Programmierung

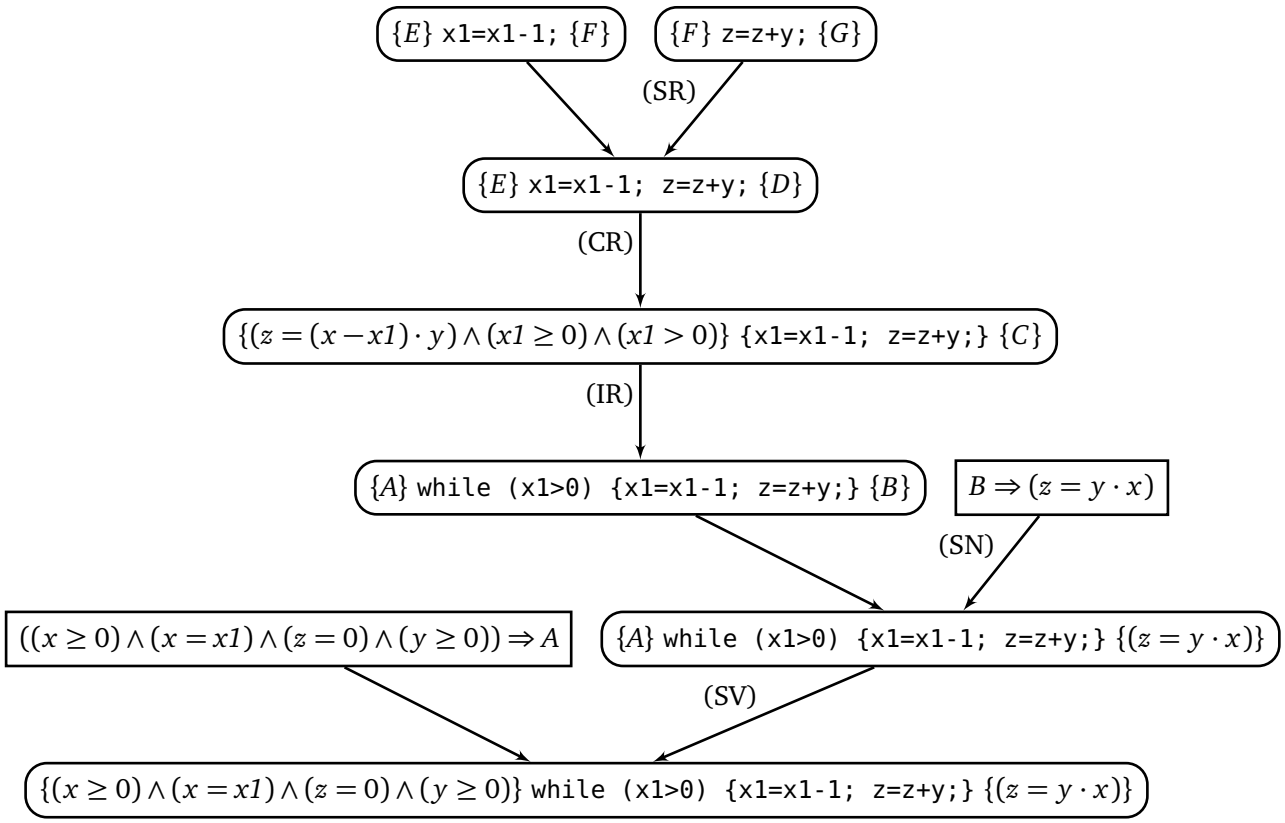
09. Übungsblatt

Zeitraum: 13. – 17. Juni 2016

Übung 1 (AGS 15.2)
Mit Hilfe des Hoare-Kalküls wurde für die Verifikationsformel

$$\{(x \geq 0) \wedge (x = x1) \wedge (z = 0) \wedge (y \geq 0)\}\ \texttt{while (x1>0) \{x1=x1-1; z=z+y;\}}\ \{(z = y \cdot x)\}$$

der folgende korrekte Beweisbaum aufgestellt. Hierbei wurden jedoch nur die Ergebnisse der jeweils angewandten Regeln aufgeschrieben. Es gelten: SV = stärkere Vorbedingung, SN = schwächere Nachbedingung, IR = Iterationsregel, CR = Compoundregel, SR = Sequenzregel.



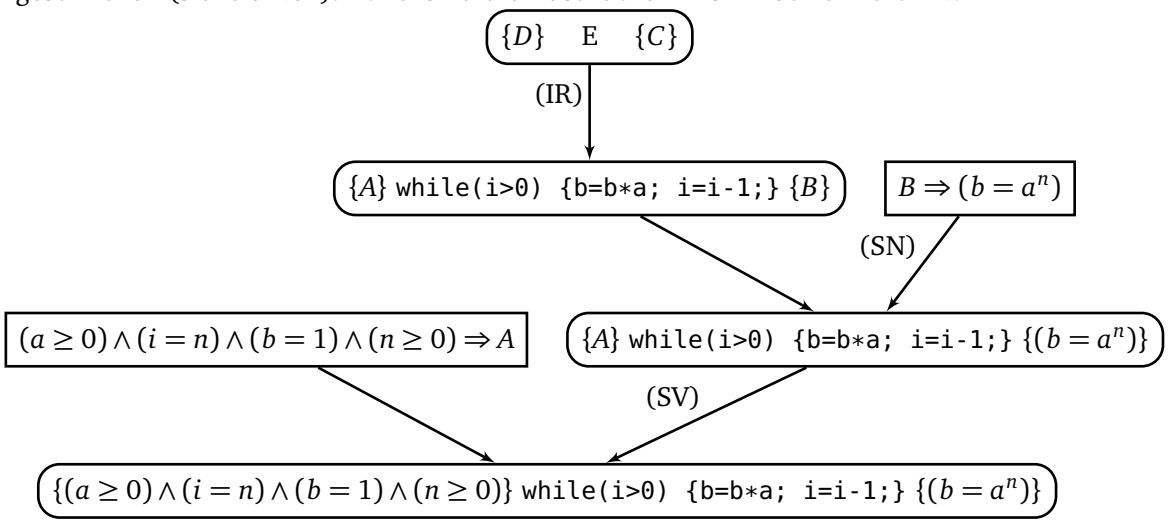
mit $F = (z + y = (x - x1) \cdot y) \wedge (x1 \geq 0)$

(a) Geben Sie die Schleifeninvariante an.
(b) Geben Sie die Ausdrücke für A, B, C, D, E und G an.
(c) Zeigen Sie die Gültigkeit der Verifikationsformel $\{E\}$ `x1=x1-1;` $\{F\}$.

Übung 2 (AGS 15.22)
Für die Verifikationsformel

$$\{(a \geq 0) \wedge (i = n) \wedge (b = 1) \wedge (n \geq 0)\}\ \texttt{while (i>0) \{ b=b*a; i=i-1;\}}\ \{(b = a^n)\}$$

wurden mit dem Hoare-Kalkül die ersten drei (korrekten) Regelanwendungen des Beweisbaums aufgeschrieben (siehe unten). Dabei sind die Ausdrücke A bis E 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.

Übung 3 (AGS 14.17)
(a) Gegeben sei folgendes Fragment eines C_1-Programms mit den Funktionen `f` und `g`:

```
while(*p > i) {
  f(p);
  i = i + 1;
}
p = &i;
```

Übersetzen Sie die Sequenz dieser Statements in entsprechenden AM_1-Code mit baumstrukturierten Adressen (mittels *stseqtrans*). Sie müssen keine Zwischenschritte angeben. Nehmen Sie an, die `while`-Anweisung sei das zweite Statement in `g`, und es sei

$$tab_{g+lDecl} = \left\{\texttt{f}/(\text{proc}, 1), \texttt{g}/(\text{proc}, 2), \texttt{i}/(\text{var}, \text{lokal}, 1), \texttt{p}/(\text{var-ref}, -2)\right\}.$$

(b) Gegeben sei folgender AM_1-Code:

```
1: INIT 1;           8: LIT 2;           15: STORE(lokal,-2); 22: LOAD(global,1);
2: CALL 18;          9: LOADI(-3);       16: JMP 4;           23: PUSH;
3: INIT 0;          10: MUL;             17: RET 2;           24: CALL 3;
4: LOAD(lokal,-2); 11: STOREI(-3);       18: INIT 0;          25: WRITE(global,1);
5: LIT 0;          12: LOAD(lokal,-2);  19: READ(global,1);  26: JMP 0;
6: GT;             13: LIT 1;           20: LOADA(global,1);
7: JMC 17;         14: SUB;             21: PUSH;
```

Führen Sie die AM_1 auf der Konfiguration $\sigma = (22, \varepsilon, 1:3:0:1, 3, \varepsilon, \varepsilon)$ weiter. Sie müssen nur 13 Schritte ausführen!

Zusatzaufgabe 1 (AGS 14.16)

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

```
#include <stdio.h>

int x, y;

void f(...) {...}

void g(int a, int *b) {
  int c;
  c = 3;
  if (c == *b)
    while (a > 0)
      f(&a, b);
}

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

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

(b) Gegeben sei folgender AM_1-Code:

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

Führen Sie das folgende Ablaufprotokoll der AM_1 weiter, indem Sie sie schrittweise ablaufen lassen. Sie müssen nur die ersten 16 Zeilen ausfüllen. Zusätzlich müssen Sie nur Zellen ausfüllen, deren Wert sich im Vergleich zur letzten Zeile geändert hat. Die Startkonfiguration lautet

$$(7,\ 1:1,\ 1:3:0:1:1:21:3:0,\ 7,\ \varepsilon,\ \varepsilon).$$

Zusatzaufgabe 2 (AGS 12.4.26)

(a) Berechnen Sie die Normalform des untenstehenden λ-Terms, indem Sie ihn *schrittweise* reduzieren. Geben Sie dabei vor jedem Schritt für die relevanten Teilausdrücke die Mengen der gebunden bzw. frei vorkommenden Variablen an.

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

(b) Gegeben seien der λ-Term

$$\begin{aligned}\langle G\rangle = (\lambda gxy.&\langle ite\rangle(\langle iszero\rangle x\ )\ y\\ &(\langle ite\rangle(\langle iszero\rangle(\langle pred\rangle x))\\ &\quad(\langle mult\rangle\langle 2\rangle y)\\ &\quad(\langle mult\rangle(g\ (\langle pred\rangle x\ )\ y)(g\ (\langle pred\rangle(\langle pred\rangle x))\ (\langle succ\rangle y)))))\end{aligned}$$

und der Fixpunktkombinator $\langle Y\rangle = \big(\lambda z.\big((\lambda u.z(uu))(\lambda u.z(uu))\big)\big)$. Geben Sie die durch $\langle Y\rangle\langle G\rangle$ beschriebene rekursive Funktion als Haskell-Funktion `g` an.

(c) Gegeben sei der λ-Term

$$\langle F\rangle = (\lambda fxyz.\langle ite\rangle(\langle iszero\rangle\ y)\ (\langle add\rangle\ x\ x)\ (\langle mult\rangle\ z\ (\ f\ (\langle succ\rangle x)\ (\langle pred\rangle y\ )\ z)))$$

Berechnen Sie die Normalform des Terms $\langle Y\rangle\langle F\rangle\langle 4\rangle\langle 1\rangle\langle 2\rangle$ unter Angabe geeigneter Zwischenschritte. Führen Sie im Rechenprozess zweckmäßige Abkürzungen der λ-Terme ein.