Institut für Informatik **Ludwig-Maximilians-Universität München**
Prof. Dr. Volker Heun

Formale Sprachen und Komplexität, SS 18
Tutoriumsblatt 7

Aufgabe 7-1 Turingmaschine zum Berechnen einer Funktion auf Zahlen

Sei *pre* die (partielle) Vorgängerfunktion auf $\mathbb{N}$, also

$$\begin{array}{rccl} pre: & \mathbb{N} & \to & \mathbb{N} \\ & n & \mapsto & \begin{cases} n-1 & \text{falls } n > 0 \\ \textit{undefiniert} & \text{falls } n = 0 \end{cases} \end{array}$$

Sei $\Sigma = \{0, 1\}$ die Menge der Binärziffern. Die Elemente von Σ^* sollen als natürliche Zahlen in Binärdarstellung interpretiert werden. Um keine Ausnahmen behandeln zu müssen, seien führende Nullen erlaubt und ε eine zulässige Darstellung von $0 \in \mathbb{N}$.

a) Geben Sie eine deterministische Turingmaschine $dec = (Z, \Sigma, \Gamma, \delta, r, \_, \{s\})$ an, die *pre* berechnet. Das Blankzeichen ist $\_$, der Startzustand ist r, der einzige Zustand, in dem M halten kann (und der einzige Endzustand) ist s.

Lösungsvorschlag:

$Z = \{r, n, e, s\}$, $\Sigma = \{0, 1\}$, $\Gamma = \{0, 1, \_\}$

$\delta(r,\ 0) = (r,\ 0,\ R)$	nach rechts laufen
$\delta(r,\ 1) = (r,\ 1,\ R)$	bis Wortende
$\delta(r,\ \_) = (e,\ \_,\ L)$	dort weiter wie bei Übertrag 1
$\delta(n,\ 0) = (n,\ 0,\ L)$	nach links mit Übertrag 0
$\delta(n,\ 1) = (n,\ 1,\ L)$	lässt Binärziffern gleich
$\delta(n,\ \_) = (s,\ \_,\ N)$	bei Wortanfang Stop
$\delta(e,\ 0) = (e,\ 1,\ L)$	nach links mit Übertrag 1
$\delta(e,\ 1) = (n,\ 0,\ L)$	verändert Binärziffern
$\delta(e,\ \_) = (e,\ 1,\ L)$	Blanks am Wortanfang wie 0 behandeln (also keine Terminierung

b) Wie verhält sich M mit einer Binärdarstellung von $0 \in \mathbb{N}$ als Eingabewort?

Lösungsvorschlag:

M ersetzt das erste Blank vor dem Wort auf dem Band durch den Übertrag 1 und bewegt den Lese-/Schreibkopf um eine Position nach links.

Dieser Übergang wiederholt sich unendlich oft.

M hält also in diesem Fall nicht.
Das heißt, der Funktionswert ist undefiniert, wie verlangt.

Aufgabe 7-2 Turingmaschine zum Erkennen einer formalen Sprache

Sei $\Sigma = \{a, b, X\}$ und $L = \{\ wXw \mid w \in \{a, b\}^*\ \}$. Die Sprache L enthält also Wörter mit genau einem X, das man wie ein Gleichheitszeichen für Wörter über $\{a, b\}$ auffassen kann:

$$X \in L, \quad aXa \in L, \quad abXab \in L, \quad abbabXabbab \in L, \quad \ldots$$
$$aba \notin L, \quad aX \notin L, \quad aXb \notin L, \quad abXabb \notin L, \quad \ldots$$

Die Sprache L ist nicht vom Typ 2 (kontextfrei), sie kann also nicht von einem Kellerautomaten akzeptiert werden.

Geben Sie eine deterministische Turingmaschine $M = (Z, \Sigma, \Gamma, \delta, cC, \_, E)$ mit $T(M) = L$ an. Wenn die Maschine hält, soll das Band genau das Eingabewort enthalten, der Lese-/Schreibkopf auf dem ersten Zeichen dieses Eingabeworts stehen, und die Maschine im Zustand $s0$ sein (wenn sie das Wort nicht akzeptiert) oder im Zustand $s1$ (wenn sie das Wort akzeptiert).

Lösungsvorschlag:

Extrablatt/Folie

Aufgabe 7-3 LOOP-Programme

Implementieren Sie die folgenden Code-Fragmente als LOOP- und GOTO-Programme.

a) `x0 := x1 + x2`

Lösungsvorschlag:

```
x0 := x1 + 0;
LOOP x2 DO
        x0 := x0 + 1
END
```

b) `x0 := x1 * x2` (Multiplikation)

Lösungsvorschlag:

```
LOOP x1 DO
        LOOP x2 DO
                x0 := x0 + 1
        END
END
```

c) `x0 := x1^{x2}` (Potenzierung)

Lösungsvorschlag:

Vorüberlegung: ein handliches Programm wäre

```
x0 := x0 + 1; //setze x0 auf 1
LOOP x2
        x0 := x0 * x1
END
```

Aber das ist nicht LOOP-konform.
Um `x0 = x0 * x1` zu berechnen, brauchen wir eine Hilfsvariable `x3`.

```
x3 := 0;                        //eigentlich kein LOOP
                                //neue Variable benutzen
LOOP x1                         //x1 mal x3 + x0
        LOOP x0                 //x0 mal x3 + 1
                x3 := x3 + 1
        END
END;
x0 := x3 + 0
```

Insgesamt ergibt sich:

```
x0 := x0 + 1;
LOOP x2                         //Potenzierung
        x3 := x4 + 0;           //x3 := 0
        LOOP x1                 //x3 := x1 * x0
                LOOP x0
                        x3 := x3 + 1
                END
        END
        x0 := x3 + 0;
END
```