Logik, Semantik und Verifikation SS 2002:
Musterlösung zum 10. Übungsblatt

Prof. Dr. Gert Smolka, Dipl.-Inform. Tim Priesnitz

Aufgabe 10.1: Bindungsstruktur und Umbenennung (12) Wir lösen die Teilaufgaben nur für A_1, denn A_2 und A_3 sind analog.

(a) $A_1 = \exists\overline{X}_1\ (X_1 \leq Y \wedge \exists\overline{Y}_1\ (Z \leq Y_1) \wedge Y \leq X_1)$

(b) $FV(A_1) = \{Y, Z\}$.

(c) $A_1' = \exists X\ (X \leq Y \wedge \exists U\ (Z \leq U) \wedge Y \leq X)$

(d)

$$\begin{aligned} A_1[X/Y][Z/X] &= (\exists X\ (X \leq Y \wedge \exists Y\ (Z \leq Y) \wedge Y \leq X))[X/Y][Z/X] \\ &\sim (\exists U\ (U \leq Y \wedge \exists Y\ (Z \leq Y) \wedge Y \leq U))[X/Y][Z/X] \\ &= (\exists U\ (U \leq X \wedge \exists Y\ (Z \leq Y) \wedge X \leq U))[Z/X] \\ &= (\exists U\ (U \leq Z \wedge \exists Y\ (Z \leq Y) \wedge Z \leq U)) \end{aligned}$$

Aufgabe 10.2: Hoare-Regeln (12)

(a)

$$\frac{\{A\}\ c\ \{B\} \qquad \models B \wedge \neg b \Rightarrow A}{\{A\}\ \texttt{do}\ c\ \texttt{until}\ b\ \{B \wedge b\}}$$

(b)

$$\frac{\{I \wedge X \leq a\}\ c\ \{I[X+1/X]\}}{\{I\}\ \texttt{for}\ X\ \texttt{to}\ a\ \texttt{do}\ c\ \{I \wedge X > a\}}$$

(c) Ein Gegenbeispiel ist:

$$\{X = 1\}\ X := X + 1\ \{X = 1[X+1/X]\}$$

Denn sei $\sigma \in \Sigma$ beliebig mit $\sigma(X) = 1$. Dann gilt $\sigma \models X = 1$. Einerseits ist

$$\begin{aligned} (\mathcal{C}(X := X + 1)\sigma)(X) &= (\sigma[\mathcal{A}(X+1)\sigma/X])(X) \\ &= \mathcal{A}(X)\sigma + 1 \\ &= \sigma(X) + 1 = 2 \end{aligned}$$

aber andererseits

$$X = 1[X+1/X] \iff X + 1 = 1 \iff X = 0$$

Also haben wir:

$$\not\models \{X = 1\}\ X := X + 1\ \{X = 1[X+1/X]\}$$

Aufgabe 10.3: Verifikation (10) Sei $A = (N \geq 1)$ und $B = (P = M * N)$. Als Invariante wählen wir:

$$I = (P = M * (C - 1) \wedge C \leq N + 1)$$

Verifikationsbedingungen:

- $I \wedge \neg(C \leq N) \models B$
- $I \wedge C \leq N \models I[C + 1/C][P + M/P]$
- $A \models I[1/C][0/P]$

Aufgabe 10.4: Verifikation (10) Wir beschränken uns auf die Angabe der Invariante I: $Y = Z * Z \wedge Z \leq X$.

Aufgabe 10.5: Berechnen von Verifikationsbedingungen (6)

(a)
```
fun dis b1 b2     = Not(And(Not b1, Not b2))
fun imp b1 b2     = dis (Not b1) b2
fun cond b1 b2 b3 = dis (And(b1,b2)) (And(Not b1,b3))
```

(b)
```
fun substa (Con i)      _  _ = (Con i)
  | substa (Loc l)      a  l'= if l=l' then a else (Loc l)
  | substa (Add(a1,a2)) a  l = Add(substa a1 a l,substa a2 a l)
  | substa (Mul(a1,a2)) a  l = Mul(substa a1 a l,substa a2 a l)

fun substb (LE(a1,a2))  a l = LE(substa a1 a l,substa a2 a l)
  | substb (Not b)      a l = Not(substb b a l)
  | substb (And(b1,b2)) a l = And(substb b1 a l,substb b2 a l)
  | substb b            _ _ = b
```

(c)
```
fun vcs Skip           a = (a,nil)
  | vcs (Assign(l,ae)) a = (substb a ae l,nil)
  | vcs (Seq(c1,c2))   a = let val (a2,vs2) = vcs c2 a
                               val (a1,vs1) = vcs c1 a2
                           in  (a1,vs1@vs2)
                           end
  | vcs (If(b,c1,c2))  a = let val (a1,vs1) = vcs c1 a
                               val (a2,vs2) = vcs c2 a
                           in  (cond b a1 a2,vs1@vs2)
                           end
  | vcs (While(i,b,c)) a =
        let val (a',vs) = vcs c i
            val v1      = imp (And(i,b)) a'
            val v2      = imp (And(i,Not b)) a
        in  (i,v1::v2::vs)
        end

fun vc a c b = let val (a',vs) = vcs c b
               in  imp a a'::vs
               end
```