Automaten und formale Sprachen Blatt 6

Jan Lucca Agricola (275867) & Jakob Schulz (275258)

2. Dezember 2024

1 Aufgabe

1.1

$f : \mathbb{Z} \times \mathbb{N}\backslash\{0\} \to \mathbb{N}$ mit:

$$f(m,n) = \begin{cases} \frac{1}{2}(2*m+n)*(2*m+n+1)+2*m & \text{für } m \geq 0 \\ \frac{1}{2}(-2*m-1+n)*(-2*m-1+n+1)-2*m-1 & \text{für } m < 0 \end{cases}$$

1.2

$f : \Sigma^* \to \mathbb{N}$ mit :
$w \in \Sigma^*$
$\Sigma' =$ Die Elemente von Σ in lexikographischer Ordnung
Für jedes $b \in \Sigma$ den Rang von b in Σ' (beginnend bei 1) mit der Größe des Alphabets (Σ) hoch die Position von b in w (beginnend bei 0)
Addiere das Produkt zum Ergebnis.
Das leere Wort (ϵ) hat 0 als Ergebnis

Beispiel
$\Sigma = \{a, b, c\}$

$f : \Sigma^* \to \mathbb{N}$

- $\epsilon \to 0$
- $a \to 1 * 3^0$
- $b \to 2 * 3^0$
- $c \to 3 * 3^0$
- $aa \to 1 * 3^1 + 1 * 3^0 = 4$
- $ab \to 1 * 3^1 + 2 * 3^0 = 5$
- $ac \to 1 * 3^1 + 3 * 3^0 = 6$
- $ba \to 2 * 3^1 + 1 * 3^0 = 7$
- ...

1.3

Beweisskizze:
Angenommen die Menge der totalen Funktionen $f : \mathbb{N} \to \{0, 1\}$ wäre abzählbar. Hieraus folgt, dass es eine Aufzählung dieser Funktionen geben muss. Diese könnte schematisch wie folgt aussehen:

	0	1	2	3	4	...
f_0	$f_0(0)$	$f_0(1)$	$f_0(2)$	$f_0(3)$	$f_0(4)$	...
f_1	$f_1(0)$	$f_1(1)$	$f_1(2)$	$f_1(3)$	$f_1(4)$	...
f_2	$f_2(0)$	$f_2(1)$	$f_2(2)$	$f_2(3)$	$f_2(4)$	...
f_3	$f_3(0)$	$f_3(1)$	$f_3(2)$	$f_3(3)$	$f_3(4)$	...
...	...	...	...	...	...	...

Gemäß Annahme müssten in dieser Aufzählung alle Funktionen $f : \mathbb{N} \to \{0, 1\}$ enthalten sein. Wir betrachten nun die Funktion $g : \mathbb{N} \to \{0, 1\}$, die wie folgt definiert wird:

	0	1	2	3	4	...
g	$f_0(0) + 1$	$f_1(1) + 1$	$f_2(2) + 1$	$f_3(3) + 1$	$f_4(4) + 1$	...

Die Funktion g wird in Abhängigkeit der Funktionen f_0, f_1, f_2, f_3, ... definiert. Beispielsweise liefert die Funktion $g(0)$ den Wert $f_0(0) + 1$. Dies bedeutet, dass die Funktion g nicht identisch mit der Funktion f_0 ist, da sich der Funktionswert von g und f_0 in dem Argument 0 unterscheidet. Mit analoger Begründung unterscheidet sich g von den Funktionen f_0, f_1, f_2, f_3, ... in den weiteren Argumenten. Das bedeutet, dass die Funktion g nicht in der Aufzählung vorkommen kann. Dies ist jedoch ein Widerspruch zur Annahme, dass die Aufzählung alle Funktionen $g : \mathbb{N} \to \{0, 1\}$ enthält. Die Menge aller Funktionen $f : \mathbb{N} \to \{0, 1\}$ ist überabzählbar.

2 Aufgabe

2.1

```
1 whilenot iszero(A) do
2     pred(A);
3 od;
4 succ(A);
```

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

2.2

```
whilenot iszero(B) do
    succ(A);
    pred(B);
od;
```

2.3

```
whilenot iszero(A) do
    succ(B);
    succ(C);
    pred(A);
od;
whilenot iszero(C) do
    succ(A);
    pred(C);
od;
```

2.4

Annahme, dass A nicht negativ sein kann, also entweder 0 oder positiv

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

2.5

```
whilenot iszero(C) do
    pred(C);
od;
succ(C); //C auf 1 setzen

succ(B); //damit C auch 0 ist, wenn A = B
//A auf 0 setzen, B verringern
whilenot iszero(A) do
    pred(B);
    pred(A);
od;

//Wenn B ungleich 0 ist, dann ist B >= A
whilenot iszero(B) do
    pred(C);
    pred(A);
od;
```

3 Aufgabe

3.1

```
whilenot iszero(B) do
    pred(A);
    pred(B);
od;
```

3.2

```
//C und D auf 0 setzen
whilenot iszero(C) do
    pred(C);
od;
whilenot iszero(D) do
    pred(D);
od;
```

```
8
9  //Wert von A in C und C kopieren
10 whilenot iszero(A) do
11     succ(C);
12     succ(D);
13     pred(A);
14 od;
15
16 //Für jedes B addieren den Wert von C auf A
17 whilenot iszero(B) do
18     whilenot iszero(C) do
19         succ(A);
20         succ(D);
21         pred(C);
22     od;
23
24     whilenot iszero(D) do
25         succ(C);
26     od;
27     pred(B);
28 od;
```

3.3

```
1  //Register C, D, E, F, G auf 0 setzen
2  whilenot iszero(C) do
3      pred(C);
4  od;
5  whilenot iszero(D) do
6      pred(D);
7  od;
8  whilenot iszero(E) do
9      pred(E);
10 od;
11 whilenot iszero(F) do
12     pred(F);
13 od;
14 whilenot iszero(G) do
15     pred(G);
16 od;
```

```
17
18 //Register D und E den Wert von Register A zuweisen
19 whilenot iszero(A) do
20     succ(D);
21     succ(E);
22     pred(A);
23 od;
24
25 //Register F und G den Wert von Register B zuweisen
26 wilenot iszero(B) do
27     succ(F);
28     succ(G);
29     pred(B);
30 od;
31
32 //Register F um D reduzieren
33 whilenot iszero(D) do
34     pred(F);
35     pred(A);
36 od;
37
38 //Wenn F den Wert 0 hat ist B entweder kleiner oder
      gleich A
39 if iszero(F) then
40     //Register E um den Wert G reduzieren
41     wilenot iszero(G) do
42         pred(E);
43         pred(G);
44     od;
45
46     //Wenn Register G den Wert 0 hat muss gelten A = B
47     if iszero(G) do
48         succ(C);
49     od;
50 fi;
```

3.4

```
1 whilenot iszero(C) do
2     pred(C);
```

```
od;
whilenot iszero(D) do
    pred(D);
od;
whilenot iszero(E) do
    pred(E);
od;
whilenot iszero(F) do
    pred(F);
od;
whilenot iszero(G) do
    pred(G);
od;

//Register D und E den Wert von Register A zuweisen
whilenot iszero(A) do
    succ(D);
    succ(E);
    pred(A);
od;

//Register F und G den Wert von Register B zuweisen
wilenot iszero(B) do
    succ(F);
    succ(G);
    pred(B);
od;

//Register F um D reduzieren
whilenot iszero(D) do
    pred(F);
    pred(A);
od;

//Wenn F den Wert 0 hat ist A entweder groesser oder
   gleich B
if iszero(F) then:
        //Den Wert im Register G (B) von Register E (A
           ) abziehen
        whilenot iszero(G) do
            pred(E);
```

```
42              pred(G);
43          od;
44          //Das Ergebnis von E - G (A - B) in C
                schreiben
45          whilenot iszero(E) do
46              succ(C);
47          od;
48      else
49          //Den Wert im Register E (B) von Register G (B
                ) abziehen
50          whilenot iszero(E) do
51              pred(G);
52              pred(E);
53          od;
54          //Das Ergebnis von G - E (B - A) in C
                schreiben
55          whilenot iszero(G) do
56              succ(C);
57          od;
58  fi;
```