

Übungen zu Informatik I (Lösungsvorschlag)

Aufgabe 5-1 Bildungsgesetze für Zahlenfolgen (keine Abgabe)

- a) Das Bildungsgesetz für diese Folge ist das folgende: $a_0 = 0$ und $a_{n+1} = 2a_n + 1$. In SML kann man die zugehörige Funktion wie folgt definieren:

```
fun f1 n = if n=0 then 0 else 2*f1(n-1) + 1;
```

- b) Das Bildungsgesetz dieser Folge ist das Folgende: $a_0 = a_1 = 1$ und für $n \geq 2$ gilt $a_n = a_{n-2} + a_{n-1} + 1$. Die entsprechende SML-Funktion ist wie folgt definiert:

```
fun f2 n = if n<=1 then 1 else f2(n-2) + f2(n-1) + 1;
```

- c) Diese Folge gehorcht folgendem Bildungsgesetz: $a_0 = 1$ und für $n \geq 1$ gilt

$$a_n = \begin{cases} 2 * a_{n-1} & \text{falls } a_{n-1} \text{ ungerade} \\ a_{n-1} + 1 & \text{falls } a_{n-1} \text{ gerade} \end{cases}$$

Die diesem Bildungsgesetz entsprechende Funktion kann man in SML wie folgt definieren:

```
fun f3 n = if n=0 then 1
           else if f3(n-1) mod 2 = 0 then f3(n-1) + 1
           else 2*f3(n-1);
```

Aufgabe 5-2 Partielle Exponentialreihe (keine Abgabe)

Zunächst müssen wir zwei Funktionen $pot: \mathbf{int} \rightarrow \mathbf{real} \rightarrow \mathbf{real}$ und $fak: \mathbf{int} \rightarrow \mathbf{int}$ mit folgenden Eigenschaften definieren: $pot(k)$ ist eine Funktion, die zu einer reellen Zahl x die k -te Potenz von x , also x^k berechnet und fak ist die Fakultäts-Funktion, d. h. sie berechnet zu einer natürlichen Zahl k die Zahl $k!$.

Bei der Definition von $partexp$ muss man etwas mehr aufpassen. Da der Operator $/ : \mathbf{real} \times \mathbf{real}$ zwei reelle Zahlen als Argumente erwartet, muss im Rekursionsschritt die Zahl $fak(k)$ (vom Typ \mathbf{int}) erst nach \mathbf{real} konvertiert werden.

Insgesamt erhält man folgende Definitionen:

(* Fakultätsfunktion *)

```
fun fak k = if k=0 then 1 else k*fak(k-1);
```

(* Potenzierung *)

```
fun pot k = fn x:real => if k=0 then 1.0 else x*pot (k-1) x;
```

(* Partielle Exponentialreihe *)

```
fun partexp(k) = fn x => if k=0 then 1.0
                        else partexp(k-1)(x) + (pot(k)(x)/real(fak(k)));
```

In SML kann die Definition von Funktionen höherer Ordnung auch kompakter geschrieben werden (vgl. Skriptum Seite 31). Mit dieser Schreibweise erhält man die folgende (äquivalente) Lösung:

```
(* Fakultätsfunktion *)
fun fak k = if k=0 then 1 else k*fak(k-1);

(* Potenzierung *)
fun pot k x = if k=0 then 1.0 else x*pot (k-1) x;

(* Partielle Exponentialreihe *)
fun partexp k x = if k=0 then 1.0
  else partexp (k-1)x + (pot k x)/real(fak k);
```

Aufgabe 5-3

Sequenzen von Basen

(keine Abgabe)

a) aca , $tacgat$ sind die einzigen.

Man kann aca wie folgt ableiten:

$$\begin{aligned} \langle Seq \rangle &\rightsquigarrow a \langle M \rangle \\ &\rightsquigarrow a \langle Seq \rangle a \\ &\rightsquigarrow a \{g|c\}^* a \\ &\rightsquigarrow a \{g|c\} a \\ &\rightsquigarrow aca. \end{aligned}$$

Für $tacgat$ erhält man:

$$\begin{aligned} \langle Seq \rangle &\rightsquigarrow t \langle N \rangle \\ &\rightsquigarrow t \langle Seq \rangle t \\ &\rightsquigarrow ta \langle M \rangle t \\ &\rightsquigarrow ta \langle Seq \rangle at \\ &\rightsquigarrow ta \{g|c\}^* at \\ &\rightsquigarrow ta \{g|c\} \{g|c\} at \\ &\rightsquigarrow tac \{g|c\} at \\ &\rightsquigarrow tacgat. \end{aligned}$$

b) Für $w \in \Sigma^*$ bezeichne w^R die Umkehrung des Wortes w . Damit erhalten wir $\mathcal{L}(\langle Seq \rangle) = \{www^R \mid w \in \{a, t\}^* \text{ und } u \in \{c, g\}^*\}$.

Idee: Bei den Nicht-Terminalzeichen $\langle N \rangle$ und $\langle M \rangle$ gibt es jeweils nur eine Möglichkeit, mit der Ableitung fortzufahren. Die Produktionsregeln können also verkürzt werden zu $\langle Seq \rangle ::= a \langle Seq \rangle a \mid t \langle Seq \rangle t \mid \{g|c\}^*$, die Nicht-Terminalzeichen $\langle N \rangle$ und $\langle M \rangle$ können dann entfallen. Nun sieht man, dass zu jedem a bzw. t am Anfang des Wortes in symmetrischer Manier ein a bzw. t am Ende des Wortes gehört. Aus dem $\langle Seq \rangle$ in der Mitte kann eine beliebige Folge der Zeichen c und g erzeugt werden.

Aufgabe 5-4**Gleichheit bis auf ϵ**

(1 + 2 Punkte)

Die Funktion *rabs*, berechnet den Absolutbetrag einer reellen Zahl. Die Funktion *epseq* bedient sich der Funktion *rabs*:

```
(* Teilaufgabe a *)
fun rabs(a) = if a < 0.0 then ~a else a;

(* Teilaufgabe b *)
fun epseq(eps) (a, b) = rabs(a-b) < eps;
```

Aufgabe 5-5**Zahldarstellung**

(1 + 2 + 2 Punkte)

Die folgenden drei Funktionsdefinitionen lösen alle drei Teilaufgaben:

```
(* Teilaufgabe a *)
fun ziffer n =
  if (n < 10) then chr (n + 48)
  else chr (n + 55);

(* Teilaufgabe b *)
fun binaer n =
  if (n = 0) then ""
  else (binaer (n div 2)) ^ (str (ziffer (n mod 2)));

fun hexadezimal n =
  if (n = 0) then ""
  else (hexadezimal (n div 16)) ^ (str (ziffer (n mod 16)));

(* Teilaufgabe c *)
fun darst p n =
  if (n = 0) then ""
  else (darst p (n div p)) ^ (str (ziffer (n mod p)));
```

Aufgabe 5-6**Nochmal Basen**

(3 + 1 Punkte)

Wir betrachten das Alphabet $\Sigma = \{a, c, g, t\}$.

- a) Wir setzen $G = (\Sigma, V, S, P)$ wobei $V = \{\langle Str \rangle, \langle A \rangle, \langle B \rangle\}$, $S = \langle Str \rangle$ und P die Produktionsregeln P die Produktionsregeln

$$\begin{aligned} \langle Str \rangle &::= [\{a|c|g\}\langle Str \rangle|t\langle A \rangle] \\ \langle A \rangle &::= [\{a|c|g\}\langle Str \rangle|t\langle B \rangle] \\ \langle B \rangle &::= [\{a|c|g\}\langle Str \rangle] \end{aligned}$$

umfasst.

Idee: Wenn wir ein t erzeugt haben, machen wir weiter mit $\langle A \rangle$. Kommt ein zweites t , fahren wir bei der Erzeugung des Wortes mit $\langle B \rangle$ fort. Das Nicht-Terminalzeichen $\langle B \rangle$ erlaubt nun nicht, ein weiteres t zu erzeugen.

b) Eine Ableitung ist von $ttatt$ ist wie folgt:

$$\begin{aligned}
\langle Str \rangle &\rightsquigarrow [\{a|c|g\}\langle Str \rangle|t\langle A \rangle] \\
&\rightsquigarrow t\langle A \rangle \\
&\rightsquigarrow t[\{a|c|g\}\langle Str \rangle|t\langle B \rangle] \\
&\rightsquigarrow tt\langle B \rangle \\
&\rightsquigarrow tt[\{a|c|g\}\langle Str \rangle] \\
&\rightsquigarrow tt\{a|c|g\}\langle Str \rangle \\
&\rightsquigarrow tta\langle Str \rangle \\
&\rightsquigarrow tta[\{a|c|g\}\langle Str \rangle|t\langle A \rangle] \\
&\rightsquigarrow ttat\langle A \rangle \\
&\rightsquigarrow ttat[\{a|c|g\}\langle Str \rangle|t\langle B \rangle] \\
&\rightsquigarrow ttatt\langle B \rangle \\
&\rightsquigarrow ttatt[\{a|c|g\}\langle Str \rangle] \\
&\rightsquigarrow ttatt.
\end{aligned}$$