

## Übungsblatt 11

Abgabe bis Donnerstag, 26.01.12, 12:00 Uhr

### Aufgabe 11.1

In der Vorlesung wurde ein Applet vorgestellt, das eine Turing-Maschine simuliert<sup>1</sup>. Laden Sie das Applet herunter und entpacken Sie die Dateien. Mit Hilfe des Befehls

```
javac TuringMachineApplet.java
```

können Sie das Applet kompilieren und mit dem Befehl

```
appletviewer -J-Djava.security.policy=tmpolicy TuringMachineHtml.html
```

starten. Nun können Sie eine Turing-Transitionstabelle `*.tm` laden (siehe README), in der pro Zeile eine Transition angegeben ist (Sie finden ein Beispielprogramm für das Addieren von Binärzahlen in dem entpackten Ordner).

Erstellen Sie ein Turing-Programm, das entscheidet, ob die Eingabe  $\in \{1, 0\}$  ein Palindrom ist, d.h. von vorn und von hinten gelesen gleich bleibt (z.B. 1101011). Im Endzustand soll auf dem Band eine 1 stehen, wenn die Eingabe ein Palindrom war, ansonsten eine 0.

Schreiben Sie die Transitionstabelle Ihrer Turing-Maschine in die Datei `palindrom.tm` und testen Sie das Programm mit Hilfe des Applets.

---

<sup>1</sup> <http://ais.informatik.uni-freiburg.de/turing-applet/turing-file-new.tar>

## Aufgabe 11.2

Die Syntax von WHILE-Programmen mit den Variablen  $a, b$ , und  $c$  kann durch folgende Grammatik beschrieben werden:

$$G = (\{\langle \text{Anweisung} \rangle, \langle \text{Konstante} \rangle, \langle \text{Variable} \rangle\}, T, P, \langle \text{Anweisung} \rangle).$$

Die Terminalsymbole  $T$  bestehen aus

$$T = \{\text{while, do, end, }, +, -, :=, \neq, ;, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, a, b, c\}.$$

Die Produktionen  $P$  sind wie folgt gegeben:

$$P = \left\{ \begin{array}{ll} \langle \text{Anweisung} \rangle \rightarrow \langle \text{Anweisung} \rangle; \langle \text{Anweisung} \rangle, & (1) \\ \langle \text{Anweisung} \rangle \rightarrow \langle \text{Variable} \rangle := \langle \text{Konstante} \rangle, & (2) \\ \langle \text{Anweisung} \rangle \rightarrow \langle \text{Variable} \rangle := \langle \text{Variable} \rangle + \langle \text{Konstante} \rangle, & (3) \\ \langle \text{Anweisung} \rangle \rightarrow \langle \text{Variable} \rangle := \langle \text{Variable} \rangle - \langle \text{Konstante} \rangle, & (4) \\ \langle \text{Anweisung} \rangle \rightarrow \text{while } \langle \text{Variable} \rangle \neq 0 \text{ do } \langle \text{Anweisung} \rangle \text{ end}, & (5) \\ \langle \text{Variable} \rangle \rightarrow a|b|c, & (6) \\ \langle \text{Konstante} \rangle \rightarrow \langle \text{Konstante} \rangle \langle \text{Konstante} \rangle, & (7) \\ \langle \text{Konstante} \rangle \rightarrow 0|1|2|3|4|5|6|7|8|9 & (8) \end{array} \right\}$$

Überprüfen Sie, ob folgende Ausdrücke Teil der Sprache  $L(G)$  sind. Wenn ja, geben Sie die Produktionen an, die zum Erstellen der Ausdrücke notwendig sind. Die Formatierung (Zeilenumbrüche und Einrückung) dient dabei nur zur besseren Lesbarkeit und ist nicht Teil der Grammatik.

- ```
a := 11;
b := 0;
while a ≠ 0 do
  a := a - 2;
  b := b + 1
end
```
- ```
c := 1;
b := a + b;
while a ≠ 5 do
  b := b - 2;
  a := a + 1;
end
```