Ti*k*z for state-machines

A paper about Ti*k*z for computer scientists to draw state-machines.

Hauke Stieler
Universität Hamburg [1]

April 22, 2015
ver.: 0.1 `en_EN`

Abstract

You'll find a short explanation on how to draw state-machines with Ti*k*z . Because I'm not a professional, please send ideas, mistakes, notes and suggestions to my e–mail–adress mail@hauke-stieler.de.

Generally this is a collection[2] of examples for different state-machines and -models, which are dealt with in the summer-semester 2015 in the FGI I lecture.

[1] For further contact also via my ID `4stieler`.

[2] For further information and as a book for cozy eventing, you'll find a lot of information in this 1165-page-manual: Ti*k*z manual

Contents

1 Prepare you editor

To use Ti*k*z , you need of course the same packet, which can be easily imported with \usepackage. Next to this we have to import several Ti*k*z -libraries.

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

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

There are many other libraries in Ti*k*z , but we need just these (which are important for state-machines).

2 The Ti*k*z environment and its parameter

2.1 Create an environment

Here's just an simple example on how to draw an graph (drawed state-machines are nothing else than state-*graphs* ;)).

To draw graphs, Ti*k*z uses **node**s and **path**s. Nodes are the states/boxes/-circles of a graph, while the path describes one or more line between them.

To getting stared with drawing in Ti*k*z , we must create an environment:

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

Code 2: Create a simple Ti*k*z -enviroment.

2.2 Parameter of an environment

For this environment, there are different parameter, we can choose from. For instance:

```
1  \begin{tikzpicture}[->,
2      >=stealth',
3      shorten >= 5pt,
4      node distance = 2.8cm,
5      semithick]
6  ...
7  \end{tikzpicture}
```

Code 3: A example with parameter in a Ti*k*z -environment.

Here's a short explanation of these parameter:

- `->` : All lines between nodes are arrows
- `>=stealth'` : The arrow heads are filled and triangles
- `shorten >= 5pt` : The arrow head stops `5pt` before the node
- `node distance=2.8cm` : All nodes are `2.8cm` away from each other
- `semithick` : Specifies the thickness of the line

More detailed information on the different parameter now.

2.3 Basic and miscellaneous parameter

Here just a few basic parameter.

- `-` : All connections are normal lines
- `->` : All connections are arrows
- `>= <Option>` : Specifies the type of the arrow head
- `shorten >= x` : Distance arrow head – Node (e.g. 5pt, 1cm, ...)

2.4 Color of graphs

You can change the color of the graph with the paremeter. To do that, just use the `color=<color>` option.

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

Code 4: Choosing a slight red color (60% red, 40% white/background)

3 Arrows and lines

3.1 Thickness of lines and arrows

There're different thickness options available for arrows and lines (top: very thin, bottom: thickest):

- `\ultra thin`

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

3.2 Arrow heads

To use arrows properly, add the Ti*k*z library `arrows` to your collection:

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

Code 5: The library `arrows` is used to draw arrows.

There're many different - party strange - types of arrows. The type of an arrow can also be specified at the beginning of an option.

```
1  \begin{tikzpicture}[->,
2      >=<arrow-type>]
3  ...
4  \end{tikzpicture}
```

Code 6: Specification of an arrow type.

Here's a small - for state-machines interesting - selection of different arrow types:

- no value : ⟶
- `latex` : ⟶
- `latex'` : ⟶
- `stealth` : ⟶
- `stealth'` : ⟶
- `triangle <angle>` und `open triangle <angle>` :
  With 45°: ⟶ ⟶
  With 60°: ⟶ ⟶
  With 90°: ⟶ ⟶
- `angle <angle>`
  With 45°: ⟶
  With 60°: ⟶
  With 90°: ⟶

The option ' with the latex' creates a hyperbolic sharpened triangle as the arrow head. stealth' makes the edges more round than sharp.

Furthermore can the option ->> be used to create a double arrow with two heads (e.g. with the stealth' option: $\twoheadrightarrow$). This is possible with all kinds of arrows.

4 Nodes

Nodes are used to create states for our state-machine(s). Thanks to the library automata there are lots of different predefined options. To use these options, just integrate the library automata

To position the created node very quickly and easily, you should use the library positioning:

```
1  \usetikzlibrary{automata}
2  \usetikzlibrary{positioning}
```

Code 7: Using the library automata and positioning

4.1 Create nodes

Now we can create the first nodes with \node. Wherein it should be noted that there's a specific syntax:

```
1  \node[<optionen>] (name) [<optional: position>] {text
       };
```

Code 8: Syntax of the \node command.

When no position is specified, the node will be placed at the origin (normally at the very left side). The parameter for the positioning will be discussed later.

Important to be mentioned is the opportunity to write the displayed text of a node in math-mode, which allows indices like z_0 or something.

4.1.1 Example machine without connections

A state-machine with the three states z_0, z_1 and z_2 can be look like this:

```
1  \begin{tikzpicture}[->,
2      >=stealth',
3      semithick]
4
5      \node[state,initial]     (0)               {$z_0$};
6      \node[state]             (1) [right of=0] {$z_1$};
7      \node[state,accepting]   (2) [right of=1] {$z_2$};
8  \end{tikzpicture}
```

Code 9: Drawing of a very simple machine.

Ans here's the result:

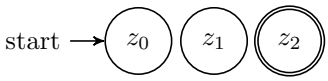

Figure 1: A simple machine with three states.

You can passing over parameters to the \node command from the automata library as well.

4.2 Options for states

Here's a list of some useful options for states (\nodes):

- `state` : Has to be specified to create a state
- `initial` : Specifies a start-state
  Creates an arrow with *start*-Label
- `accepting` : Indicates the final state of the machine
  Draws a double circle around the state
- `with output` : Creates a state with output
  More information if desired

4.3 Positioning of the states

The position of the states can easily be specified by using the `positioning` library.

4.3.1 Options for positioning

For the positioning are - amongst others - the following options available. These options can also be combined:

- `right, left, above, below` : Next to the node (in given direction)
- `of = <Node>` : Specifies the node use mean by using `right, left, ...`
- `and` : When you want to use several distances, use `and`
  Ex.: `[...=2cm and 1cm of A]`

4.3.2 Automatic distance

It should be considered that there's a specific syntax for the **automatic distance** of nodes.

```
1   [<position> of = <reference-node>]
```

Code 10: Syntax for the positioning of nodes.

The value of the automatic distance can be defined by using `node distance=...` with the normal LaTeXunits (`cm, em, pt, ...`). So for example:

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

Code 11: Automatic positioning of a node to the right of A.

This placed a node to the right of a node A, while the distance defined by `node distance=...` is used.

4.3.3 Manual distance

To use a **manual distance**, which can be specified by using the well known LaTeXunits `cm, em, pt, ...`, you have to consider the syntax:

```
1   [<position> = <distance> of <reference-node>]
```

Code 12: Syntax for the manual positioning of nodes.

For example:

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

Code 13: Manual positioning of a node right of A

This placed the one node 2cm to the right of node A.

4.3.4 Example for the positioning of nodes

Given are three nodes: a, b and c.

- Node b should be placed right and above (2cm right, 1cm above) the Node a
- Node c should be placed automatically right and below of node a

Therefore you can skillfully combine the options:

```
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: Combination of options for the positioning.

If you choose a nice value for the distance (e.g. `node distance=2cm`), it can look pretty fancy:

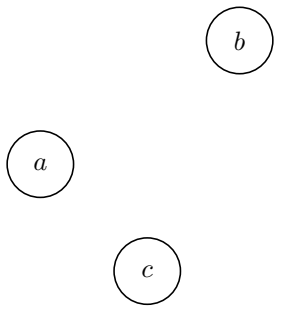

Figure 2: Three different placed nodes.

5 Nodes verbinden

To not only have a bunch of states but a real state-machine, you have to connect these states. There's a command called \path to create those connections.

5.1 Create arrows

\path has a syntax as well:

```
\path (<from-node>) edge [<options>] node {text} (<to
    -node>);
```

Code 15: Syntax of the \path command.

Important to be noted is that you need to write \path just once (see example below).
What kind of options exist, is coming soon, but at first we'll see an example of this:

```
\begin{tikzpicture}[->,
    >=stealth',
    semithick,
    node distance=2cm]

\node [state,initial] (a)                          {$a$};
\node [state] (b) [above right=1cm and 2cm of a] {$b$};
\node [state,accepting] (c) [below right of = a] {$c$};

\path (a) edge node {0} (b)
          edge node {1} (c)
      (c) edge node {2} (b);

\end{tikzpicture}
```

Code 16: The well known machine from before, but this time with some connections.

And here's the belonging Ti*k*z -machine

5.2 Options for the \path command

You can choose on different options as well:

- `right, left, above, below` : Specifies the position of the text
- `bend <right, left>` : Creates a bended edge. The direction of the bending is specified from the perspective of the arrow

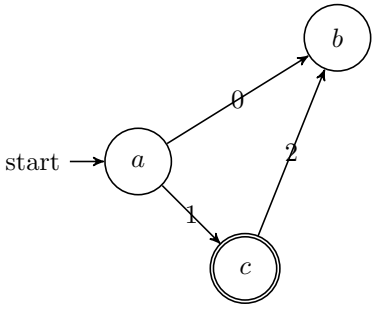

Figure 3: Three states with come connections (**without** options.

- `bend <right, left> = <angle>` : The `angle` describes the strength of the bending
- `loop <right, left, above, below>` : Creates a loop above, below, left or right of the state

6 Examples

Because the most important aspects as been discussed, you'll find a few examples down here.

6.1 Example 1

```
\begin{tikzpicture}[->,>=stealth',
    shorten >=5pt,
    node distance=2.5cm,
    semithick]

\node[initial,state]   (R)              {$z_r$};
\node[state]           (S) [right of=R] {$z_s$};
\node[state]           (T) [right of=S] {$z_t$};
\node[state,accepting] (E) [right of=T] {$z_e$};

\path   (R) edge [loop,above]       node {0} (R)
            edge [below]            node {1} (S)
        (S) edge [below]            node {0} (T)
            edge [loop,above]       node {1} (S)
        (T) edge [bend left,below]  node {0} (R)
            edge [below]            node {1} (E)
        (E) edge [loop,above]       node {0,1} (E)
            ;
\end{tikzpicture}
```

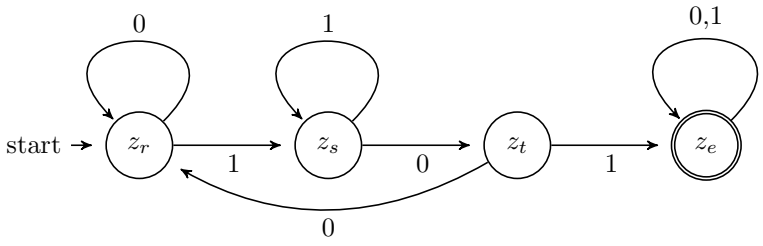

Figure 4: A machine, accepting the language L with $L = \{0,1\}^* \cdot \{101\} \cdot \{0,1\}^*$.

6.2 Example 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}
```

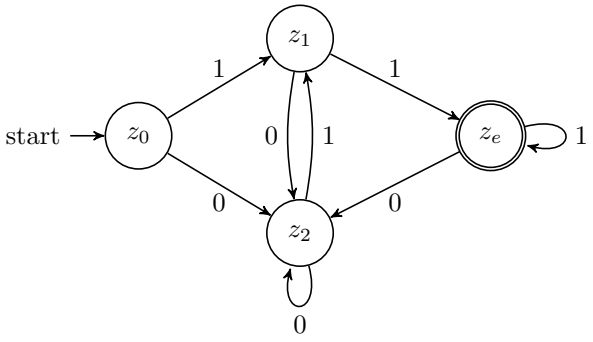

Figure 5: A machine accepting the language L with $L = \{0,1\}^* \cdot \{0,11\}$.