



## TM-Codierung

Verwenden Sie kfG Edit aus AtoCC, um die folgende kontextfreie Grammatik  $G$  für die Sprache der TM-Codierungen zu definieren.  $G = (N, T, P, s)$

$$\begin{aligned} N &= \{TMcodierung, Endzustaende, Delta, Uebergang, Kopfbewegung, \\ &\quad L, R, N, Zeichen, Null, Eins, B, Zustand, Einsenfolge\} \\ T &= \{0, 1\} \\ P &= \{TMcodierung \rightarrow Delta\ Trenner\ Endzustaende \\ &\quad Trenner \rightarrow 0\ 0 \\ &\quad Endzustaende \rightarrow Zustand\ 0\ Endzustaende \mid Zustand \\ &\quad Delta \rightarrow Uebergang\ 0\ Delta \mid Uebergang \\ &\quad Uebergang \rightarrow Zustand\ 0\ Zeichen\ 0\ Zustand\ 0\ Zeichen\ 0\ Kopfbewegung \\ &\quad Kopfbewegung \rightarrow N \mid R \mid L \\ &\quad L \rightarrow 1 \\ &\quad R \rightarrow 1\ 1 \\ &\quad N \rightarrow 1\ 1\ 1 \\ &\quad Zeichen \rightarrow B \mid Eins \mid Null \\ &\quad Null \rightarrow 1 \\ &\quad Eins \rightarrow 1\ 1 \\ &\quad B \rightarrow 1\ 1\ 1 \\ &\quad Zustand \rightarrow Einsenfolge \\ &\quad Einsenfolge \rightarrow 1 \mid 1\ Einsenfolge \\ &\quad \} \\ s &= TMcodierung \end{aligned}$$

Als Beispiel betrachten wir die Turingmaschine

$M = (\{q_0, q_1, q_2\}, \{0, 1\}, \{\$, 0, 1\}, \delta, q_0, \$, \{q_0, q_1\})$  mit

$\delta$	$\$$	0	1
$q_0$	-	-	$(q_1, 1, R)$
$q_1$	-	$(q_0, 0, L)$	$(q_2, 1, R)$
$q_2$	-	-	$(q_0, 1, L)$

Verwenden Sie AutoEdit (aus AtoCC) und simulieren Sie die Anwendung von  $M$  auf diverse Eingabewörter. Darunter sollte es Wörter  $w$  geben, für die  $M(w)$

- in einem Endzustand stoppt ( $\varepsilon, 0, 1, 0101010110$ ),
- in einem *Nicht*-Endzustand stoppt ( $11, 111, 1111$ ),
- nicht stoppt ( $10$ ).

Schauen Sie sich auch den Übergangsgraph von  $M$  an.

Kodieren Sie  $M$  aus dem Beispiel per Hand, indem Sie die folgenden Kodierungsregeln befolgen:

Symbol	Kodierung
0	1
1	11
$B$	111
$q_0$	1
$q_1$	11
$\vdots$	$\vdots$
$q_n$	$1^{n+1}$
$L$	1
$R$	11
$N$	111

Bei der Kodierung eines Zustandsübergangs (Arbeitsschritt der TM) kommt die Regel

*Uebergang  $\rightarrow$  Zustand 0 Zeichen 0 Zustand 0 Zeichen 0 Kopfbewegung*

zur Anwendung.

Dabei wählt man die Zustände in der Reihenfolge ihres Auftretens:  $\delta(q_0, 1) = (q_2, 0, R)$  wird als Wort 1011011101011 kodiert.

Analysieren Sie anschließend das von Ihnen bestimmte Wort  $w = \text{encode}(M), w \in \{0, 1\}^+$ . Da  $G$  eine kfG ist, gelingt diese Prüfung in jedem Fall. Verfälschen Sie  $w$  ein wenig und parsen Sie das so entstandene Wort.

Als nächstes soll ein Compiler entwickelt werden, der ein syntaktisch korrektes Wort aus  $L(G)$ , also eine korrekte TM-Codierung, in eine TM in der AtoCC-Repräsentation übersetzt. Beginnen Sie mit der Generierung eines LALR(1)-Parsers für  $G$ . *Wichtiger Hinweis:* Um shift-reduce-Konflikte zu vermeiden, reduzieren Sie die o.g. Grammatik  $G$ , indem Sie die Regel *Trenner*  $\rightarrow 0\ 0$  streichen und ein entsprechendes Token für den Scanner definieren.

Definieren Sie als nächstes die Zielsprache. Folgen Sie dabei dem oben angeführten Beispiel für das im Folg. die XML-basierte Darstellung von AtoCC (Auto Edit) angegeben ist.

```

<?xml version="1.0" encoding="utf-8"?>
<AUTOMATON>
  <TYPE value="TM"/>
  <ALPHABET>
    <ITEM value="0"/>
    <ITEM value="1"/>
  </ALPHABET>
  <TAPEALPHABET>
    <ITEM value="$"/>
    <ITEM value="0"/>
    <ITEM value="1"/>
  </TAPEALPHABET>
  <STATE name="q_0" finalstate="true">
    <TRANSITION target="q_1">
      <LABEL read="1" direction="RIGHT" write="1"/>
    </TRANSITION>
  </STATE>
  <STATE name="q_1" finalstate="true">
    <TRANSITION target="q_0">
      <LABEL read="0" direction="LEFT" write="0"/>
    </TRANSITION>
    <TRANSITION target="q_2">
      <LABEL read="1" direction="RIGHT" write="1"/>
    </TRANSITION>
  </STATE>
  <STATE name="q_2" finalstate="false">
    <TRANSITION target="q_0">
      <LABEL read="1" direction="LEFT" write="1"/>
    </TRANSITION>
  </STATE>
  <INITIALSTATE value="q_0"/>
  <TAPEINITIALCHAR value="$"/>
  <LAYOUT>
    <STATELAYOUT name="q_0">
      <LEFT value="100"/>
      <TOP value="114"/>
      <TRANSITIONLAYOUT target="q_1" drawvertical="true">
        <TOP value="56"/>
      </TRANSITIONLAYOUT>
    </STATELAYOUT>
    <STATELAYOUT name="q_1">
      <LEFT value="277"/>
      <TOP value="114"/>
      <TRANSITIONLAYOUT target="q_0" drawvertical="true">
        <TOP value="-71"/>
      </TRANSITIONLAYOUT>
      <TRANSITIONLAYOUT target="q_2" drawvertical="true"/>
    </STATELAYOUT>
    <STATELAYOUT name="q_2">
      <LEFT value="425"/>
      <TOP value="114"/>
      <TRANSITIONLAYOUT target="q_0" drawvertical="true">

```

```

        <LEFT value="20"/>
        <TOP value="154"/>
    </TRANSITIONLAYOUT>
</STATELAYOUT>
<SIMULATIONINPUT>
    <INPUT value="0010111001"/>
    <INPUT value="0011101111"/>
    <INPUT value="1000010111"/>
    <INPUT value="1100000001"/>
    <INPUT value="0001100111"/>
    <INPUT value="1100111101"/>
    <INPUT value="0001100111"/>
    <INPUT value="0011101111"/>
    <INPUT value="0001100111"/>
    <INPUT value="0001100111"/>
    <INPUT value="0001100111"/>
</SIMULATIONINPUT>
<NOTEPAD>
    {\rtf1\ansi\ansicpg1252\deff0\deflang1031{\fonttbl{\f0\fnil\fcharset0 Arial;}}
\viewkind4\uc1\pard\fs18
\par }

</NOTEPAD>
</LAYOUT>
</AUTOMATON>

```

Verwenden Sie VCC, um einen entsprechenden Compiler zu beschreiben und zu erzeugen. Wenden Sie diesen Compiler auf die TM-Kodierung von  $M$  an und überprüfen Sie, ob die ursprüngliche Maschine  $M$  (in Auto Edit) entsteht.