



Musterlösungen zum Übungsblatt 13

Lösung

1. Semantik rekursiver Funktionsdeklarationen

Man beachte das $g(x,y) \stackrel{!}{=} x^y$ ist, da 0^0 nicht definiert ist. $g(x,y)$ ist also eine Fortsetzung der Funktion $f(x,y) = x^y$ auf $\mathbb{N}^\perp \times \mathbb{N}^\perp$.

Berechnung einiger Folgenglieder von (exp_n) und Überführung in eine geschlossene Form, um eine Annahme für eine geschlossene Form zu erhalten.

(Hier wurden einige Zwischenschritte weggelassen, die jedoch anzugeben wären, wenn in einer Aufgabenstellung die Angabe des Lösungswegs gefordert ist.)

Berechnung von exp_0 : $\text{exp}_0(x,y) = \perp$, für alle $x,y \in \mathbb{N}$ (nach Definition) *Berechnung von exp_1 :*

$$\text{exp}_1(x,y) = \begin{cases} \perp, & \text{falls } x = \perp \vee y = \perp \\ 1, & \text{falls } x \neq \perp \wedge y = 0 \\ x * \perp, & \text{falls } x \neq \perp \wedge y > 0 \end{cases}$$

$$\text{exp}_1(x,y) = \begin{cases} \perp, & \text{falls } x = \perp \vee y = \perp \vee y > 0 \\ 1, & \text{falls } x \neq \perp \wedge y = 0 \end{cases}$$

$$= \begin{cases} \perp, & \text{falls } x = \perp \vee y = \perp \vee y \geq 1 \\ g(x,y), & \text{falls } x \neq \perp \wedge y < 1 \text{ (für } y < 1, \text{ also } y = 0 \text{ ist } g(x,y) = 1) \wedge y = 0 \end{cases}$$

Berechnung von exp_2 :

$$\text{exp}_2(x,y) = \begin{cases} \perp, & \text{falls } x = \perp \vee y = \perp \\ 1, & \text{falls } x \neq \perp \wedge y = 0 \\ x * 1, & \text{falls } x \neq \perp \wedge y - 1 > 0 \wedge y - 1 = 0 \\ x * \perp, & \text{falls } x \neq \perp \wedge y - 1 = 0 \end{cases}$$

$$= \begin{cases} \perp, & \text{falls } x = \perp \vee y = \perp \vee y > 1 \\ 1, & \text{falls } x \neq \perp \wedge y = 0 \\ x, & \text{falls } x \neq \perp \wedge y = 1 \end{cases}$$

$$= \begin{cases} \perp, & \text{falls } x = \perp \vee y = \perp \vee y \geq 2 \\ g(x,y), & \text{falls } x \neq \perp \wedge y < 2 \end{cases}$$

Hieraus ergibt sich die folgende Annahme für exp_n für $n \in \mathbb{N}$ und für $x, y \in \mathbb{N}^\perp$:

$$\text{exp}_n(x,y) = \begin{cases} \perp, & \text{falls } x = \perp \vee y = \perp \vee y \geq n \\ g(x,y), & \text{falls } x \neq \perp \wedge y < n \end{cases} \quad (*)$$

Induktionsanfang: $n = 0$

Für $x, y \in \mathbb{N}^\perp$ gilt:

$$\text{exp}_0(x, y) = \{\text{Def.}\} \perp$$

$$= \begin{cases} \perp, & \text{falls } x = \perp \vee y = \perp \vee y \geq 0 \\ x^y, & \text{falls } x \neq \perp \vee y < 0 \end{cases} = \{y^3 0\} \perp$$

Induktionsannahme: Für beliebiges, aber festes $n \in \mathbb{N}$ und $x, y \in \mathbb{N}^\perp$ gelte (*).

Induktionsschluß:

Zu zeigen: Für n aus der Induktionsannahme und $x, y \in \mathbb{N}^\perp$ gilt:

$$\text{exp}_{n+1}(x, y) = \begin{cases} \perp, & \text{falls } x = \perp \vee y = \perp \vee y \geq n + 1 \\ x^y, & \text{falls } x \neq \perp \wedge y < n + 1 \end{cases}$$

$$\text{exp}_{n+1}(x, y) = \{\text{Def. von } \text{exp}_{n+1}\} \tau[\text{exp}_n](x, y)$$

$$= \{\text{Def. von } \tau\} \begin{cases} \perp, & \text{falls } x = \perp \vee y = \perp \\ 1, & \text{falls } x \neq \perp \wedge y = 0 \\ x * \text{exp}_n(x, y - 1), & \text{falls } x \neq \perp \wedge y > 0 \end{cases}$$

$$= \{\text{I.V.}\} \begin{cases} \perp, & \text{falls } x = \perp \vee y = \perp \\ 1, & \text{falls } x \neq \perp \wedge y = 0 \\ x * \perp, & \text{falls } x \neq \perp \wedge y > 0 \wedge (x = \perp \vee y - 1 = \perp \vee y - 1 \geq n) \quad (*1) \\ x * g(x, y-1), & \text{falls } x \neq \perp \wedge y > 0 \wedge x \neq \perp \wedge y - 1 < n \quad (*2) \end{cases}$$

{ ist strikt; Vereinfachung der Bedingungen}*

$$= \begin{cases} \perp, & \text{falls } x = \perp \vee y = \perp \\ 1, & \text{falls } x \neq \perp \wedge y = 0 \\ \perp, & \text{falls } x \neq \perp \wedge y \geq n + 1 \\ g(x, y), & \text{falls } x \neq \perp \wedge y > 0 \wedge y < n + 1 \end{cases}$$

{Zusammenfassen des 1. und 3. Falls und des 2. und 4. Falls}

$$= \begin{cases} \perp, & \text{falls } x = \perp \vee y = \perp \vee y \geq n + 1 \\ g(x, y), & \text{falls } x \neq \perp \wedge y < n + 1 \end{cases}^{(*)3}$$

Ⓡ Behauptung

(*1) Zusammenfassen der Bedingungen:

$$x \neq \perp \wedge y > 0 \wedge (x = \perp \vee y - 1 = \perp \vee y - 1 = n) \text{ gdw.}$$

$$x \neq \perp \wedge [(y > 0 \wedge x = \perp) \vee (y > 0 \wedge y - 1 = \perp) \vee (y > 0 \wedge y - 1 \geq n)] \text{ gdw.}$$

$$x \neq \perp \wedge [(y > 0 \wedge x = \perp) \vee \text{false} \vee y \geq n + 1] \text{ gdw.}$$

$$(x \neq \perp \wedge y > 0 \wedge x = \perp) \vee (x \neq \perp \wedge y \geq n + 1) \text{ gdw.}$$

$$\text{false} \vee (x \neq \perp \wedge y \geq n + 1) \text{ gdw.}$$

$$x \neq \perp \wedge y \geq n + 1 \quad (*2) \text{ Wegen } y > 0 \text{ ist } x * g(x, y - 1)$$

$$= x * \begin{cases} 1, & \text{falls } y = 0 \\ x^{y-1} & \text{sonst} \end{cases} = x * x^{y-1} = x^y$$

(*3) Zusammenfassen des 2. und 4. Falls:

Für $y = 0$ ist $g(x, y) = 1$ (2. Fall), also können die beiden Fälle zusammengefaßt werden.

Die gemeinsame Bedingung für x und y ergibt sich wie folgt:

$$(x \neq \perp \wedge y = 0) \vee (x \neq \perp \wedge y > 0 \wedge y < n + 1) \text{ gdw.}$$

$$x \neq \perp \wedge [(y = 0 \vee x \neq \perp) \wedge (y = 0 \vee y > 0) \wedge (y = 0 \vee y < n + 1)] \text{ gdw.}$$

$$x \neq \perp \wedge [(y = 0 \vee x \neq \perp) \wedge \text{true} \wedge y < n + 1] \text{ gdw.}$$

$$x \neq \perp \wedge (y = 0 \vee x \neq \perp) \wedge y < n + 1 \text{ gdw.}$$

$$[(x \neq \perp \wedge y = 0) \vee x \neq \perp] \wedge y < n + 1 \text{ gdw.}$$

$$(x \neq \perp \wedge y = 0) \vee (x \neq \perp \wedge y < n + 1) \text{ gdw.}$$

$$x \neq \perp \wedge (y = 0 \vee y < n + 1) \text{ gdw.}$$

$$x \neq \perp \wedge (y < n + 1)$$

Eine geschlossene Form für $\text{exp}(x, y) = \text{exp}_\infty(x, y)$ ergibt sich, wenn man die Einschränkung für den Wertebereich von y ($y < n + 1$) aufhebt. Dann erhält man:

$$\text{exp}(x, y) = \begin{cases} \perp, & \text{falls } x = \perp \vee y = \perp \\ g(x, y), & \text{falls } x \neq \perp \wedge y \neq \perp \end{cases}$$

Lösung

2. Beweis mittels Parameterinduktion

Wir verwenden das Prinzip der Parameterinduktion für den Parameter y , um zunächst

(+) $p \cdot \text{exp}(m, n) = \text{exp_embed}(m, n, p)$ für alle $m, n, p \in \mathbb{N}$

zu zeigen.

Induktionsanfang: $n = 0$:

$p \cdot \text{exp}(m, 0) = p \cdot 1 = p$ für alle $m, p \in \mathbb{N}$.

$\text{exp_embed}(m, 0, p) = p$ für alle $m, p \in \mathbb{N}$.

In beiden Funktionsdeklarationen tritt jeweils der Fall $y = 0$ bei der Fallunterscheidung ein. So ist 1 bzw p das Ergebnis.

→ Die Behauptung ist korrekt für $n = 0$.

Induktionshypothese: (+) gelte für ein beliebiges, aber festes $n \in \mathbb{N}$ und alle $m, p \in \mathbb{N}$

Induktionsschluß für $n + 1$:

Wegen $n > 0$ tritt der else-Fall ein. So ist mit $x = m$ und $y = n + 1$:

$$p \cdot \text{exp}(m, n + 1) = p \cdot (m \cdot \text{exp}(m, n)) = (p \cdot m) \cdot \text{exp}(m, n)$$

$$\stackrel{(\text{I.V.})}{=} \text{exp_embed}(m, n, p \cdot m) \text{ für alle } m, p \in \mathbb{N} (*)$$

Außerdem ist

$$\text{exp_embed}(m, n+1, p) \stackrel{\text{else-Fall}}{=} \text{exp_embed}(m, n, p \cdot m) \text{ für alle } m, p \in \mathbb{N} (**)$$

Aus (*) und (**) folgt sofort (+). Mit $p = 1$ folgt aus (+) die Behauptung.

Lösung

3. Terminierungsbeweise für rekursive Funktionsdeklarationen

Zunächst muß eine Funktion $h : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ mit der folgenden Eigenschaft gefunden werden:

$h(x, y)$ ist obere Schranke für die Anzahl der rekursiven Aufrufe von $f(x, y)$. (1)

$h(x, y)$ ist monoton fallend über den Argumenten der rekursiven Aufrufe von f , d.h. in diesem Fall:

$$h(x, y) > h(x, (x + y) \div 2) \text{ und (2a)}$$

$$h(x, y) > h(1 + (x+y) \div 2, y). \text{ (2b)}$$

für alle $(x, y) \in \mathbb{N}^2$

Sind diese Bedingungen erfüllt, so läßt sich die Terminierung von f für $(x, y) \in \mathbb{N}^2$ mit vollständiger Induktion leicht zeigen.

Annahme für die Abstiegsfunktion h :

Für $x \geq y$ terminiert f offensichtlich sofort. Die Rekursionstiefe ist 0.

Andernfalls terminiert f nur dann, wenn der Abstand zwischen beiden Argumenten (also ihre Differenz) immer kleiner wird, damit der Basisfall $x = y$ oder $x \geq y$ erreicht wird.

Daher bietet sich folgende Definition für die Funktion h an, für die die Eigenschaften (1), (2a) und (2b) noch nachgewiesen werden müssen:

$h : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ durch

$$h(x, y) = \begin{cases} y - x, & \text{falls } x < y \\ 0, & \text{falls } x \geq y \end{cases}$$

für alle $(x, y) \in \mathbb{N}^2$

Bemerkung: Eine geeignete Abstiegsfunktion h findet sich

lediglich durch Probieren oder Intuition. Für die hier gewählte Abstiegsfunktion ist noch nicht klar, dass sie wirklich die Eigenschaften (1), (2a) und (2b) erfüllt. Man kann es lediglich vermuten und muß dies im Folgenden noch nachweisen.

Diese Eigenschaften sollen nun mit vollständiger Induktion bewiesen werden:

1. Induktionsanfang:

zu zeigen: $h(x, y) = 0 \Rightarrow f(x, y)$ terminiert ohne einen rekursiven Schritt (also auch $f(x, y) \neq \perp$)

Für alle $(x, y) \in \mathbb{N}^2$ gilt:

$$h(x, y) = 0 \stackrel{(\text{Def. von } h)}{\Rightarrow} x = y \stackrel{(\text{Def. von } f)}{\Rightarrow} f(x, y) = 1 \neq f(x, y) = x$$

Für $h(x, y) = 0$ terminiert f also sofort ohne einen rekursiven Schritt.

Es gilt also erst recht: $f(x, y) \neq \perp$

2. Induktionsannahme

Für ein beliebiges, aber festes $k \in \mathbb{N}$ und $(x, y) \in \mathbb{N}^2$ gelte:

$h(x, y) \leq k \Rightarrow f(x, y)$ terminiert in höchstens $h(x, y)$ Schritten, also natürlich auch $f(x, y) \neq \perp$.

3. Induktionsschluß:

zu zeigen: unter Voraussetzung der Induktionsannahme gilt nun

$h(x, y) \leq k+1 \Rightarrow f(x, y)$ terminiert in höchstens $h(x, y)$ Schritten, also natürlich auch $f(x, y) \neq \perp$, für $(x, y) \in \mathbb{N}^2$

Um den Induktionsschluß also auf die Induktionsannahme zurückführen zu können, muss gezeigt werden, dass nach Ausführung eines rekursiven Schritts $h(u, v) \leq k$ ist, wobei u und v die Argumente für x und y sind, die f beim rekursiven Aufruf übergeben werden. Es müssen also die Eigenschaften (2a) und (2b) gezeigt werden.

Damit folgt dann sofort die Terminierung von f nach höchstens $k+1$ Schritten, also: $f(x, y) \neq \perp$ für alle $(x, y) \in \mathbb{N}^2$

Beweis von (2a):

Wir betrachten lediglich den Fall $x < y$, da der andere Fall schon im Induktionsanfang behandelt wurde. Für $(x, y) \in \mathbb{N}^2$ gilt:

$$h(x, y) = \{x < y\} y - x = (2 * y) / 2 - x > \{x < y\} (x + y) / 2 - x \stackrel{3}{=} (x + y) \div 2 - x = h(x, (x + y) \div 2)$$

Beweis von (2b):

Wir betrachten lediglich den Fall $x < y$, da der andere Fall schon im Induktionsanfang behandelt wurde. Für $(x, y) \in \mathbb{N}^2$ gilt:

$$h(x, y) = \{x < y\} y - x = y - (2 * x) / 2 > \{x < y\} y - (y + x) / 2 \stackrel{3}{=} y - (x + y) \div 2 \stackrel{3}{=} y - (1 + (x + y) \div 2) = h(1 + (x + y) \div 2, y)$$

→ Behauptung

Bemerkung:

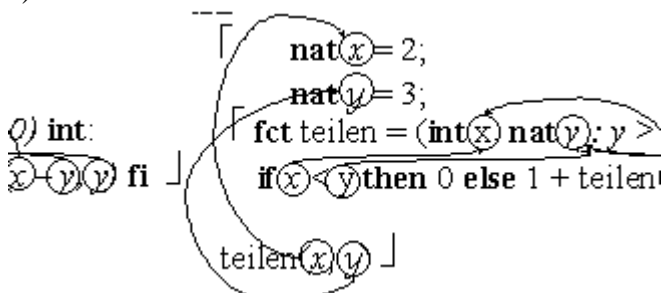
Im Allgemeinen genügt es für Terminierungsbeweise zum einen die Monotonie der Abstiegsfunktion h über der Folge der Argumente von f bei den rekursiven Aufrufen zu zeigen (hier: Eigenschaften (2a) und (2b)), und zum anderen nachzuweisen, dass in einem Fall die Rekursionstiefe von f majorisiert (hier: Induktionsanfang). Daraus folgt dann sofort, dass h die Rekursionstiefe von f generell majorisiert (hier: Eigenschaft (1)) und dass f terminiert.

Das hier gewählte etwas umständlichere Beweisverfahren verdeutlicht, warum der Beweis dieser drei Eigenschaften genügt.

Lösung

4. Gültigkeit und Lebensdauer (I)

a)



b)

| | Lebensdauer | | | | Gültigkeit | | | |
|--|-------------|---|----|----|------------|---|----|----|
| | x | y | x* | y* | x | y | x* | y* |
| nat x = 2; | • | | | | • | | | |
| nat y = 3; | • | • | | | • | • | | |
| fct teilen = (int x, nat y : y > 0) int: | • | • | • | • | | | • | • |
| if x < y then 0 else 1 + teilen(x - y, y) fi | • | • | • | • | | | • | • |
|] | • | • | • | • | | | • | • |
| teilen(x, y)] | • | • | | | • | • | | |

Lösung

5. Java: Schiffe versenken

Teilaufgabe 1:

Für das Spielfeld soll das zweidimensionale 8x8-Feld `brett` mit Integerwerten verwendet werden. Im Pseudocode wird sein Inhalt mit `brett [Spalte] [Zeile]` angesprochen.

Als Kodierung für die Zustände wird

Schiff = 1

Treffer = 2 (Bezeichnung für den Zustand "schon versenkt") und

Wasser = 3

gewählt.

Pseudocode:

Setze alle Felder zu Wasser;

Setze Schiffe auf gegebene Positionen;

Merke Anzahl der gesetzten Schiffsteile

Solange Anzahl der noch nicht getroffenen Schiffsteile > 0

Lies Spaltenindex als Buchstabe ein;

Wandle Spaltenindex in Zahl "spalte" um;
Lies Zeilenindex als Zahl "zeile" ein;
Falls brett[spalte] [zeile] = Schiff dann

Ausgabe: "Wasser!"

Sonst

Falls brett [spalte] [zeile] = Treffer

Ausgabe: "Hier wurde schon einmal getroffen."

Sonst { Es ist brett [spalte] [zeile] = Schiff }

Ausgabe: "Treffer!"

brett[spalte] [zeile] = Treffer

Reduziere die Anzahl der noch nicht getroffenen Schiffsteile um 1

Ausgabe: "Alles versenkt."

Teilaufgabe 2:

```
import info1.*;
```

```
public class Schiffe {
```

```
    // Zustaende eines Feldes
```

```
    static final int SCHIFF = 1;
```

```
    static final int TREFFER = 2;
```

```
    static final int WASSER = 3;
```

```
    // Symbolische Konstanten fuer Spaltenkoordinaten
```

```
    static final int A = 0;
```

```
    static final int B = 1;
```

```
    static final int C = 2;
```

```
    static final int D = 3;
```

```
    static final int E = 4;
```

```
    static final int F = 5;
```

```
    static final int G = 6;
```

```
    static final int H = 7;
```

```
    public static void main(String[] args ) {
```

```
        // Spielbrett anlegen
```

```
        int[][] brett = new int[8][8];
```

```
        // Spielbrett enthaelt zunaechst keine Schiffe
```

```
        for(int spalte = A; spalte <= H; spalte++)
```

```
            for(int zeile = 0; zeile <=7; zeile++)
```

```
                brett[spalte][zeile] = WASSER;
```

```
        // Schiffe setzen
```

```
        brett[F][0] = SCHIFF;
```

```
        brett[B][1] = SCHIFF;
```

```
        brett[C][1] = SCHIFF;
```

```
        brett[D][1] = SCHIFF;
```

```

brett[E][1] = SCHIFF;
brett[C][2] = SCHIFF;
brett[F][2] = SCHIFF;
brett[A][3] = SCHIFF;
brett[F][3] = SCHIFF;
brett[F][4] = SCHIFF;
brett[B][5] = SCHIFF;
brett[C][5] = SCHIFF;
brett[D][5] = SCHIFF;
brett[B][6] = SCHIFF;
brett[C][6] = SCHIFF;

// Zaehler für noch nicht versenkte Schiffteile
int zaehler = 15;

char c; // Spaltenindex als Buchstabe
int spalte; // Spaltenindex als Zahl
int zeile; // Zeilenindex

// Solange noch nicht alle Schiffe versenkt sind
while ( zaehler > 0 ) {

    // Spalte als Buchstaben einlesen
    System.out.print("Buchstabe (a-f): ");
    c = Console.in.readWord().charAt( 0 );
    // Spaltenindex in Zahl umwandeln
    spalte = c - 'a';
    // Zeile als Ziffer einlesen; Array geht von 0..7
    System.out.print("Ziffer (1-8): ");
    zeile = Console.in.readInt() - 1;

    // Schuss auswerten
    if(brett[spalte][zeile] == WASSER )
        System.out.println("Wasser!");
    else
        if (brett[spalte][zeile]==TREFFER)
            System.out.println("Hier wurde schon einmal getroffen.");
        else{ //brett[spalte][zeile]==Schiff
            System.out.println("Treffer! ");
            zaehler--;
            // Treffer vermerken
            brett[spalte][zeile] = TREFFER;
        }
    } // Schleife zu Ende
    System.out.println("Alle Schiffe versenkt. ");
}
}

```
