

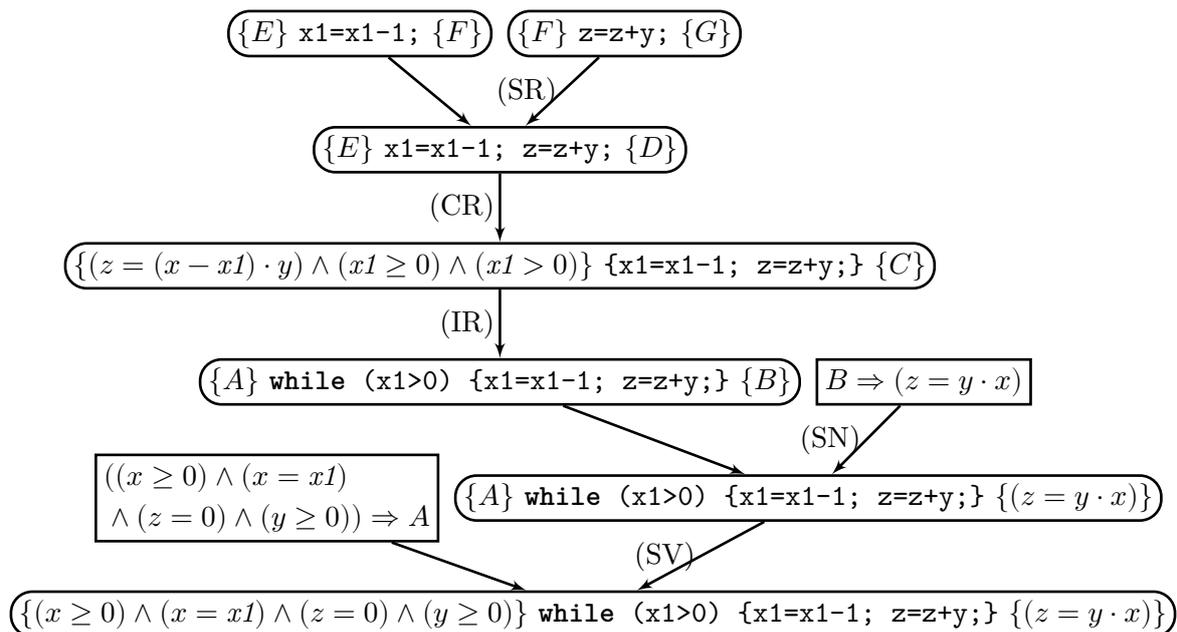
Programmierung

Aufgabe 1 (AGS 16.2)

Mit Hilfe des Hoare-Kalküls wurde für die Verifikationsformel

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

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.

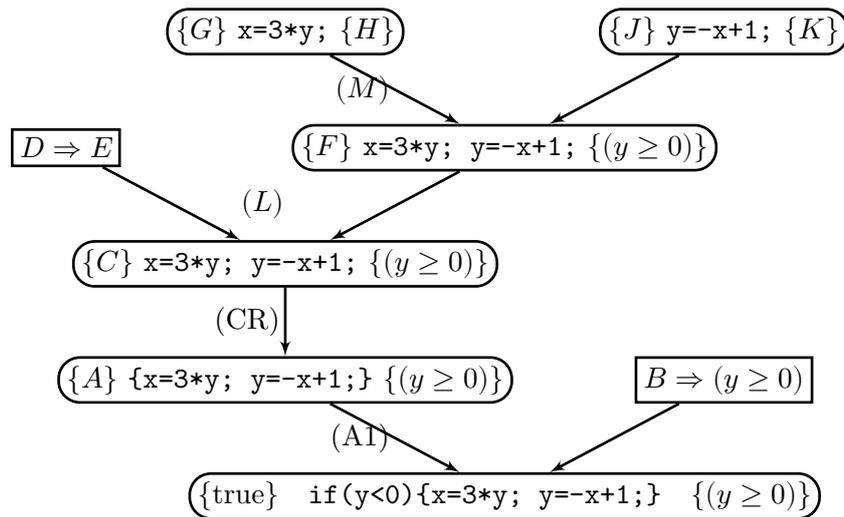


mit $F = (z + y = (x - x1) \cdot y) \wedge (x1 \geq 0)$

- (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 $\{E\} x1=x1-1; \{F\}$.

Aufgabe 2

Die Verifikationsformel $\{\text{true}\} \text{ if } (y < 0) \{x = 3 * y; y = -x + 1;\} \{(y \geq 0)\}$ soll mit dem Hoare-Kalkül bewiesen werden. Ein Teil eines Beweisbaums wurde unten bereits aufgeschrieben, die Ausdrücke A bis M sind jedoch noch unbekannt. Der Ausdruck true bezeichnet eine beliebige tautologische Formel, wie z. B. $(1 = 1)$. Es gelten die Abkürzungen: A1 = erste Alternativregel, CR = Compregel.



(a) Geben Sie die Ausdrücke A bis M an.

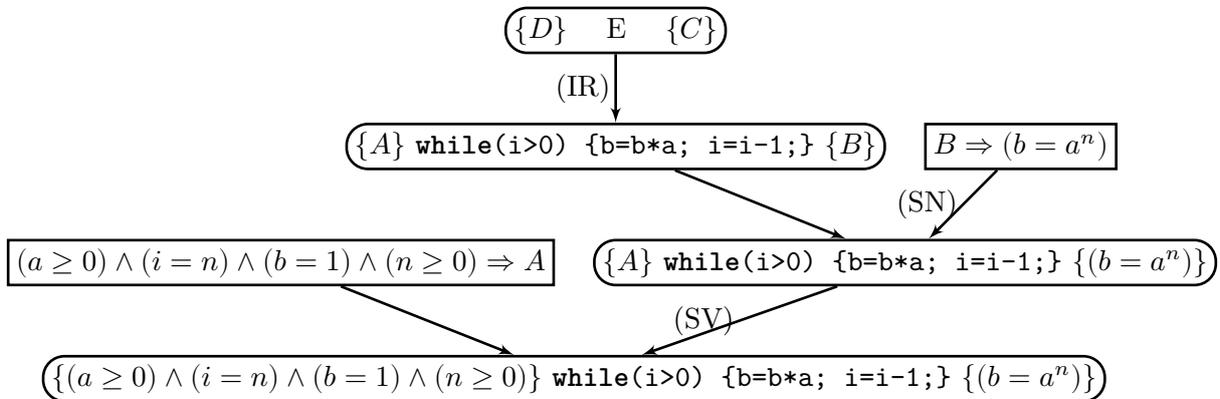
(b) Zeigen Sie schrittweise, dass $\text{true} \wedge (y < 0) \implies (-3 \cdot y + 1 \geq 0)$ gilt.

Aufgabe 3 (AGS 16.22)

Für die Verifikationsformel

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

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.

Zusatzaufgabe 1 (AGS 15.11)

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

```
#include <stdio.h>
int x;
```

```

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

void f(int a, int *b) {
    int c;
    if (x > 1) g(b); else h(a, &x);
    c = *b + 1;
}

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 $tab_{f+IDecl}$ an.

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

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