1.3 Verifikation

- Problem:
  Gegeben Spezifikation (P,Q) und Implementierung S
  Gesucht formaler (automatisierbarer ?) Beweis, dass S (P,Q) erfüllt.

- Bisher nicht möglich $P \Rightarrow Q$ zu beweisen, "wenn S ausgeführt wird".
- Notwendige Voraussetzung: Präzise Semantik von Anweisungen

- Zuweisungsaxiom

Zuweisungsaxiom: Wenn vor der Zuweisung der Zustand P(e) für einen Ausdruck e gilt, dann nach der Zuweisung P(x), wobei e durch x ersetzt wurde.

{P} x=e {P[e/x]}

P[a/b] ist der Ausdruck, der sich durch Ersetzen aller freien (nicht quantifizierten) Vorkommen von a durch b ergibt

Beispiele:

{x+1==11} x = x + 1 {x == 11}

Erhöht man x um 1, dann ist x gleich 11, wenn x+1 vorher den Wert 11 hatte (d.h x den Wert 10)

{x*x == 100} x = x**x {x == 100}

Umgekehrt: Gilt P(x) nach Ausführung der Anweisung x=e, dann ist P[x/e] **schwächste Voraussetzung** der Zuweisung

Zuweisung

Warum nicht {P} **x=e** {P[x/e]} ??

Beispiel:

{ prim(7) } x=7 {prim(x)} // prim(x) : " x ist Primzahl "

lies: wenn 7 eine Primzahl ist und 7 der Variablen x zugewiesen wird, dann kann man schließen: x ist eine Primzahl

aber nicht :

{ prim(x) } x=25 {prim(25)} // falsch.

Aus prim(x) folgt nicht nach Zuweisung x = 25, dass prim[x/25] , also 25 eine Primzahl ist

{P} **x=e** {P[e/x] } bedeutet:

Alles, was für e gilt, gilt nach Ausführung der Zuweisung für x, alles andere ist unverändert.

Zuweisung

Folgerungsbeziehung

" {P} x=e {Q} ist wahr" oder "x = e erfüllt die Spezifikation (P,Q) " stellt eine logische Folgerungsbeziehung zwischen P und Q her:

P => Q [e/x] (e ersetzt durch x)

Andere Schreibweise: $P \overset{x=e}{\Rightarrow} Q$

Zuweisung: Beispiel

- Behauptung: $\{x \geq 0\}\ x = x+1\ \{x > 0\}$

Sieht man zwar auf den ersten Blick, wenn man weiß, was eine Zuweisung ist, aber...

Formale Beweisführung:

$\{x \geq 0\}$ => $\{x+1 \geq 1\}$ //Arithmetik
=> $\{x \geq 1\}$ // Zuweisungsaxiom: [x+1 | x]
=> $\{x > 0\}$ //Arithmetik

Rückschreitendes Beweisen

Rückschreitende Beweisführung oft einfacher

"Wenn P (x) nach der Zuweisung x = e gilt, gilt P(e) vor Zuweisung "

$\{P[x/e]\}$ **x=e** $\{P\}$

Beispiel

Zeige: x=x+2 erfüllt Spezifikation
($\{x \geq -1\}$, $\{x > 0\}$)

$\{x > 0\}$ gelte nach Zuweisung gemäß Spezifikation
=> x+2 > 0 //vor Zuweisung (Axiom)
=> x > - 2 // Arithmetik
=> x ≥-1 // Arithmetik

Sequenz

Schlussregel für Sequenz:

$$\frac{\{P\}\ S1\ \{Q\} \wedge \{Q'\}\ S2\ \{R\} \wedge Q \Rightarrow Q'}{\{P\}\ S1;S2\ \{R\}}$$

Ableitungsregel :
Prämisse
Konsequenz

Wie beweist man {P} S1; S2; {R} ?

1. Finde Q so dass {P} S1 {Q} gilt.
2. Schwäche Q ggf. zu Q' ab (d.h. Q => Q')
3. Beweise auf dieselbe Art {Q'} S2 { R}

Sequenz (2)

Wie beweist man {P} S1; S2; {R} ? (Rückschreitend)

1. Finde schwächste Voraussetzung Q' für R unter S2
2. Verstärke Q' ggf. zu Q , d.h Q => Q'
3. Finde schwächste Voraussetzung P' von Q' unter S1
4. Zeige P => P'

Fallunterscheidung

Abweisende Schleife

{P} **`if (B) then S1 else S2`** {Q]

Idee:

Unabhängig von B muss nach Ausführung die Zusicherung {Q} gelten

Vor Ausführung der Fallunterscheidung gilt die Zusicherung: {P ∧ B) oder {P ∧ ¬ B}

$$\frac{\{P \wedge B\}\ \mathbf{S1}\ \{Q\} \ \wedge\ \{P \wedge \neg B\}\ \mathbf{S2}\ \{Q\}}{\{P\}\ \mathbf{if\ (B)\ S1\ else\ S2}\ \{Q\}}$$

Fallunterscheidung: Beispiel(1)

Zeige:

P ≡ {x==X ∧ y ==Y} // X , Y Werte der Variablen x, y

if (x>y) then y =x else x = y

Q ≡ {x==y ∧ x == max2(X,Y) } // max2: Maximum von zwei Zahlen

Beweisplan:

Zeige unabhängig

1. {P ∧ x > y} y=x {Q}
2. {P ∧ ¬ (x > y)} y=x {Q}
3. Schlussregel anwenden

Fallunterscheidung: Beispiel(2)

1. Schwächste Voraussetzung von Q bezüglich y=x :

   $\{x==X \wedge x == max2\ (X,Y)\} \equiv P'$ (Zuweisungsaxiom)

   $P \equiv \{x ==X \wedge y ==Y\} \wedge \{ x > y \} \Rightarrow \{true\ x == max2\ (X,Y)\} \equiv P'$

2. Analog
3. Schlussregel f. Fallunterscheidung, 1., 2. beweisen Behauptung

Schlussregel nur anwendbar, wenn Auswertung von B keine Seiteneffekte hat

Schleife(1)

Abweisende Schleife

{P} **while (B) S** {Q}

Problem: möglichst schwache Voraussetzung P, möglichst starke Konsequenz Q.
Aber für einen weiteren Schleifendurchgang muss wieder Voraussetzung {P} gelten.
"Vor der Schleife ist nach der Schleife"
Aber: Bei Verlassen der Schleife gilt $\neg$ B, sonst B

I sei Prädikat, das vor immer Auswerten von B gilt (Schleifeninvariante)

Schleife (2)

$$\frac{\{I \wedge B\}\ \mathbf{S}\ \{I\}}{\{I\}\ \mathbf{while}\ (B)\ \mathbf{S}\ \{I \wedge \neg B\}}$$

Schwierigkeit: finden einer geeigneten Invariante I

Einfachste Invariante, die immer gilt: {1==1} ≡ true

Denkaufgabe: was hat Invariante mit Induktionshypothese zu tun?

Schleife (3)

Allgemeiner Beweisplan für:

```
{P}      //Voraussetzung
S1       // hier gilt {I}
while (B) S;
         // hier gilt {I ∧ ¬ B}
S2
{Q}      // Effekt
```

1. Invariante I finden, Invarianz beweisen
2. Invariante mit **S1** herstellen so dass: {P} **S1** {I} gilt (Initialisierung)
3. **S2** finden, so dass {I ∧ ¬ B} **S2** {Q} (Finalisierung)

Schleife (3): Beispiel

Berechne x div y und x rem y (Quotient und Rest)

```
void remQuot (intx, int y){
P ≡ { x=X ≥ 0}
  q = 0 ;
  r = x;
  while (r ≥ y) {
     r = r – y;
     q = q + 1;
  }
}
Q ≡ { 0 ≤ r < y  ∧  x = y*q + r}
```

Invariante I drückt Beziehung zwischen Programmvariablen aus, hier z.B.:
I ≡ x = y*q + r ∧ r ≥ 0

I2 ≡ x = y*(q+1)+r-y ∧ r-y ≥ 0
Zuweisung (schwächste Vor.)
I1 ≡ x = y*(q+1) + r ∧ r ≥ 0
Zuweisung (schwächste Vor.)
I ≡ x = y*q + r ∧ r ≥ 0

I ∧ B ≡ x = y*q + r ∧ r ≥ y => x = y*(q+1)+r-y ∧ r-y ≥ 0 ≡ I2

Schleife (4): Beispiel

Initialisierung:
Zeige: {P} q=0; r=x; {I}

rückschreitend: {I} <= // r=x;
x=q*y+x ∧ x ≥ 0 <= // q=0;
x=0*y +x ∧ x ≥ 0 <= P

Finalisierung: analog

Achtung: nur partielle Korrektheit bewiesen!

Die Voraussetzungen im Beispiel sind zu schwach, um totale Korrektheit zu zeigen!
Annahme y > 0 erforderlich, sonst "Endlosschleife" möglich.

Schleife (5): Terminierung

Idee für Nachweis, dass Schleife terminiert:

Finde eine Funktion der Programmvariablen, die streng monoton abnimmt und nach unten beschränkt ist.

Terminierungsfunktion ("bound function")

Einfachster Fall: Variable i, die Schleifendurchläufe zählt

```
while (i>0) {....; i=i-1};
```

Terminierungsfunktion ("bound function"):

t(z) = i // z drückt Abhängigkeit vom
          // Programmzustand aus

Schleife (5): Terminierung im Beispiel

Stärkere Voraussetzung im Beispiel "remQuot":

$P' = P \wedge \{y>0\}$

Neue Invariante $I' = I \wedge \{y>0\}$

I' invariant, da y nicht verändert wird

Terminierungsfunktion: t(z) = r

streng monoton fallend, da gilt:

$\{y > 0 \wedge r==R\}\ \ r=r-y\ \{r < R\}$

beschränkt, da I => $r \geq 0$

Beispiel: Polynomberechnung nach Horner-Schema (1)

$a_{N-1}\, x^N + \ldots + a_1 x + a_0 = (\ldots(a_{N-1}\, x + a_{N-2})x + \ldots + a_1)x + a_0$

```
float horner (float a[],float x) {
  //requires: N = a.length > 0
  //effect: returns Q ≡ Σ 0 ≤ i ≤ N-1  a[i]x^i
}
```

Konstruktion des Algorithmus:

Variablen **s** für zu berechnende Summe,
n als Feldindex

Anfangswert **s = 0; n = N**

Schleife konstruieren, in der Summe akkumuliert wird.

Beispiel: Verifikation Horner-Schema (2)

Überlegung: nach n Schleifendurchläufen:

$n=N\text{-}1:\ s = a[N\text{-}1]$

$n=N\text{-}2:\ s = a[N\text{-}1]\, x + a[N\text{-}2]$

...

Allgemein: $s = \Sigma_{n \le i \le N\text{-}1}\ a[i]x^{i-n}$

Schleifeninvariante:

$$I \equiv\ s = \Sigma_{n \le i \le N\text{-}1}\ a[i]x^{i-n} \wedge\ \mathbf{0} \le n \le N$$

Beispiel: Verifikation Horner-Schema (3)

Algorithmus:

```
{ N > 0}
s = 0;
n = N;
while (n > 0){
   n = n-1;
   s = s*x + a[n];
}
```

a. Invariante beweisen ...s.u.
b. Finalisierung:
$I \wedge n == 0 \Rightarrow Q$
(einfach in I einsetzen)
c. Initialisierung
wegen
$\Sigma_{N \le i \le N-1} \; a[i]x^i = 0$
und Zuweisungsaxiom
{P} x=e {P[e|x]}:
$s = \Sigma_{n \le i \le N-1} \; a[i]x^i$

$\{Q \equiv s = \Sigma_{0 \le i \le N-1} \; a[i]x^i \}$

Aufgabe: wie sehen Schleifenrumpf, Invariante und Beweis aus, wenn Initialisierung n=N-1 ?

Beispiel: Verifikation Horner-Schema (4)

Invariante beweisen `n=n-1; s = s*x+a[n]`
Vorwärtsbeweis, d.h. $I \wedge B$ `==============>` I
oder zeige: {I ∧ n > 0} n=n-1; s=s*x+a[n] {I} ist wahr.

$$I \wedge n > 0 \Rightarrow s = \Sigma_{n \le i \le N-1} \; a[i]x^{i-n} \wedge \; 0 \le n-1 \le N$$

$$\Rightarrow s*x + a[n-1] = (\Sigma_{n \le i \le N-1} \; a[i]x^{i-n})*x + a[n-1] \wedge 0 \le n-1 \le N$$

$$\Rightarrow s*x + a[n-1] = (\Sigma_{n-1 \le i \le N-1} \; a[i]x^{i-(n-1)}) \wedge 0 \le n-1 \le N$$

***Substitution** gemäß A1 {P} n=n-1{P[n-1/n]} :*

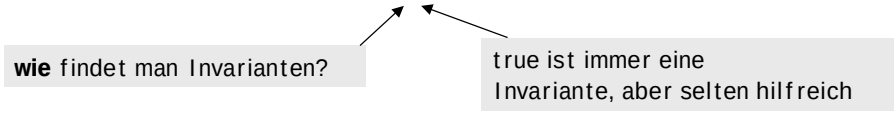
$$s*x + a[n] = (\Sigma_{n \le i \le N-1} \; a[i]x^{i-n}) \wedge 0 \le n \le N$$

***Substitution** gemäß A1 {P} s=s*x+a[n]{P[s*x+a[n] / s]} :*

$$s = (\Sigma_{n \le i \le N-1} \; a[i]x^{i-n}) \wedge 0 \le n \le N$$

Iterative Programme

Schleifen : rückschreitende Verifikation wenig geeignet
stattdessen *geeignete* Invariante finden



Konstruktion von Schleifen

1. Nachbedingung Q festlegen („was ist das Ziel?“)
2. Invariante INV finden
3. Voraussetzung P festlegen und initialisieren, damit INV gilt
4. Bedingung B festlegen
5. Invariante beweisen: „INV gilt auch nach Schleifendurchlauf“
6. Zeigen: INV $\wedge \neg$ B => Q
   Wenn das nicht gelingt, bessere Invariante suchen, weiter bei 2.

Heuristiken zum Finden von Invarianten (1)

A. Konjunktiven Term entfernen in
Nachbedingung Q = Q1 $\wedge$ Q2

Beispiel: Suche von Wert x in Feld a, a.length = N,
x kommt als Feldelement vor

Indexvariable sei i, damit Nachbedingung Q:

$0 \leq i < N \wedge (\forall\ j: 0 \leq j < i : a[j] \neq x) \wedge a[i]=x$

naheliegend: **a[i] = x** entfernen, Rest ist Invariante

Damit auch B gefunden:
Nach Verlassen soll **INV $\wedge$ a[i] = x** gelten, also
B $\equiv$ a[i] != x

Heuristiken zum Finden von Invarianten (2)

B. Konstante durch Variable ersetzen

Beispiel:
Summe der Feldelemente von a mit a.length = N
Nachbedingung Q ≡ s = Σ a[i] : 0 <= i < N
ersetze Konstante N durch Variable j :
INV ≡ s = Σ a[i] : 0 <= i < j

d.h.: "s enthält immer die Summe der ersten j Feldelemente"

Bedingung: INV ∧ ¬ B => Q , ¬ B ≡ j ≥ N oder B = j <N

```
while (j < a.length) { s = s + a[j];}
```

Aufgabe: auch die Konstante 0 hätte durch Variable ersetzt werden können. Wie?

Heuristiken zum Finden von Invarianten (3)

C. Variablenbereich festlegen

Beispiel: Suche das erste Vorkommen von x in Feld a, a.length = n,
x kommt als Feldelement vor (wie A.)
Nachbedingung Q:

$0 \le i < N \wedge (\forall\ j,\ 0 \le j<i:\ a[j] \mathrel{!}= x) \wedge a[i]=x$

Sei K der gesuchte Index a[K]=x
K ist (noch unbekannter) Wert, keine Variable. Wert von K soll nach Durchlaufen der Schleife in Variable j stehen.
Für K gilt also: $0 <= K \wedge (\forall\ j:\ 0<=j < K :\ a[j] \mathrel{!}= x) \wedge a[K] = x$
j nimmt also die Werte $0 \le j \le K$ an
Das ist die Invariante!