RHEINISCH-
WESTFÄLISCHE
TECHNISCHE
HOCHSCHULE
AACHEN

Lehrstuhl für Hochleistungsrechnen
RWTH Aachen · D-52074 Aachen · GERMANY
http://programmierung.informatik.rwth-aachen.de

Prof. Christian Bischof, Ph.D.
Andre Vehreschild, Jakob T. Valvoda

Übung *Programmierung WS 06/07* – Blatt 13

Lösungen müssen bis zum **5. Februar 2007, 17:00 Uhr** in den Kasten Ihrer Übungsgruppe eingeworfen werden. Sie finden die Kästen am Eingang Halifaxstr. des Informatikzentrums (Ahornstr. 55). Bitte vergessen Sie nicht, die **Nummer Ihrer Übungsgruppe**, sowie die **Namen** und **Matrikelnummern** der Mitglieder Ihrer 2er-Gruppe auf jedes Lösungsblatt Ihrer Abgabe zu schreiben. Bitte heften Sie ihre Blätter zusammen.

Geben Sie bitte die erstellten Haskell- und Prolog-Programme sowohl in gedruckter Form, als auch per E-Mail an Ihren Tutor ab. Vergessen Sie auch bei der elektronischen Abgabe per E-Mail nicht die Angabe der **Nummer Ihrer Übungsgruppe**, sowie die jeweiligen **Namen** und **Matrikelnummern**.

Aufgabe 1 (1 + 1 + 1 = 3 Punkte)

a) Gibt es eine Möglichkeit, ein Programm, welches eine `for`-Schleife enthält, mit dem Hoare-Kalkül zu verifizieren? Begründen Sie Ihre Antwort in eigenen Worten.

b) Diskutieren Sie die folgende Aussage zur logischen Programmierung: „Fakten bestehen aus Klauseln und Regeln“. Falls sie richtig ist, dann erläutern Sie die einzelnen Begriffe. Sollte sie falsch sein, dann ändern Sie die Aussage und erläutern Sie die einzelnen Begriffe.

c) Was bezeichnet der Begriff Inferenz? Wozu wird Inferenz in Prolog verwendet?

Aufgabe 2 (5 Punkte)

Gegeben sei das folgende Java-Programm P zur Berechnung des Rests der ganzzahligen Division zweier natürlicher Zahlen x und y, d.h. `rest` $= x$ `mod` y.

```
mult = 0;
rest = x;
while( rest >= y ) {
    rest = rest - y;
    mult = mult + 1;
}
return rest;
```

Vervollständigen Sie die Verifikation des Algorithmus P im Hoare-Kalkül, indem Sie die unterstrichenen Teile (falls erforderlich) ergänzen. Hierbei dürfen zwei Zusicherungen nur dann direkt untereinander stehen, wenn die untere aus der oberen folgt. Hinter einer Programmanweisung darf nur dann eine Zusicherung stehen, wenn sie aus einer Regel des Hoare-Kalküls folgt. Geben Sie bei jedem Schritt den Namen der verwendeten Hoare-Regel an.

Hinweis: Der Rest einer ganzzahligen Division $x \,\texttt{div}\, y = z$ kann ausgedrückt werden als $\texttt{rest} = x - y \cdot z$, wobei $0 \leq \texttt{rest} < y$ gelten muss.

Vorbedingung: $\langle\, x \geq 0 \,\rangle$

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
mult = 0;
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
rest = x;
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
while( rest >= y ) {
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
    rest = rest - y;
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
    mult = mult + 1;
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
}
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

Nachbedingung: $\langle\, \texttt{rest} = x \,\texttt{mod}\, y \,\rangle$

```
return rest;
```

Aufgabe 3 (1 + 5 = 6 Punkte)

Gegeben sei folgendes Java-Programm P, welches für eine ganze Zahl x das Quadrat $k = x^2$ berechnet:

```
i = 0;
j = -1;
k = 0;
while( i < x ) {
    i = i + 1;
    j = j + 2;
    k = k + j;
}
return k;
```

Durch „scharfes Hinsehen“ lassen sich ganz einfach die folgenden Bedingungen für die Schleifeninvariante ableiten:

$$j = 2i - 1 \quad \wedge \quad k = i^2 \quad \wedge \quad i \leq x$$

a) Üben Sie das „scharfe Hinsehen“ zum Ermitteln der Schleifeninvarianten, indem Sie die Werte aller betroffenen Variablen nach jeder Iteration der `while`-Schleife berechnen. Geben Sie hierfür die Werte bei Schleifenantritt und nach jeder Iteration der Schleife in einer Tabelle an. Führen Sie die Analyse für den Wert $x = 5$ vollständig durch und versuchen Sie, aus den Werten die obigen Zusammenhänge abzuleiten.

b) Vervollständigen Sie die Verifikation des Algorithmus im Hoare-Kalkül, indem Sie die unterstrichenen Teile (falls erforderlich) ergänzen. Hierbei dürfen zwei Zusicherungen nur dann direkt untereinander stehen, wenn die untere aus der oberen folgt. Hinter einer Programmanweisung darf nur dann eine Zusicherung stehen, wenn sie aus einer Regel des Hoare-Kalküls folgt. Geben Sie bei jedem Schritt den Namen der verwendeten Hoare-Regel an.

Hinweise:

- Verwenden Sie zur Ableitung der Regeln in der Schleife die Binomische Formel $(a+b)^2 = a^2 + 2ab + b^2$.
- Sie können das Zuweisungsaxiom für die Herleitung von Teilen Ihres Beweises auch rückwärts anwenden. Dieses Vorgehen ist in vielen Fällen einfacher, falls eine Nachbedingung angegeben werden kann. Hierbei fangen Sie mit der entsprechenden Nachbedingung an und arbeiten sich schrittweise von Hinten nach Vorne durch. Im Anschluss wird die Beweisskizze dann wiederum vorwärts überprüft.

  Das Zuweisungsaxiom wird rückwärts wie folgt angewendet: Sei $\langle\varphi\rangle$`var=ausdruck;`$\langle\psi\rangle$, und ψ gegeben. φ wird nun ermittelt, indem alle Vorkommen von `var` in ψ durch `ausdruck` ersetzt werden.

  Beispiel: Aus $\langle\varphi\rangle$`x=z+1;`$\langle x \geq 0 \wedge z + y = x^2\rangle$ folgt $\langle\varphi\rangle = \langle z + 1 \geq 0 \wedge z + y = (z+1)^2\rangle$.

Vorbedingung: $\langle\ x \geq 0\ \rangle$

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
i = 0;
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
j = -1;
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
k = 0;
```

$\langle$ _____ $\rangle$

$\langle\ j = 2i - 1 \wedge k = i^2 \wedge i \leq x\ \rangle$ Schleifeninvariante

```
while( i < x ) {
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
    i = i + 1;
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
    j = j + 2;
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
    k = k + j;
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
}
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

Nachbedingung: $\langle\ k = x^2\ \rangle$

```
return k;
```

Aufgabe 4 (1 + 1 + 1 + 1 + 1 = 5 Punkte)

Gegeben seinen die folgenden Fakten, welche das Patenprogramm und das Tierleben im Aachener Tierheim beschreiben (siehe `www.tierheim-aachen.de`). Im Tierheim sind einzelne Pfleger für bestimmte Tierarten verantwortlich. Ehrenamtliche Paten betreuen zusätzlich einzelne Tiere und gehen z.B. mit Hunden spazieren. Desweiteren enthält die Wissensbasis Informationen über das Verhalten der Tiere untereinander:

```
tier(sandy,hund). tier(walli,hund). tier(patch,hund). tier(candy,katze).
tier(funky,katze). tier(pauline,haengebauchschwein). tier(otis,katze).
tier(brutus,maus). tier(hilde,maus). tier(mork,degu). tier(nexi,degu).
tier(elsa,hase).

pfleger(alex). pfleger(sandra). pfleger(nicole). pfleger(sonja).

verantwortlich(alex,hund). verantwortlich(sandra,katze). verantwortlich(sandra,maus).
verantwortlich(sandra,degu). verantwortlich(sonja,haengebauchschwein).

pate(kathrin). pate(mark). pate(rebecca).

betreut(kathrin,walli). betreut(mark,sandy). betreut(mark,hilde).
betreut(rebecca,otis). betreut(rebecca,mork).

hatAngstVor(katze,hund). hatAngstVor(maus,katze). hatAngstVor(hase,hund).
hatAngstVor(degu,hund). hatAngstVor(degu,katze).
```

a) Formulieren Sie für das einstellige Prädikat `mensch` geeignete Regeln, welche besagen, dass Paten und Pfleger Menschen sind. Formulieren Sie eine Anfrage, die prüft, ob `hilde` ein Mensch ist.

b) Erweitern Sie das zweistellige Prädikat `betreut` um eine Regel, welche ausdrückt, dass ein Pfleger ein Tier betreut, wenn er für die Tierart verantwortlich ist. Formulieren Sie eine Anfrage, die prüft, ob `pauline` betreut wird.

c) Formulieren Sie für das zweistellige Pradikat `fuerchtet` eine Regel, welche besagt, dass sich ein Tier T_1 vor einem anderen Tier T_2 fürchtet, wenn die Tierart von T_1 vor der Tierart von T_2 Angst hat. Formulieren Sie eine Anfrage, die alle Tiere ausgibt, welche sich vor `walli` fürchten. Fürchtet `pauline` irgendjemanden? Formulieren Sie diese Anfrage.

d) Ein Tier ist froh, wenn es einen Paten hat, der sich darum kümmert. Formulieren Sie für das einstellige Prädikat `froh` eine Regel, welche besagt, ob ein Tier froh ist. Ist `pauline` froh? Formulieren Sie eine geeignete Anfrage.

e) Die Pfleger müssen Kontakt zu denjenigen Paten aufbauen, die ein Tier derjenigen Tierart betreuen, für die der Pfleger verantwortlich ist. Formulieren Sie Regeln für das zweistellige Prädikat `kontakt`, welche dies aussagen. Zu welchen Paten muss `sandra` einen Kontakt aufbauen?

Hinweis: Sie finden die Wissensbasis in der Datei `tierheim.pl` auf den Webseiten der Vorlesung.

Aufgabe 5 (1 + 1 + 1 + 2* = 3 Punkte + 2* Bonuspunkte)

Formulieren Sie Prädikate, welche folgendes berechnen:

a) das Maximum von zwei Zahlen,

b) das Maximum von drei Zahlen,

c) den Betrag einer Zahl,

d)* (Bonusaufgabe)
die rekursive Funktion $f(x, y) = z$ mit

$$f(x,y) = \begin{cases} y+1 & \text{falls } x = 0 \wedge y \geq 0 \\ f(x-1,1) & \text{falls } y = 0 \wedge x > 0 \\ f(x-1, f(x,y-1)) & \text{sonst} \end{cases}$$

Hinweis: Verwenden Sie zur Definition der Funktion f das in Prolog vordefinierte Prädikat `is` um arithmetische Ausdrücke auszuwerten.

Beachten Sie ebenfalls, dass die Funktion $f(x, y)$ extrem schnell wächst. Zum Testen eignen sich deshalb Werte von $0 \leq x \leq 3$. Zum Beispiel ist $f(3, 2) = 29$, aber bereits $f(4, 2) \approx 2 \cdot 10^{19\,728}$. Der Wert von y kann hingegen in einem gewissen Rahmen variiert werden.

Aufgabe 6 (6* + 3* = 9* Bonuspunkte)

Die Punkte dieser Aufgabe zählen nicht zu der erforderlichen Gesamtpunktzahl.

a) Implementieren Sie folgende Funktionen **in Haskell**:

- `primes :: [Int]`, welche die unendliche Liste von Primzahlen berechnet.
  Hinweis: gehen Sie hierbei nach dem Verfahren des Eratosthenes (Übung 4, Aufgabe 3) vor. Implementieren Sie eine Funktion, welche eine unendliche Liste der natürlichen Zahlen darstellt und streichen Sie sukzessive jeweils die Vielfachen der aktuellen Primzahl aus der Liste raus. Benutzen Sie hierfür die vordefinierte Funktion `filter`.
- `isPrim :: Int -> Bool`, welche prüft, ob eine Zahl eine Primzahl ist. Verwenden Sie hierfür die zuvor implementierte unendliche Liste von Primzahlen (`primes`).
- `prime :: Int -> Int`, welche die $n$te Primzahl berechnet.
- mit `take n primes` kann eine Liste mit den n ersten Primzahlen erzeugt werden. Implementieren Sie eine Funktion `takePrimes :: Int -> [Int]`, welche eine Zahl n übergeben bekommt und eine Liste mit allen Primzahlen bis zu dieser Zahl berechnet. Handelt es sich bei n selbst um eine Primzahl, dann soll diese ebenfalls in der Liste enthalten sein.

b) Implementieren Sie den folgenden Primzahlentest **in Prolog**:

- Implementieren Sie ein zweistelliges Prädikat `nichtteilbar(X,Y)`, das genau dann wahr ist, wenn die Zahl `Y` kein Teiler von `X` ist.
- Implementieren Sie ein zweistelliges Prädikat `primTest(X,T)`, welches wahr ist, wenn eine Zahl `X` nicht durch die Zahlen $\{2, \ldots, \texttt{T}\}$ geteilt werden kann.
- Implementieren Sie schliesslich das Prädikat `prim(X)`, welches wahr ist, wenn `X` eine Primzahl ist. Sie können (und sollten) hier die obigen Prädikate verwenden.

Hinweis: Ungleichheit kann in Prolog mit dem Operator `=\=` getestet werden.