Übungsblatt 8
Programme und Rekursion

Theoretische Informatik
Studiengang Angewandte Informatik
Wintersemester 2015/2016
Prof. Barbara Staehle, HTWG Konstanz

Aufgabe 8.1

[topic = Berechnung der Subtraktion]

Sei die natürliche Subtraktion"definiert als $- : \mathbf{N}_0 \times \mathbf{N}_0 \to \mathbf{N}_0, (x - y) = \begin{cases} x - y & \text{falls } x \geq y, \\ 0 & \text{falls } x < y. \end{cases}$

Es gilt also z.B. $4 - 3 = 1, 7 - 5 = 2, 2 - 3 = 0, 1 - 9 = 0$.

Teilaufgabe 8.1.1

Erstellen Sie das Loop-Programm sub.loop mit $x_0 := \text{sub}(x_1, x_2)$ welches als Ergebnis $x_1 - x_2$ (wie oben definiert) zurückliefert.
Lösung

```
x0 := x1;
LOOP x2 DO
  x0 := pred(x0)
END
```

Teilaufgabe 8.1.2

Berechnen Sie für den Vektor $\nu = (0, 5, 3)$ und Ihr Programm $P : x_0 := \text{sub}(x_1, x_2)$ das Ergebnis von $\delta(\nu, P)$. Geben Sie insbesondere alle ca. 10 Zwischenschritte Ihrer Berechnung an.
Lösung:
$\delta(v, P)$
$= \delta((0, 5, 3), P)$
$= \delta(\alpha((0, 5, 3), x_0 := x_1),\ LOOP\ x_2\ DO\ pred(x_0)\ END)$

$= \delta(\alpha((0,5,3), x_0 := x_1),\ LOOP\ x_2\ DO\ x_0 :=\ pred(x_0)\ END)$
$= \delta((5,5,3),\ LOOP\ x_2\ DO\ pred(x_0)\ END)$
$= \delta((5,5,3), x_0 := pred(x_0); x_0 := pred(x_0); x_0 := pred(x_0))$
$= \delta(\alpha((5,5,3), x_0 := pred(x_0); x_0 := pred(x_0); x_0 := pred(x_0)))$
$= \delta((4,5,3), x_0 := pred(x_0); x_0 := pred(x_0))$
$= \delta(\alpha((4,5,3), x_0 := pred(x_0); x_0 := pred(x_0)))$
$= \delta((3,5,3), x_0 := pred(x_0))$
$= (2,5,3)$

Teilaufgabe 8.1.3

Geben Sie nun mit Hilfe der vereinfachten Notation, die Sie in der Vorlesung kennengelernt haben, die Zustandsänderungen des Eingabevektors ν bis das Programm beendet ist in den beiden folgenden Fällen an: *Tupel untereinander schreiben ohne =!*

1. $\nu = (0,3,5)$
   Łösung

   $$(0,3,5) \overset{initial\ x_0=x_1}{=} (3,3,5) \overset{x2=5}{=} (2,3,5) \overset{x2=4}{=} (1,3,5) \overset{x2=3}{=} (0,3,5) \overset{x2=2}{=} (0,3,5) \overset{x2=1}{=} (0,3,5)$$

2. $\nu = (0,5,0)$
   Łösung
   x_2 ist $0 \rightarrow$ LOOP läuft 0 mal

   $$(0,5,0) \overset{initial\ x_0=x_1}{=} (5,5,0)$$

Hinweis: Geben Sie alle Zustandsänderungen an, auch wenn sich der Nachfolgezustand nicht vom Vorgängerzustand unterscheidet.

Teilaufgabe 8.1.4

[topic = Subtraktion als primitiv-rekursive Funktion, credits = 1]

Die Subtraktion ist auch eine primitiv rekursive Funktion. Wenn Sie $\text{sub}(x_1, x_2)$ als primitiv-rekursive Funktion schreiben möchten, welche Sachverhalte könnten Sie sich zu Nutze machen, bzw. welche Ideen erscheinen Ihnen nützlich?

Hinweis: Es ist **NICHT** verlangt, dass Sie die Subtraktion tatsächlich als primitiv-rekursive Funktion angeben.

Lösung:
$sub(x_1, x_2) = sub(x_1, x_2 - 1) - 1$

Aufgabe 8.2

[topic = Berechnung der Fakultät]

Bevor Sie mit der Bearbeitung der Aufgaben beginnen, machen Sie sich anhand der Beispiele $4!, 5!, 7!$ klar, wie viele Multiplikationen Sie für die Berechnung von $n!$ jeweils durchführen müssen.

Teilaufgabe 8.2.1

[topic = factorial.loop, credits = 3]

Erstellen Sie das Loop-Programm factorial.loop mit $x0 :=$ factorial$(x1)$, welches als Ergebnis $x_1!$ zurückliefert.

Hinweise:

- Es ist eventuell von Vorteil, während des Programmablaufs die Hilfsvariable x_2 zu verwenden.
- Achten Sie darauf, dass $1! = 0! = 1$ korrekt berechnet wird.

Lösung:

```
x0 := 1;
x2 = x1-1;
LOOP x2 DO
  x0 := mult(x0,x1)
  x1 := pred(x1)
END
```

Teilaufgabe 8.2.2

[topic = Arbeitsweise von factorial.loop, credits = 1]

Vollziehen Sie anhand der Veränderungen im Eingabevektor $\nu = (0, 3, 0)$ die Arbeitsweise Ihres Programms für die Berechnung von 3! nach.

Łösung

$$(0,3,0)(1,3,3)(3,3,3)(3,3,2)(3,3,2)(6,3,2)(6,3,1)(6,3,1)(6,3,0)$$

Teilaufgabe 8.2.3

[topic = factorial.while, credits = 2]

Erstellen Sie das While-Programm factorial.while mit $x_0 :=$ factorial(x_1), welches als Ergebnis $x_1!$ zurückliefert. Erstellen Sie Ihre Programm als echtes While-Programm ohne die Verwendung der Loop-Schleife.

Hinweis: Achten Sie darauf, dass $1! = 0! = 1$ korrekt berechnet wird.

Lösung:

```
x0 := 1;
WHILE x1 DO
  x0 := mult(x0,x1)
  x1 := pred(x1)
END
```

Teilaufgabe 8.2.4

[topic = Arbeitsweise von factorial.while, credits = 1]

Vollziehen Sie anhand der Veränderungen im Eingabevektor $\nu = (0, 3)$ die Arbeitsweise Ihres Programms für die Berechnung von 3! nach.

Lösung

$$(0,3,0)(1,3,3)(3,3,3)(3,3,2)(3,3,2)(6,3,2)(6,3,1)(6,3,1)$$

Teilaufgabe 8.2.5

[topic = Fakultät als primitiv-rekursive Funktion, credits = 1]

Die Fakultät ist eine primitiv rekursive Funktion. Wenn Sie factorial(x_1, x_2) als primitiv-rekursive Funktion schreiben möchten, welche Sachverhalte könnten Sie sich zu Nutze machen, bzw. welche Ideen erscheinen Ihnen nützlich?

Hinweis: Es ist **NICHT** verlangt, dass Sie die Fakultät tatsächlich als primitiv-rekursive Funktion angeben.