Fragen, Antworten, Kommentare und Hinweise

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/SnpY8fJpL5s

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

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

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

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

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

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

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