Programmierung

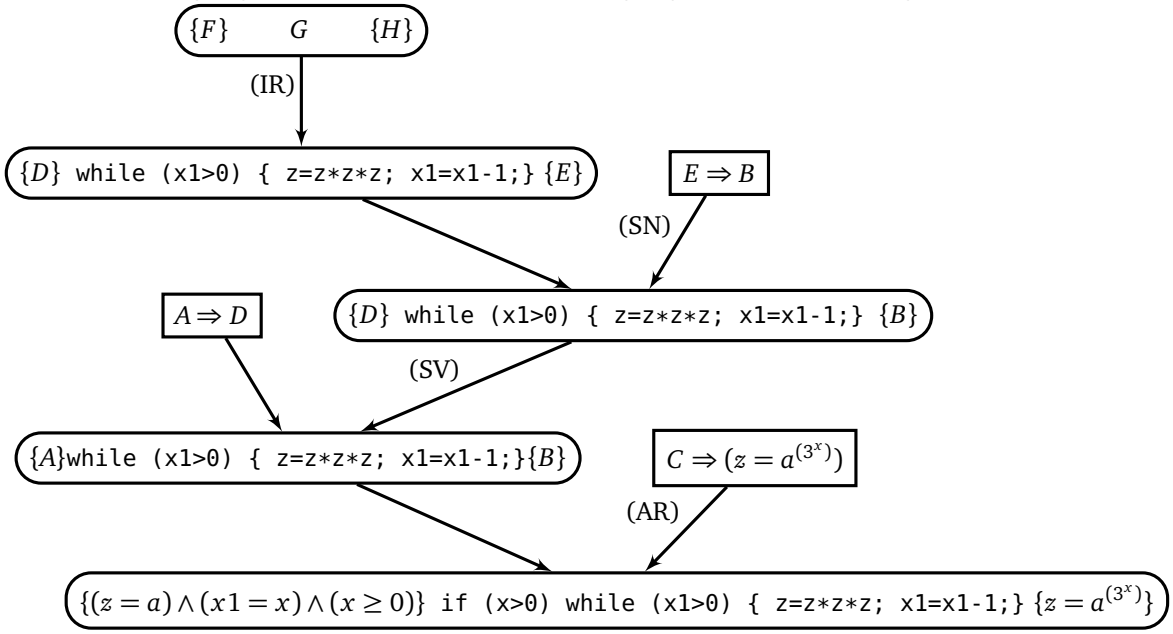
11. Übungsblatt

Zeitraum: 27. Juni – 01. Juli 2016

Übung 1 (AGS 15.9)
Für die Verifikationsformel

$$\{(z = a) \wedge (x1 = x) \wedge (x \geq 0)\}\ \texttt{if (x>0) while (x1>0) \{ z=z*z*z; x1=x1-1;\}}\{z = a^{(3^x)}\}$$

wurden die ersten vier (korrekten) Regelanwendungen des Beweisbaums aufgeschrieben (siehe unten). Dabei sind die Ausdrücke A bis H noch unbekannt. Es gelten: AR = Alternativregel, SV = stärkere Vorbedingung, SN = schwächere Nachbedingung, IR = Iterationsregel.



(a) Geben Sie eine geeignete Schleifeninvariante an.
(b) Geben Sie die Ausdrücke A, B, C, D, E, F, G und H an. Kürzen Sie gegebenenfalls die Schleifeninvariante mit SI ab.

Übung 2 (AGS 16.4)
(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.

Übung 3 (AGS 16.6)
Wandeln Sie das folgende H_0-Programm in ein AM_0-Programm um. Sie müssen keine Zwischenschritte angeben.

```
module Main where

fac :: Int -> Int -> Int
fac x1 x2 = if x1 > 0 then fac (x1 - 1) (x1 * x2)
                      else x2

main = do x1 <- readLn
          print (fac x1 1)
```

Zusatzaufgabe 1 (AGS 14.15)

(a) Gegeben sei die Symboltabelle

$$tab_{g+lDecl} = [\mathrm{g}/(\mathrm{proc}, 1), \mathrm{x}/(\mathrm{var}, \mathrm{global}, 1), \mathrm{p}/(\text{var-ref}, -2), \mathrm{i}/(\mathrm{var}, \mathrm{lokal}, 1)],$$

sowie folgendes Fragment aus der Funktion `g` des zugehörigen C_1-Programms:

```
if (x > 0) {
  g(&x);
  i = x / 2;
} else {
  x = *p;
}
printf("%d", x);
```

Übersetzen Sie dieses Fragment anhand $tab_{g+lDecl}$ in entsprechenden AM_1-Code mit baumstrukturierten Adressen (mittels *stseqtrans*). Nehmen Sie an, `if` sei das dritte Statement in `g`. Sie brauchen keine Zwischenschritte anzugeben.

(b) Gegeben sei folgender AM_1-Code:

```
 1: INIT 2;            7: EQ;          13: RET 2;           19: PUSH;
 2: CALL 14;           8: JMC 13;      14: INIT 0;          20: CALL 4;
 3: JMP 0;             9: LOADI(-2);   15: READ(global,1);  21: WRITE(global,2);
 4: INIT 1;           10: LIT 23;      16: LOAD(global,1);  22: RET 0;
 5: LOAD(lokal, -3);  11: ADD;         17: PUSH;
 6: LIT 1;            12: STOREI(-2);  18: LOADA(global,2);
```

Protokollieren Sie 16 Schritte des schrittweisen Ablaufes dieses Programms auf der AM_1, ausgehend vom Startzustand $\sigma = (15, \varepsilon, 0:0:3:0:0, 4, 1, \varepsilon)$.