11. Übung (2.–6. Juli 2018)

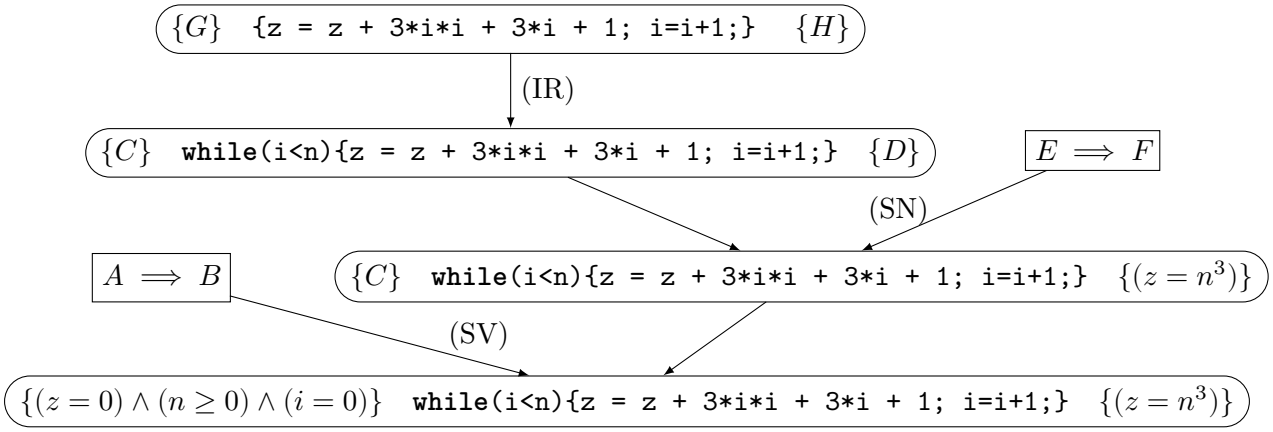
Programmierung

Aufgabe 1 (AGS 16.26)

Die Verifikationsformel

$\{(z = 0) \wedge (n \geq 0) \wedge (i = 0)\}$ `while(i<n){z = z + 3*i*i + 3*i + 1; i=i+1;}` $\{(z = n^3)\}$

soll mit dem Hoare-Kalkül bewiesen werden. Ein Teil eines Beweisbaums wurde unten bereits aufgeschrieben, die Ausdrücke A bis H sind jedoch noch unbekannt.



(a) Geben Sie eine geeignete Schleifeninvariante an.

(b) Geben Sie die Ausdrücke A bis H an. Sie können dabei die Schleifeninvariante mit *SI* abkürzen.

Aufgabe 2 (AGS 17.4)

(a) Gegeben ist folgendes H_0-Programm:

```
module Main where

test :: Int -> Int -> Int -> Int
test x1 x2 x3 = if x1 == 0 then x3
                else test (x1 - 1) x3 ((x2 * x3) + x3)

main = do x1 <- readLn
          x2 <- readLn
          print (test x2 3 x1)
```

Transformieren Sie dieses Programm nach den Transformationsvorschriften, wie sie in der Vorlesung angegeben wurden, in ein AM_0-Programm mit baumstrukturierten Adressen. Sie brauchen dabei keine Zwischenschritte anzugeben.

(b) Eine Folge e_i $(i \geq 1)$ von Zahlen sei wie folgt definiert:

- Die ersten beiden Glieder der Folge seien 1.
- Ab dem dritten Glied der Folge soll gelten: Jedes Folgenglied ist gleich der Summe der Quadrate der beiden Vorgängerglieder.

Geben Sie ein H_0-Programm `P` an, welches das n-te Folgenelement dieser Folge berechnet und ausgibt.

Aufgabe 3 (AGS 17.27)

(a) Sei $f\colon \mathbb{N} \to \mathbb{N}$ eine Funktion mit $f(n) = \sum_{i=1}^{n} \prod_{j=1}^{i} j$. Schreiben Sie ein H_0-Programm, welches eine natürliche Zahl n einliest und $f(n)$ ausgibt.

(b) Transformieren Sie die folgende Funktion `f` eines H_0-Programms in ein AM_0-Programm. Sie können dabei die Adressen frei vergeben, sie müssen nicht baumstrukturiert sein. Berechnen Sie also *funtrans*(`f :: Int -> ...`). Geben Sie dabei keine Zwischenschritte an.

```
f :: Int -> Int
f x1 = if x1 < 42
       then x1
       else if x1 > 42
            then f (x1 `div` 2)
            else 42
```

(c) Das folgende H_0-Programmstück ist durch Nutzung der Transformationsfunktion aus der Vorlesung/Übung aus Statements eines entsprechenden C_0-Programms entstanden. Geben Sie diese Statements des C_0-Programms an.

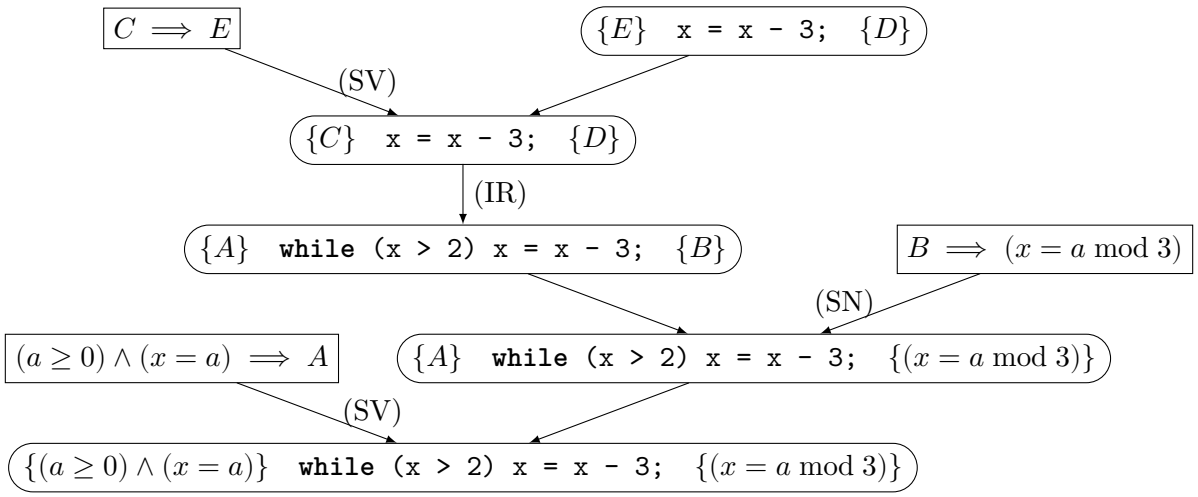
```
f1  x1 = if x1 `mod` 2 == 0 then f11 x1
                            else f12 x1
f11 x1 = f2 (x1 `div` 2)
f12 x1 = f2 (x1 - 1)
f2  x1 = f3 (2 * x1)
```

Zusatzaufgabe 1 (AGS 16.23)

Die Verifikationsformel

$$\{(a \geq 0) \wedge (x = a)\} \quad \texttt{while (x > 2) x = x - 3;} \quad \{(x = a \bmod 3)\}$$

soll mit dem Hoare-Kalkül bewiesen werden, wobei die Operation "'mod"' den Rest bei ganzzahliger Division bildet, z. B. $2 \bmod 3 = 2$ und $5 \bmod 3 = 2$.
Der Beweisbaum wurde unten bereits aufgeschrieben, die Ausdrücke A bis E sind jedoch noch unbekannt.



(a) Geben Sie eine geeignete Schleifeninvariante an.

(b) Geben Sie die Ausdrücke A, B, C, D, und E an. Sie können dabei die Schleifeninvariante mit SI abkürzen.

Zusatzaufgabe 2 (AGS 15.11)

(a) Folgendes Fragment eines C_1-Programms sei bekannt:

```
1  #include <stdio.h>
2
3  int x;
4
5  void h(...) {...}
6  void g(...) {...}
7
8  void f(int a, int *b) {
9    int c;
10   if (x>1) g(b); else h(a,&x);
11   c=*b + 1;
12 }
13
14 void main(){...}
```

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

Geben Sie zunächst die dazu benötigte Symboltabelle an (in den Übungen wurde diese mit $tab_{f+lDecl}$ bezeichnet).

(b) Lassen Sie die AM_1, beginnend mit $(12,\ \varepsilon,\ 0 : 3 : 0 : 7,\ 3,\ 5,\ \varepsilon)$, auf dem unten gegebenen AM_1-Code solange ablaufen, bis die Maschine terminiert.

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