Alp2 - Übungsblatt 5 (Aufgabe 3)

Bearbeitet von: Jasmine Cavael & Alexander Chmielus
Tutor: Fabian Halama
Tutorium 10 (Do. 16-18)

Mit diesem Schreibprogramm ist es leider etwas umständlich, mathematische Zeichen einzufügen in den Formeln. Wir werden daher Python-Syntax verwenden:
&& als logisches UND
|| als logisches ODER
! als logische Negation
--> Implikation
== ist gleich (wir benutzen = als Zuweisung, daher der Unterschied.)
!= ist nicht gleich

```
{P} = {b > 0 && a > 0}
counter = 1
power = b
{INV} = {b > 0 && (power == b ** counter) && (a >= counter >= 0)}
while counter < a:
        power = power * b
        counter = counter + 1
{Q} = {power == b ** a}
```

Zuerst beweisen wir die Richtigkeit der Invariante mit Hilfe der **While-Regel**.

```
{INV && B} = {b > 0 && (power == b ** counter) && (a >= counter >= 0) && (a > counter)}
while counter < a:
        power = power * b
        counter = counter + 1
{INV} = {b > 0 && (power == b ** counter) && (a >= counter >= 0)}
```

Zuweisungsaxiom (counter = counter + 1):

{INV3} = {b > 0 && (power == b ** (counter + 1) && (a >= counter + 1 >= 0)

Zuweisungsaxiom (power = power * b):

{INV4} = {b > 0 && ((power * b) == b ** (counter + 1) && (a >= counter + 1 >= 0)

Für die **Konsequenzregel (der stärkeren Vorbedingung)** müssen wir zeigen, dass
{INV && B} --> {INV4}.

(b > 0) steht bei beiden, hat sich also erledigt.
(b > 0) --> (b > 0)

Für ((power * b) == (b ** (counter + 1))) schreiben wir das ein wenig um:
= ((power * b) == ((b ** counter) * b)) und dividieren auf beiden Seiten durch b:
= (power == (b ** counter) und das steht auch in beiden Formeln.
(power == (b ** counter) --> (power == (b ** counter)

Und jetzt zeigen wir, dass (a >= counter + 1 >= 0).
Wir wissen aus {INV && B}, dass (a > counter). Das impliziert, dass a >= counter +1, zumindest in den ganzen Zahlen, wovon wir hier ausgehen. Außerdem wissen wir aus {INV && B}, dass (a >= 0) daraus können wir schließen:
(a > counter && a >= counter >= 0) --> (a >= counter + 1 >= 0)

Die Schleife sieht nun so aus:
{INV && B} = {b > 0 && (power == b ** counter) && (a >= counter >= 0) && (a > counter)}
while counter < a:
    {INV4}
    power = power * b
    [INV3}
    counter = counter + 1
{INV} = {b > 0 && (power == b ** counter) && (a >= counter >= 0)}
, wobei {INV && B] S {INV4} durch die **Konsequenzregel** gültig ist und die gesamte Formel nach der **Sequenzregel.**
Durch die While-Regel ist somit die Korrektheit der Schleife und der Invariante bewiesen.

Als nächstes zeigen wir mit Hilfe der Konsequenzregel, dass {P} S1, S2 {INV} gültig ist:
{P} = {b > 0 && a > 0}
counter = 1
power = b
{INV} = {b > 0 && (power == b ** counter) && (a >= counter >= 0)}

Zuweisungsaxiom (power = b):
{INV1} = {b > 0 && (b == b ** counter) && (a >= counter >= 0)}

Zuweisungsaxiom (counter = 1):
{INV2} = {b > 0 && (b == b ** 1) && (a >= 1 >= 0)} = {b > 0 && TRUE && (a >= 1 >= 0)}
    = {b > 0 && (a >= 1 >= 0)}

Für die **Konsequenzregel (der stärkeren Vorbedingung)** zeigen wir nun, dass {P} -> {INV1}:
(b > 0) --> (b > 0)
Wir wissen, dass a > 0. Da wir hier nur von ganzen Zahlen ausgehen, folgt daraus a >= 1 und 1 >= 0 ist offensichtlich richtig.
{b > 0 && a > 0} --> {b > 0 && (a >= 1 >= 0)}

Der Programmblock sieht nun so aus:

```
{P}
{INV2}
counter = 1
{INV1]
power = b
{INV} = {b > 0 && (power == b ** counter) && (a >= counter >= 0)}
while counter < a:
        power = power * b
        counter = counter + 1
{INV && !B}
```

Nach der Konsequenzregel und der Sequenzregel ist also auch dieser Programmteil mit den dazugehörigen Formeln richtig.

Nach der While-Schleife haben wir:
{INV && !B} = {b > 0 && (power == b ** counter) && (a >= counter >= 0) && (counter >= a)}
Jetzt müssen wir zeigen, dass {INV && !B} --> {Q}. Da counter >= a und a >= counter gelten, muss a == counter sein. Nun ersetzen wir das in (power == b ** counter) und erhalten {Q}.
{(counter >= a && (a >= counter >= 0) && (power == b ** counter)] -> {power == b ** a}

Das Vollständige Programm sieht nun so aus:

```
{P}
{INV2}
counter = 1
{INV1]
power = b
{INV} = {b > 0 && (power == b ** counter) && (a >= counter >= 0)}
while counter < a:
        {INV4}
        power = power * b
        {INV3}
        counter = counter + 1
{INV && !B}
{Q}
```

Wobei sämtliche Formeln {X} gültig sind. Durch die Implikationen der Konsequenzregeln und der Sequenzregel ist die Gültigkeit des Programms {P} S {Q} bewiesen.