HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Blatt 6

Automaten und formale Sprachen Praktikum

Teammitglieder: Luis Staudt
Dominik Meurer

1

1.1

	0	**1**	**2**	**3**	**4**	**...**
0	0	1	3	6	10	
1	2	4	7	11	16	
2	5	8	12	17	23	
3	9	13	18	24	31	
4	14	19	25	32	39	
5	20	26	33	40	47	
6	27	34	41	48	55	
...						

Horizontale 0 Zeile
$f_0(n) = \frac{n(n+1)}{2}$

1.1.1 Gesamte Funktion

$$f(n,m) = \frac{(n+m)(n+m+1)}{2} + m$$

1.2

Es kann wie ein Zahlensystem mit 3 Zeichen gesehen werden, das in das Dezimalsystem übersetzt wird.

$$\begin{aligned}
a &:= 1 \\
b &:= 2 \\
c &:= 3 \\
\Sigma^* &= c_n * 3^n + c_{n-1} * 3^{n-1} + ... + c_1 * 3^1 + c_0 * 3^0 \\
aaabbbaaac &= 1 * 3^9 + 1 * 3^8 + 1 * 3^7 + 2 * 3^6 + 2 * 3^5 + 2 * 3^4 + 1 * 3^3 + 1 * 3^2 + 1 * 3^1 + 3 * 3^0 \\
&= 19683 + 6561 + 2187 + 1458 + 486 + 162 + 27 + 9 + 3 + 3 \\
&= 30579
\end{aligned}$$

$$\Sigma^* \to \mathbb{N}$$
$$a \to 1$$
$$b \to 2$$
$$c \to 3$$
$$aa \to 4$$
$$ab \to 5$$
$$ac \to 6$$
$$ba \to 7$$

1.3

	0	1	2	3	4	...		
0	$f_0(0)$	$f_0(1)$	$f_0(2)$	$f_0(3)$	$f_0(4)$			gerade is 0
1	$f_1(0)$	$f_1(1)$	$f_1(2)$	$f_1(3)$	$f_1(4)$			ungerade ist 0
2	$f_2(0)$	$f_2(1)$	$f_2(2)$	$f_2(3)$	$f_2(4)$			durch 5 teilbar ist 0
3	$f_3(0)$	$f_3(1)$	$f_3(2)$	$f_3(3)$	$f_3(4)$			...
4	$f_4(0)$	$f_4(1)$	$f_4(2)$	$f_4(3)$	$f_4(4)$			
5	$f_5(0)$	$f_5(1)$	$f_5(2)$	$f_5(3)$	$f_5(4)$			
6	$f_6(0)$	$f_6(1)$	$f_6(2)$	$f_6(3)$	$f_6(4)$			
...								
g	$f_0(0)+1$	$f_1(1)+1$	$f_2(2)+1$	$f_3(3)+1$	$f_4(4)+1$			hier gilt $1+1=0$

die funktion g ist per definition nicht oben enthalten. Daher kann die Tabelle nicht vollständig sein. Aus dem Wiederspruch folgt, dass die Menge aller Funktionen nicht abzählbar ist.

2

2.1

```
whilenot iszero(A) do # Setzt A auf 0
  pred(A);
od;

# Inkrementiert A auf 5
succ(A);
```

```
succ(A);
succ(A);
succ(A);
succ(A);
```

Code Listing 1: Inkrementieren (A $\bar{5}$)

2.2

```
whilenot iszero(B) do # Dekrementiert B auf 0 und A hoch (A = A + B)
  succ(A);
  pred(B);
od;
```

Code Listing 2: Addition (A + B)

2.3

```
whilenot iszero(C) do # Setzt C auf 0
  pred(C);
od;

whilenot iszero(A) do # Addiert A auf B und kopiert A nach C
  succ(B);
  pred(A);
  succ(C);
od;

whilenot iszero(C) do # Restauriert A aus C
  pred(C);
  succ(A);
od;
```

Code Listing 3: Kopieren (A nach B)

2.4

```
whilenot iszero(B) do # Setzt B auf 0
  pred(B);
od;

whilenot iszero(A) do # Dekrementiert A auf 0, B ist nach jedem Schritt 1
  pred(A);
  pred(B);
  succ(B);
od;
```

Code Listing 4: Signum-(Vorzeichen-)Test

2.5

```
whilenot iszero(C) do # Setzt C auf 0
  pred(C);
od;

whilenot iszero(B) do # Dekrementiert B auf 0 und dekrementiert A
  pred(A);
  pred(B);
od;

whilenot iszero(A) do # Dekrementiert A auf 0, C ist nach jedem Schritt 1
  pred(A);
  pred(C);
  succ(C);
od;
```

Code Listing 5: Vergleichstest (A > B)

3

3.1

```
whilenot iszero(B) do # Dekrementiert B auf 0 und dekrementiert A (A = A - B)
  pred(A);
  pred(B);
od;
```

Code Listing 6: Subtraktion (A - B)

3.2

```
whilenot iszero(C) do # Setzt C auf 0
  pred(C);
od;

whilenot iszero(B) do # Dekrementiert B auf 0 und addiert A zu C in jedem Schritt
  whilenot iszero(A) do # Addiert A zu C und inkrementiert Temp
    succ(C);
    pred(A);
    succ(Temp);
  od;

  # A ist hier 0

  whilenot iszero(Temp) do # Restauriert A aus Temp
    succ(A);
    pred(Temp);
  od;
```

```
18
19   pred(B); # B wird um 1 dekrementiert nach jedem Schritt
20 od;
```

Code Listing 7: Multiplikation (A * B)

3.3

3.3.1

```
whilenot iszero(Temp) do # Setzt Temp auf 0
  pred(Temp);
od;

succ(Temp); # Temp auf 1 setzen
whilenot iszero(Temp) do # Dekrementiert A und B, bis eines von beiden 0 ist
  if iszero(A) then
    pred(Temp);
  fi;
  if iszero(B) then
    pred(Temp);
  fi;

  pred(A);
  pred(B);
od;

# Wenn A und B 0 sind, dann sind sie gleich
if iszero(A) then
  if iszero(B) then
    succ(C);
  fi;
fi;
```

Code Listing 8: Testen auf Gleicheit

3.3.2

```
copy(R1, R2) # Kopiert R1 nach R2
  whilenot iszero(R_Copy) do # Setzt R_Copy auf 0
    pred(R_Copy);
  od;

  whilenot iszero(R1) do # Kopiert R1 nach R2 und R_Copy
    succ(R2);
    pred(R1);
    succ(R_Copy);
  od;

  whilenot iszero(R_Copy) do # Restauriert R1 aus R_Copy
    pred(R_Copy);
    succ(R1);
  od;

whilenot iszero(C) do
  pred(C);
od;

copy(A, A_Copy);
copy(B, B_Copy);

whilenot iszero(A) do # Dekrementiert A auf 0 (A wird 0, B unbekannt)
  pred(A);
  pred(B);
od;

whilenot iszero(B_Copy) do # Dekrementiert B auf 0 (A unbekannt, B_Copy wird 0)
  pred(B_Copy);
  pred(A_Copy);
od;

# Wenn A und B_Copy 0 sind, dann sind A und B gleich
if iszero(B) then
  if iszero(A_Copy) then
    succ(C);
  fi;
fi;
```

Code Listing 9: Testen auf Gleicheit -> Komplex aber erster Versuch

3.4

```
whilenot iszero(Temp) do # Setzt Temp auf 0
  pred(Temp);
od;

succ(Temp); # Setzt Temp auf 1
whilenot iszero(Temp) do # Dekrementiert A und B, bis eines von beiden 0 ist
    if iszero(A) then # Temp wird 0, da A 0 ist
    pred(Temp);
  fi;
  if iszero(B) then # Temp wird 0, da B 0 ist
    pred(Temp);
  fi;

  pred(A);
  pred(B);
od;

if iszero(A) then # Wenn A 0 ist, dann ist B der Abstand
  whilenot iszero(B) do # Dekrementiert B und C, bis B 0 ist
    pred(B);
    succ(C);
  od;
else # Wenn B 0 ist, dann ist A der Abstand
  whilenot iszero(A) do # Dekrementiert A und C, bis A 0 ist
    pred(A);
    succ(C);
  od;
fi;
```

Code Listing 10: Abstand