`RET 0;`
5:	`LOADI(-2);`	10:	`INIT 2;`	15:	`LOADA(lokal,1);`		

Betrachten Sie nun die AM_1, die sich bereits im Zustand (Konfiguration)

$$\sigma = (12,\ \varepsilon,\ 0:0:3:0:0:8,\ 4,\ 6,\ \varepsilon)$$

befindet. Lassen Sie die AM_1, beginnend mit σ, solange ablaufen, bis die Maschine stoppt. Dokumentieren Sie den Zustand der AM_1 nach Ausführung jedes Befehls.