

Lösungshinweise/-vorschläge zum Übungsblatt 3: Grundlagen der Programmierung (WS 2023/24)

Aufgabe 1 Peano und Leibniz Entwurfsmuster (Präsenzaufgabe)

Motivation: Diese Aufgabe soll Sie auf die beiden folgenden Einreichaufgaben vorbereiten. Dazu schauen wir uns einige Beispielanwendungen des Peano und Leibniz Entwurfsmusters an und beleuchten die Unterschiede der beiden Ansätze. Sie können sich an den Vorlesungsfolien 200 bis 234 sowie an den Kapiteln 3.7.1 und 3.7.2 im Skript orientieren.

Schreiben Sie Ihre Lösungen in die Datei `Entwurfsmuster.fs` aus der Vorlage `Aufgabe-3-1.zip`.

- a) Wiederholen Sie Peano und Leibniz Entwurfsmuster. Wo liegen die Unterschiede der beiden Entwurfsmuster? Warum verwenden wir diese?

Peano, linear viele Schritte

```
let rec f (n: Nat): t =  
  if n = 0N then ...  
  else ... f (n - 1N) ...
```

Leibniz, logarithmisch viele Schritte

```
let rec f (n: Nat): t =  
  if n = 0N then ...  
  else ... n % 2N ... f (n / 2N) ...
```

Skript: „Rekursion birgt die Gefahr der Nichtterminierung. Entwurfsmuster bannen diese Gefahr und unterstützen Programmierende bei der systematischen Lösung von Problemen.“

- b) Schreiben Sie eine Funktion `add: Nat -> Nat`, die alle natürlichen Zahlen bis `n` aufaddiert. Gehen Sie nach Peano Entwurfsmuster vor.

Beispiele:

`add 0N = 0N`

`add 2N = 3N`

`add 3N = 6N`

`add 10N = 55N`

```
let rec add (n: Nat): Nat =  
  if n = 0N then 0N  
  else n + add (n - 1N)
```

- c) Schreiben Sie eine rekursive Funktion $\text{mod5}: \text{Nat} \rightarrow \text{Nat}$, die für eine gegebene Zahl n den Rest der Ganzzahldivision durch 5 berechnet. Es soll also $\text{mod5 } n = n \% 5$ gelten. Gehen Sie nach Peano Entwurfsmuster vor. Verwenden Sie *nicht* die Funktionen $/$ oder $\%$.

Beispiele:

$\text{mod5 } 0N = 0N$ $\text{mod5 } 2N = 2N$ $\text{mod5 } 4N = 4N$ $\text{mod5 } 19N = 4N$ $\text{mod5 } 21N = 1N$
 $\text{mod5 } 1N = 1N$ $\text{mod5 } 3N = 3N$ $\text{mod5 } 5N = 0N$ $\text{mod5 } 20N = 0N$ $\text{mod5 } 22N = 2N$

```

let rec mod5 (n: Nat): Nat =
  if n = 0N then 0N
  else
    let m = mod5 (n - 1N)
    if m = 4N then 0N
    else m + 1N

```

- d) Schreiben Sie eine rekursive Funktion $\text{mult42}: \text{Nat} \rightarrow \text{Nat}$, die die übergebene Zahl n mit 42 multipliziert. Sie dürfen in Ihrer Lösung jedoch **nicht die Multiplikationsfunktion $*$ verwenden**. Gehen Sie stattdessen **nach Leibniz Entwurfsmuster** vor, sodass die **Berechnung in logarithmisch vielen Schritten** erfolgt.

Beispiele:

$\text{mult42 } 0N = 0N$ $\text{mult42 } 1N = 42N$ $\text{mult42 } 2N = 84N$ $\text{mult42 } 5N = 210N$

```

let rec mult42 (n: Nat): Nat =
  if n = 0N then 0N
  else
    let r = mult42 (n / 2N)
    r + r + if n % 2N = 0N then 0N else 42N

```

- e) Schreiben Sie eine Funktion $\text{count5}: \text{Nat} \rightarrow \text{Nat}$, die zählt, wie oft die Ziffer 5 in einer Zahl vorkommt. Gehen Sie nach Leibniz Entwurfsmuster vor.

Hinweis: Auch wenn das Leibniz Entwurfsmuster in der Vorlesung mit Divisor 2 eingeführt wurde, dürfen Sie wenn nötig einen anderen Divisor wählen. Wichtig ist lediglich, dass sowohl bei der Division als auch beim Modulo-Operator der selbe Divisor verwendet wird.

Beispiele:

$\text{count5 } 3N = 0N$ $\text{count5 } 76567N = 1N$ $\text{count5 } 1234N = 0N$ $\text{count5 } 445566N = 2N$

```

let rec count5 (n: Nat): Nat =
  if n = 0N then 0N
  else
    if n % 10N = 5N then 1N else 0N
    +
    count5 (n / 10N)

```

Aufgabe 2 Peano Entwurfsmuster (Einreichaufgabe, 6 Punkte)

Motivation: In dieser Aufgabe sollen Sie das Peano Entwurfsmuster einüben. Sie können sich an den Vorlesungsfolien 200 bis 217 sowie am Skript Kapitel 3.7.1 orientieren.

Schreiben Sie Ihre Lösungen in die Datei `peano.fs` aus der Vorlage `Aufgabe-3-2.zip`.

Anmerkung: Ihre Implementierung soll dem Peano Entwurfsmuster folgen, Sie sollen nicht die `peano-pattern` Funktion von Folie 215 verwenden.

- a) Schreiben Sie eine rekursive Funktion `mult3: Nat -> Nat`, die für eine gegebene Zahl n den Wert $3n$ berechnet, jedoch ohne dabei die F# eigene Multiplikationsfunktion `*` zu verwenden.

Beispiele:

$$\text{mult3 } 0N = 0N$$

$$\text{mult3 } 1N = 3N$$

$$\text{mult3 } 5N = 15N$$

```
let rec mult3 (n: Nat): Nat =
    if n = 0N then 0N
    else
        3N + mult3 (n - 1N)
```

- b) Schreiben Sie eine rekursive Funktion `divide3: Nat -> Nat`, die für eine gegebene natürliche Zahl x den Wert $x \div 3$ (Ganzzahldivision) berechnet, jedoch ohne dabei die F# eigene Divisionsfunktion `/` zu verwenden. Die Verwendung der Multiplikationsfunktion ist hingegen gestattet.

Beispiele:

$$\text{divide3 } 0N = 0N$$

$$\text{divide3 } 1N = 0N$$

$$\text{divide3 } 6N = 2N$$

$$\text{divide3 } 8N = 2N$$

```
let rec divide3 (x: Nat): Nat =
    if x = 0N then 0N
    else
        let m = divide3 (x - 1N)
        if m * 3N + 3N <= x
            then m + 1N
            else m
```

Aufgabe 3 Leibniz Entwurfsmuster (Einreichaufgabe, 6 Punkte)

Motivation: In dieser Aufgabe sollen Sie das Leibniz Entwurfsmuster einüben. Sie können sich an den Vorlesungsfolien 219 bis 234 sowie am Skript Kapitel 3.7.2 orientieren.

Schreiben Sie Ihre Lösungen in die Datei Leibniz.fs aus der Vorlage Aufgabe-3-3.zip.

Hinweis: Wie schon in der Präsenzaufgabe erwähnt muss der Divisor im Leibniz Entwurfsmuster nicht unbedingt 2 sein.

- a) Schreiben Sie eine rekursive Funktion `quersumme: Nat -> Nat`, welche die Summe der Ziffern einer natürlichen Zahl `n` berechnet.

Beispiele:

`quersumme 0N = 0N`

`quersumme 123N = 6N`

`quersumme 1234N = 10N`

`quersumme 4711N = 13N`

```
let rec quersumme (n: Nat): Nat =
  if n = 0N then 0N
  else (n % 10N) + quersumme (n / 10N)
```

- b) Schreiben Sie eine rekursive Funktion `sortedDigits: Nat -> Bool`, die für eine gegebene Zahl `n` prüft, ob die Ziffern von `n` (in Zehnerdarstellung) aufsteigend sortiert sind.

Beispiele:

`sortedDigits 0N = true`

`sortedDigits 5N = true`

`sortedDigits 159N = true`

`sortedDigits 1111N = true`

`sortedDigits 1101N = false`

`sortedDigits 543N = false`

```
let rec sortedDigits (n: Nat): Bool =
  if n = 0N then true
  else
    let firstDigitsSorted = sortedDigits (n / 10N)
    let lastDigit = n % 10N
    let secondLastDigit = (n / 10N) % 10N
    lastDigit >= secondLastDigit && firstDigitsSorted
```

Aufgabe 4 Dynamische Semantik (Einreichaufgabe, 5 Punkte)

Motivation: Sie sollen anhand einiger Beispiele die Regeln der dynamischen Semantik für Funktionen üben. Sie können sich für diese Aufgabe an den Vorlesungsfolien 188 bis 198 sowie am Skript Kapitel 3.6 orientieren.

Werten Sie die folgenden Mini-F#-Ausdrücke mit den Regeln der **dynamischen Semantik** aus und geben Sie einen entsprechenden Beweisbaum an.

Tipp: Sie können Ihre Ergebnisse mit dem F# Interpreter überprüfen.

Werten Sie den Ausdruck `(fun x -> g x)(let x = 5 in x)` bezüglich der folgenden Umgebung aus:

$\delta := \{g \mapsto \langle y \mapsto 4, n, (n+y) \rangle\}$

Definiere aus Platzgründen $\delta_1 := \delta, \{x \mapsto 5\}$ und $\delta_2 := \{y \mapsto 4, n \mapsto 5\}$

$$\frac{\delta \vdash (\text{fun } x \rightarrow g \ x) \Downarrow \langle \delta, x, g \ x \rangle \quad \frac{\frac{\overline{\delta \vdash 5 \Downarrow 5}}{\delta \vdash \text{let } x = 5 \Downarrow \{x \mapsto 5\}} \quad \overline{\delta_1 \vdash x \Downarrow 5}}{\delta \vdash \text{let } x = 5 \text{ in } x \Downarrow 5} \quad \frac{\overline{\delta_1 \vdash g \langle \{y \mapsto 4\}, n, n+y \rangle} \quad \overline{\delta_1 \vdash x \Downarrow 5}}{\delta_1 \vdash g \ x \Downarrow 9} \quad \frac{\overline{\delta_2 \vdash n \Downarrow 5} \quad \overline{\delta_2 \vdash y \Downarrow 4}}{\delta_2 \vdash n+y \Downarrow 9}}{\delta \vdash (\text{fun } x \rightarrow g \ x)(\text{let } x = 5 \text{ in } x) \Downarrow 9}$$

c) Werten Sie den Ausdruck g ($\text{fun } (n: \text{Nat}) \rightarrow n*7$) bezüglich der folgenden Umgebung aus:

$$\delta := \{g \mapsto \langle \emptyset, f, (f \ 2N) \rangle\}$$

Definiere aus Platzgründen $\delta_2 := \{f \mapsto \langle \delta, n, n * 7 \rangle\}$

$$\frac{\frac{\delta \vdash g \Downarrow \langle \emptyset, f, f \ 2 \rangle}{\delta \vdash \text{fun } n \rightarrow n*7 \Downarrow \langle \delta, n, n*7 \rangle} \quad \frac{\frac{\delta_2 \vdash f \Downarrow \langle \delta, n, n*7 \rangle}{\delta_2 \vdash 2 \Downarrow 2} \quad \frac{\frac{\frac{\{g \mapsto \langle \emptyset, f, f \ 2 \rangle, n \mapsto 2\} \vdash n \Downarrow 2}{\{g \mapsto \langle \emptyset, f, f \ 2 \rangle, n \mapsto 2\} \vdash 7 \Downarrow 7}}{\{g \mapsto \langle \emptyset, f, f \ 2 \rangle, n \mapsto 2\} \vdash n*7 \Downarrow 14}}{\delta_2 \vdash f \ 2 \Downarrow 14}}{\delta \vdash g \ (\text{fun } n \rightarrow n*7) \Downarrow 14}$$

Aufgabe 6 Regelsammlung Statische und Dynamische Semantik

In der Vorlesung erweitern wir unsere Programmiersprache Mini-F# Schritt für Schritt um neue Konzepte. Dabei werden immer neue Regeln zur statischen und dynamischen Semantik eingeführt. Um Aufgaben wie “Geben Sie einen Beweisbaum an” lösen zu können, müssen Sie die Regeln zur Hand haben. Derartige Aufgaben waren bislang in jeder GdP-Klausur enthalten. Es ist also wahrscheinlich, dass auch Ihre Klausur mindestens eine solche Aufgabe enthalten wird. Wir erlauben Ihnen in der Klausur einen Spickzettel mitzunehmen (mehr dazu zu einem späteren Zeitpunkt), dieser sollte auf jeden Fall die Regeln enthalten.

Legen Sie sich daher schon jetzt eine Zusammenstellung aller bislang eingeführten Regeln an und erweitern Sie diese im Laufe des Semesters kontinuierlich um neu hinzugekommene Regeln. So haben Sie es leichter, Beweisbaum-Aufgaben für die Hausaufgaben zu lösen und haben zudem schon einen Teil der Klausurvorbereitung erledigt.