Fragen, Antworten und Kommentare zur aktuellen Vorlesung

Das Video zur Lösung der Aufgabe 31 finden Sie unter: https://youtu.be/riCU3u6HtvE . Bei genauerer Betrachtung steht in der ersten langen Zeile einmal nur " x >" statt "x > 0".

Das Video zur Lösung der Aufgabe 32 finden Sie unter: https://youtu.be/hcQ-hWSgT6M . Im letzten Satz wird wieder von einer Abschwächung gesprochen. Das ist falsch, da ja z=x gegeben ist und dann eine Verstärkung zur ursprünglichen Bedingung stattfindet. Die Logik ist damit korrekt, die Argumentation aber unsauber.

Das Thema „Korrektheitsbeweis“ wird auch in der Übung in der nächsten Woche behandelt.

Korrektur: Im letzten Beispiel im Vorlesungsvideo zu b) ca. ab 83:10 ist die Formulierung bzw. Beweisführung schlicht falsch, ist aber zum Glück einfach zu retten. Damit der geschriebene Text stimmt, muss im Programm x<10 durch x<=10 ersetzt werden. Alternativ, wenn Sie das Programm so lassen, landen Sie im else-Zweig und erhalten x=20, was ebenfalls nicht in der als Nachbedingung angegebenen Zustandsmenge liegt.

Frage: Was ist eigentlich genau die Invariante, die Voraussetzung zur Nutzung der Regel ist ja genauer {p ∧ B} P {p}?

Antwort: Gemeint ist damit immer nur p, d. h. eine Eigenschaft, die vor und nach der Programmausführung von P gilt. Sie haben aber richtig beobachtet, dass in den bisherigen (und kommenden) Aufgaben {p} P {p} bewiesen wird. Das ist aber ok, da wir Vorbedingungen verstärken dürfen, da p ∧ B -> p gilt. Im Beweis müsste genauer nur p ∧ B gezeigt werden. Das spielt bei unseren Beispielen keine Rolle, da B nicht den Schleifenrümpfen vorkommt. Die Notwendigkeit der Voraussetzung zeiget folgendes Beispiel:.

{z=x ∧ x>0}

```
y := 0;
while (y < x) {
  y := y + 1;
}
```

{z=y ∧ z=x}

Da weder x und z im Programm verändert werden, bleibt z=x erhalten und es wird sich mit z nur der Ausgangswert von x gemerkt. In die Invariante muss die Negation der Booleschen Bedingung, also genauer y >= x so einfließen, dass am z=y gilt. Dies ist durch folgende Invariante p möglich:
p ≡ (z >= y ∧ z=x)
Zunächst der Versuch des Invarianten-Nachweises:
p [y := y + 1] ≡ (z >= y+1 ∧ z=x)
Es gilt aber definitiv nicht: (z >= y ∧ z=x) (z >= y+1 ∧ z=x) (z. B. mit z=4 und y=4)
Da wir aber nicht {p} P {p} sondern nur {p ∧ B} P {p} zeigen müssen, wird dies geprüft:
„p ∧ B“ ≡ (z >= y ∧ z = x ∧ y < x) dies impliziert (z >= y ∧ z = x ∧ y < z). Da y und z int-Werte sind, gilt y < z ≡ y + 1 <= z. Damit kann von (z >= y ∧ z = x ∧ y < z) auf (z >= y+1 ∧ z=x) geschlossen werden. Somit ist die Vorbedingung der Schleifenregel gezeigt.

Nach der Schleife gilt: (z >= y ∧ z=x ∧ y>=x), da aus (z >= y >= x) mit z=x auch z=y folgt, passt die Nachbedingung.
Vor der Schleife gilt: (z >= y ∧ z=x) [y :=0] ≡ (z >= 0 ∧ z=x) ≡ (x >= 0 ∧ z=x) was zu war.

Frage: Ich habe da noch eine weitere Frage für verschachtelte If anweisungen wie bspw. im Aufgabenblatt 8

```
{true}
if (x>3) {
  y := 1;
} else {
    if(x<5) {
      y := 0;
    } else {
      y := 42;
    }
 }
}
{y=0 ∨ y=1}
```

Kann ich das auch beweisen in dem ich ein Programm ablauf simuliere, also bspw. Zustand z1: z1(x) = 4, dann würde da z1´(y) = 1 rauskommen und mit der nachbedinung {y=0 ∨ 1 =1 } erfolgen und man würde doch dann hier die Partielle Korrektheit für diesen fall beweisen können ? oder wie ist der gedanken weg den ich hier machen soll. Da es hier ja kaum ein "kochrezept" gibt.

Antwort: Nö, der Ansatz ist nicht zielführend. Mit Beispielen kann man nur zeigen, dass ein Hoare-Tripel nicht gilt. Ein Positiv-Beispiel zeigt nur wie ein Test, dass ein konkreter Fall funktioniert. Simulation oder Tests sind nur in extremen Spezialfällen (z. B. wenn es nur endlich viele Zustände geben kann) für eine Verifikation nutzbar.

Es gibt einen Ansatz der symbolischen Ausführung, der etwas mit Ihrer Idee zu tun hat. In Richtung Klausur geht der Positiv-Beweis nur mit dem Beweissystem.

Frage: Wie finde ich systematisch heraus, ob ein Hoare-Tripel mit einem if korrekt sein kann.

Antwort: Ähnlich wie beim Testen prüfen Sie die Extremwerte, hier des Programms und der Vorbedingung. Bei

{x>0 ∧ x <16}

```
if (x>8) {
  x := x - 8;
} else {
  x := x + 3;
}
```

{x>0 ∧ x <12}

sind dies x=1 und x=15 aus der Vorbedingung und x=8 und x=9 aus dem if. Selten kann ein Blick auf die Extremwerte der Nachbedingung (1, 11) rückwärts auch helfen.

Frage: Das Thema Schleifeninvarianten scheint sehr komplex zu sein.

Antwort: Das stimmt, ist wahrscheinlich der schwerste Teil der Veranstaltung. Das nachfolgende und die Klausuraufgaben abschließende Thema „Endliche Automaten“ ist dafür deutlich einfacher.

Frage: Wie finde ich Schleifeninvarianten?

Antwort: Dafür gibt es kein allgemeines systematisches Verfahren, kann es nach Halteproblem-Problematik auch nicht geben. Es gibt aber in bestimmten Situationen ähnliche Ansätze. Typische Fälle sind Schleifen, die einfach hoch- oder runterzählen.

Beispiel 1:

{y=x ∧ x>0}
```
z := 0;
while (not(x == 0)) {
  z := z + 1;
  x := x – 1;
}
```
{y=z}
Die Schleifeninvariante nutzt die Nachbedingung und baut den Schleifenzähler in diese ein. Weiterhin muss die Abbruchbedingung der Schleifenbedingung eingehen, so dass, wenn diese Bedingung gilt, möglichst genau die gewünschte Nachbedingung als Ergebnis herauskommt. Im Beispiel bleibt nach der Ausführung des Schleifenrumpfes die Summe aus z+x konstant und sollte in die Schleifeninvariante einfließen. Im konkreten Fall ist dies bereits der zentrale Teil der Invariante.

(y=z+x) [x:=x-1] ≡ (y=z+x-1)

(y=z+x-1) [z:=z+1] ≡ (y=z+1+x-1) ≡ (y=z+x)

Bewiesen ist damit die Nachbedingung y=z+x ∧ x=0. Die gewünschte Nachbedingung folgt mit der Abschwächung (y=z+x ∧ x=0) -> y=z.

Es bleibt aber noch zu zeigen {y=x ∧ x>0} z:=0; {y=z+x ∧ x≠0}:

(y=z+x ∧ x≠0)[z:=0] ≡ (y=0+x ∧ x≠0) ≡ (y=x ∧ x≠0),

da aber die Verstärkung gilt (y=x ∧ x>0) -> (y=x ∧ x≠0), gilt das Hoare-Tripel.

Beispiel 2:

{y=x ∧ x>0}
```
z := 0;
while (not(x == 0)) {
  z := z + 3;
  x := x – 1;
}
```
{3y=z}

Dies ist nur eine kleine Variation von Beispiel 1 und zeigt, wie ein Faktor einfließen kann. Es bleibt nach der Ausführung des Schleifenrumpfes die Summe aus z+3x konstant und sollte in die Schleifeninvariante einfließen.

(3y=z+3x) [x:=x-1] ≡ (3y=z+3(x-1))

(3y=z+3(x-1)) [z:=z+3] ≡ (3y=z+3+3(x-1)) ≡ (3y=z+3x)

Bewiesen ist damit die Nachbedingung 3y=z+3x ∧ x=0. Die gewünschte Nachbedingung folgt mit der Abschwächung (3y=z+3x ∧ x=0) -> 3y=z.

Es bleibt aber noch zu zeigen {y=x ∧ x>0} z:=0; {3y=z+3x ∧ x≠0}:

(3y=z+3x ∧ x≠0)[z:=0] ≡ (3y=0+3x ∧ x≠0) ≡ (3y=3x ∧ x≠0),

da aber die Verstärkung gilt (y=x ∧ x>0) -> (3y=3x ∧ x≠0), gilt das Hoare-Tripel.

Beispiel 3:

{y=x ∧ x>0}
```
z := 0;
while (not(x == z)) {
  z := z + 1;
}
```
{y=z}

Der identische Ansatz von oben passt nicht, man kann allerdings überlegen, dass am Ende x=z gelten wird und daraus die Invariante y=x herleiten. Alternativ ist auch die Überlegung zielführend, dass sich x in der Schleife nicht verändert und x das gewünschte Ergebnis ist.

(y=x) [z:=z+1] ≡ (y=x)

Bewiesen ist damit die Nachbedingung y=x ∧ x=z. Die gewünschte Nachbedingung folgt mit der Abschwächung (y=x ∧ x=z) -> y=z.

Es bleibt aber noch zu zeigen {y=x ∧ x>0} z:=0; {y=x ∧ x≠z}:

(y=x ∧ x≠z)[z:=0] ≡ (y=x ∧ x≠0),

da aber die Verstärkung gilt (y=x ∧ x>0) -> (y=x ∧ x≠0), gilt das Hoare-Tripel.

Beispiel 4:

{y=x ∧ x>0}
```
z := 0;
v := 0;
while (not(x == z)) {
  z := z + 1;
  v := v + 3;
}
```
{3y=v}

Formal ist dies nur eine kleine Änderung von Beispiel 3, bei der Invariante muss jetzt aber v berücksichtigt werden und dass am Ende x=z gilt. In vorherigen Schritten fehlt x-z mal die 3 um zur Nachbedingung zu kommen. Daraus folgt die Invariante 3y = v + 3(x-z)

$(3y=v+3(x-z))[v:=v+3] \equiv (3y=v+3+3(x-z)) \equiv (3y=v+3(1+x-z))$

$(3y=v+3(1+x-z))\ [z:=z+1] \ \equiv \ (3y=v+3(1+x-(z+1))) \equiv (3y=v+3(x-z))$

Bewiesen ist damit die Nachbedingung $3y=v+3(x-z) \wedge x=z$. Die gewünschte Nachbedingung folgt mit der Abschwächung $(3y=v+3(x-z) \wedge x=z) \rightarrow 3y=v$.

Es bleibt aber noch zu zeigen $\{y=x \wedge x>0\}\ z:=0;\ v:=0;\ \{3y=v+3(x-z) \wedge x \neq z\}$:

$(3y=v+3(x-z) \wedge x \neq z)\ [v:=0] \ \equiv (3y=3(x-z) \wedge x \neq z)$

$(3y=3(x-z) \wedge x \neq z)\ [z:=0] \equiv (3y=3x \wedge x \neq 0)$

da aber die Verstärkung gilt $(y=x \wedge x>0) \rightarrow (3y=3x \wedge x \neq 0)$ gilt das Hoare-Tripel.