Ti*k*z für Automaten

Ein Paper über Ti*k*z für Informaiker zum zeichnen von Automaten

Hauke Stieler
Universität Hamburg [1]

22. April 2015
ver.: 0.2 `de_DE`

Zusammenfassung

Hier wird es eine kurze Erklärung zum zeichnen von Automaten mit Hilfe von Ti*k*z geben. Da ich selbst kein Profi im gebiet bin, bitte ich Ideen, Fehler, Anmerkungen und Anregungen an meine E–Mail–Adresse mail@hauke-stieler.de zu senden.

Allgemein wird dies auch mehr eine Sammlung an Beispielen[2] zu verschidenen Automaten(modellen), die im Laufe der FGI I Vorlesung im SoSe15 fortgeführt wird.

[1] Für weiteren Kontakt kerne auch über die Kennung `4stieler`.

[2] Für eine weitaus ausführlichere Darbietung von Ti*k*z gibt es diese 1165-Seiten-PDF als genüssliche Abendlektüre: Ti*k*z manual

Inhaltsverzeichnis

1 Seinen Editor vorbereiten

Natürlich braucht man zum benutzen von Ti*k*z das gleichnamige Packet, welches man einfach mit \usepackage einfach benutzen kann. Neben diesem müssen wir aber noch weitere, spezielle Ti*k*z -Bibliotheken importieren.

```
1  \usepackage{tikz}
2  \usetikzlibrary{arrows,automata,positioning}
```

Code 1: Alle Pakete, die wir benutzen (möchten).

Es gibt noch viele weitere Bibliotheken von Ti*k*z , doch die hier genannten brauchen wir (und sind für Automaten die wichtigsten).

2 Die Ti*k*z Umgebung und ihre Parameter

2.1 Eine Umgebung erzeugen

Hier nun ein kleines Beispiel zum zeichnen von Graphen (gezeichnete Automaten sind ja auch Zustands*graphen* ;)).

Ti*k*z benutzt zum zeichnen von Graphen sogenannte nodes und paths. Nodes sind dabei die Zustände/Boxen/Kreise eines Graphen, der path beschreibt eine oder mehrere Linien dazwischen.

Um in Ti*k*z los zeichnen zu können, müssen wir eine Umbebung beginnen/erzeugen:

```
1  \begin{tikzpicture}
2  ...
3  \end{tikzpicture}
```

Code 2: Eine einfache Ti*k*z Umgebung erstellen.

2.2 Parameter einer Umgebung

Für diese Umgebung können wir uns verschiedene Parameter heraussuchen, die wir nutzen möchten. Hier ein Beispiel:

```
\begin{tikzpicture}[->,
    >=stealth',
    shorten >= 5pt,
    node distance = 2.8cm,
    semithick]
...
\end{tikzpicture}
```

Code 3: Ein Beispiel mit Parametern in einer Ti*k*z Umgebung.

Hier eine kurze Erklärung der Parameter:

- `->` : Alle Linien zwischen nodes sind nun Pfeile
- `>=stealth'` : Die Pfeilspitzen sind ausgefüllt und dreieckig
- `shorten >= 5pt` : Die Pfeilspitzen hören `5pt` vor der node auf
- `node distance=2.8cm` : Alle nodes sind `2.8cm` voneinander entfernt
- `semithick` : Gibt die Dicke eines paths an

Zu den einzelnen Parametern jedoch jetzt mehr.

2.3 Grundlegende und verschiedene Parameter

Hier nun ein paar grundlegende Parameter.

- `-` : Alle verbindungen sind normale Linien
- `->` : Alle verbindungen sind Pfeile
- `>= <Option>` : Legt die Art der Pfeilspitze fest
- `shorten >= x` : Entfernung Pfeilspitze – Node in x (z.B. 5pt, 1cm, ...)

2.4 Farbe des Graphen

Über die Parameter kann man auch gleich die Farbe des gesamten Bildes angeben. Dazu einfach `color=<color>` als Option wählen.

Dabei kann man Standardfarben, wie `red`, `green`, `blue`, ... wählen und diese in ihrer Helligkeit per `!<prozent>` beeinflussen.

```
1  \begin{tikzpicture}[->,
2      thin,
3      color=red!60]
4  ...
5  \end{tikzpicture}
```

Code 4: Wählen einer hell roten Farbe (60% rot, 40% weiß/hintergrund)

3 Pfeile und Linien

3.1 Dicke der Pfeile/Linien

Es gibt verschiedene Dicken für Pfeile und Linien (oben: super dünn, unten: super dick):

- `\ultra thin`
- `\very thin`
- `\thin`
- `\semithick`
- `\thick`
- `\very thick`
- `\ultra thick`

3.2 Pfeilspitzen

Um Pfeile richtig nutzen zu können braucht man die Ti*k*z Bibliothek `arrows`:

```
1  \usetikzlibrary{arrows}
```

Code 5: Die Bibliothek `arrows` wird zum Pfeile zeichnen benötigt.

Es gibt viele verschiedene - teils seltsame - Pfeilarten. Die Pfeilart kann ebenfalls als Option am Anfang angegeben werden.

```
\begin{tikzpicture}[->,
    >=<Pfeilart>]
...
\end{tikzpicture}
```

Code 6: Angabe einer Pfeilart.

Hier eine kleine - ebenfalls für Automaten interessante - Auswahl:

- keine Angabe : ⟶
- `latex` : ⟶
- `latex'` : ⟶
- `stealth` : ⟶
- `stealth'` : ⟶
- `triangle <Winkel>` und `open triangle <Winkel>` :
  Mit 45°: ⟶ ⟶
  Mit 60°: ⟶ ⟶
  Mit 90°: ⟶ ⟶
- `angle <Winkel>`
  Mit 45°: ⟶
  Mit 60°: ⟶
  Mit 90°: ⟶

Die Option `'` bewirkt bei `latex'` das hyperbolische zuspitzen des Dreiecks. Bei `stealth'` werden die Ecken abgerundet.

Außerdem kann man mit der Option `->>` einen doppelten Pfeil erzeugen (z.B. mit der `stealth'` Option: ⟶). Dies geht bei allen Arten von Pfeilen.

4 Nodes

Nodes sind die Zustände in einem Automaten. Dank der Bibliothek `automata` gibt es schon verschiedene vordefinierte Eigenschaften.

Um mit Automaten und verschiedenen Optionen zu Automaten arbeiten zu können muss die Bibliothek `automata` eingebunden werden.
Damit man die erzeugten Nodes auch einfach und schnell positionieren kann, sollte man zudem die Bibliothek `positioning` benutzen:

```
\usetikzlibrary{automata}
\usetikzlibrary{positioning}
```

Code 7: Nutzen der Bibliothek `automata` und `positioning`

4.1 Erstellen von Nodes

Nun können wir die ersten Nodes mit \node erstellen. Wobei folgende Syntax beachtet werden sollte:

```
\node[<Optionen>] (name) [<Optional: Position>] {text
    };
```

Code 8: Syntax des \node Befehls.

Wird keine Position angegeben befindet sich der Zustand am Nullpunkt (normalerweise ganz links). Die Parameter der Positioniertung werden später noch besprochen.

Wichtig zu bemerken ist auch, dass der Text auch im Mathe-Modus angegeben werden kann um so z.B. Indizes anzugeben.

4.1.1 Beispielautomat ohne Übergänge

So kann nun ein Automat aussehen, wenn man drei Zustände z_0, z_1, und z_2 erzeugt.

```
\begin{tikzpicture}[->,
    >=stealth',
    semithick]

    \node[state,initial]    (0)              {$z_0$};
    \node[state]            (1) [right of=0] {$z_1$};
    \node[state,accepting]  (2) [right of=1] {$z_2$};
\end{tikzpicture}
```

Code 9: Zeichnen eines einfachen Automaten.

Hier also das Ergebnis:

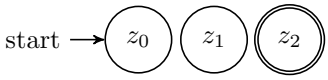

Abbildung 1: Ein simpler Automat mit 3 Zuständen.

Auch hier können dem Befehl \node verschiedene Parameter aus automata übergeben werden.

4.2 Optionen für Zustände

Hier eine Liste an nützlichen Optionen für Zustände (\nodes):

- `state` : Muss angegeben werden, damit es sich um einen Zustand handelt
- `initial` : Gibt an, dass dies der/ein Startzustand ist
  Erzeugt ein automatisches *start*-Label mit Pfeil
- `accepting` : Gibt an, dass dies der/ein Endzustand ist
  Erzeugt einen doppelten Kreis um den Zustand
- `with output` : Erzeugt einen Zustand mit Ausgabe
  Dazu jedoch mehr, wenn gewünscht

4.3 Positionierung der Zustände

Zustände können durch die Bibltiothek `positioning` ganz einfach angegeben werden.

4.3.1 Optionen für die Positionierung

Für die Positionierung von Nodes gibt es unter anderem folgende Optionen, die - wie weiter unten beschrieben - auch kombiniert werden können:

- `right, left, above, below` : Rechts, links, über oder unter einer Node
- `of = <Node>` : Gibt von welcher Node man bei `right, left, ...` spricht
- `and` : Wenn man mehrere Entfernungen angeben möchte, nutzt man `and`
  Bsp.: `[...=2cm and 1cm of A]`

4.3.2 Automatische Entfernung

Dabei ist für **automatische Entfernungen** die einfache Syntax zu beachten:

```
1   [<Position> of = <Referenz-Node>]
```

Code 10: Syntax für die Positionierung von Nodes.

Den Wert der automatischen Entfernung wird mit `node distance=...` in `cm, em, pt, ...` angegeben. Also z.B.:

```
1   [right of = A]
```

Code 11: Automatische Positionierung einer Node rechts von A.

Dies positioniert die eine Node rechts neben der Node A, wobei die durch `node distance=...` gewählte Entfernung genommen wird.

4.3.3 Manuelle Entfernung

Für **manuelle Entfernungen**, die man wie gewohnt per `cm, em, pt, ...` angeben kann (wie in LaTeX üblich), benutzt man folgende Syntax:

```
1  [<Positionen> = <Entfernungen > of <Referenz-Node>]
```

Code 12: Syntax für die Positionierung von Nodes.

Also z.B.:

```
1  [right = 2cm of A]
```

Code 13: Manuelle Positionierung einer Node rechts von A.

Dies positioniert die eine Node 2cm rechts neben der Node A.

4.3.4 Beispiel für das Positionieren von Nodes

Man hat drei Nodes: a, b und c.

- Node b soll rechts über (2cm rechts, 1cm über) der Node a sein
- Node c soll automatisch unten rechts von a angeordnert werden

Hierzu kann man die Optionen geschickt kombinieren:

```
1  \begin{tikzpicture}[->,
2      >=stealth',
3      semithick,
4      node distance=2cm]
5
6  \node [state] (a)                                  {a};
7  \node [state] (b) [above right=1cm and 2cm of a] {b};
8  \node [state] (c) [below right of = a]           {c};
9
10 \end{tikzpicture}
```

Code 14: Kombination von Optionen zur Positionierung.

Wenn man als Abstand einzelner Nodes noch einen netten Wert (z.B. `node distance=2cm`) wählt, dann sirht das so aus:

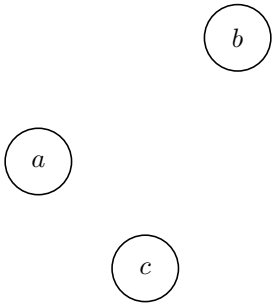

Abbildung 2: Drei unterschiedlich angeordnete Nodes.

5 Nodes verbinden

Damit man nicht nur einen Haufen Zustände hat, sondern einen Automaten, muss man diese logischerweise verbinden. Dazu kann man den `\path` Befehl nehmen.

5.1 Pfeile erstellen

Auch `\path` hat eine bestimmte Syntax, die man beachten muss:

```
1   \path (<von-Node>) edge [<Options>] node {text} (<
        nach-Node>);
```

Code 15: Syntax des `\path` Befehls.

Wichtig zu bemerken ist, dass man nur einmal `\path` schreiben braucht (s. Beispiel unten).
Was es alles für Optionen gibt, kommt gleich, zunächst einmal das Beispiel:

```
1  \begin{tikzpicture}[->,
2      >=stealth',
3      semithick,
4      node distance=2cm]
5
6  \node [state,initial] (a)                          {a};
7  \node [state] (b) [above right=1cm and 2cm of a] {b};
8  \node [state,accepting] (c) [below right of = a] {c};
9
10 \path (a) edge node {0} (b)
11           edge node {1} (c)
12       (c) edge node {2} (b);
13
14 \end{tikzpicture}
```

Code 16: Der bekannte (leicht geänderte) Automat von vorhin. Diesmal mit Verbindungen untereinander, aber **ohne** Optionen.

Dazu der passende Ti*k*z -Automat:

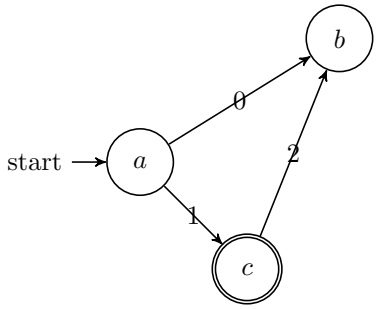

Abbildung 3: Drei unterschiedlich angeordnete Nodes mit Verbindungen **ohne** Optionen.

5.2 Optionen für den `\path` Befehl

Auch hier kann man verschiedene Optionen wählen:

- `right, left, above, below` : Optionen für die Beschriftung, legt fest, wo der Text dargestellt werden soll
- `bend <right, left>` : Erzeugt eine gebogene Kante. Die Richtung der Biegung (nach rechts oder links) ist von Pfeilrichtung aus gesehen
- `bend <right, left> = <Winkel>` : der `Winkel` beschreibt die Stärke der Biegung und wird in Grad angegeben

- `loop <right, left, above, below>` : Erzeugt eine Schleife rechts, links, über oder unter einer Node

6 Beispiele

Da nun die wichtigsten Dinge behandelt wurden kommen hier ein paar Beispiele:

6.1 Beispiel 1

```
1  \begin{tikzpicture}[->,>=stealth',
2      shorten >=5pt,
3      node distance=2.5cm,
4      semithick]
5
6  \node[initial,state]   (R)               {$z_r$};
7  \node[state]           (S) [right of=R] {$z_s$};
8  \node[state]           (T) [right of=S] {$z_t$};
9  \node[state,accepting] (E) [right of=T] {$z_e$};
10
11 \path    (R) edge [loop,above]       node {0} (R)
12             edge [below]             node {1} (S)
13         (S) edge [below]             node {0} (T)
14             edge [loop,above]        node {1} (S)
15         (T) edge [bend left,below]   node {0} (R)
16             edge [below]             node {1} (E)
17         (E) edge [loop,above]        node {0,1} (E)
18             ;
19 \end{tikzpicture}
```

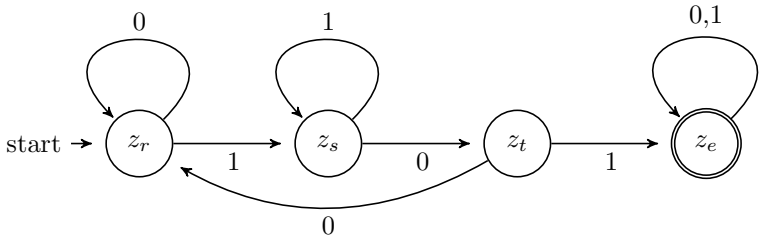

Abbildung 4: Ein Automat, der die Sprache $L = \{0,1\}^* \cdot \{101\} \cdot \{0,1\}^*$ akzeptiert.

6.2 Beispiel 2

```
\begin{tikzpicture}[->,
    >=stealth',
    node distance=2.8cm,
    semithick]

\node[initial,state]   (0)
                {$z_0$};
\node[state]           (1) [above right=0.65cm and
    1.5cm of 0]  {$z_1$};
\node[state]           (2) [below right=0.65cm and
    1.5cm of 0]  {$z_2$};
\node[state,accepting] (E) [right=3.75cm of 0]
                {$z_e$};

\path    (0) edge [above]                node {1} (1)
             edge [below]                node {0} (2)
         (1) edge [left, bend right=10]  node {0} (2)
             edge [above]                node {1} (E)
         (2) edge [right,bend right=10]  node {1} (1)
             edge [below,loop below]     node {0} (2)
         (E) edge [below]                node {0} (2)
             edge [right,loop right]     node {1} (E)
             ;
\end{tikzpicture}
```

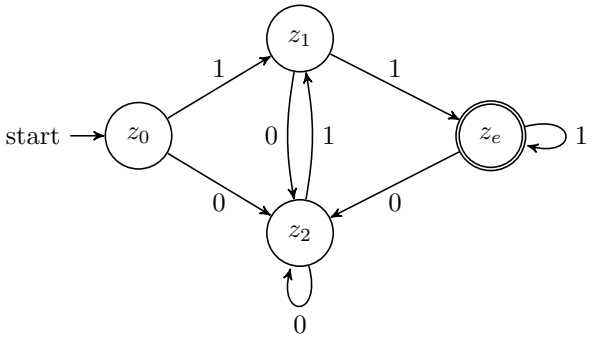

Abbildung 5: Ein Automat, der die Sprache $L = \{0,1\}^* \cdot \{0,11\}$ akzeptiert.