Drawing Finite State Machines in LaTeX using `tikz` A Tutorial

Satyaki Sikdar
ssikdar@nd.edu

August 31, 2017

1 Introduction

Paraphrasing from [beg14], LaTeX (pronounced lay-tek) is an open-source, multiplatform document preparation system for producing professional-looking documents, it is not a word processor. It is particularly suited to producing long, structured documents, and is very good at typesetting equations.

The capabilities of the system are greatly enhanced with the help of native and third-party packages[1]. The `tikz` [2] library is one such package which primarily focuses on data visualization.

This tutorial will aim to introduce the reader to the `tikz` library, particularly for drawing state diagrams of *DFAs* and *NFAs*.

2 Setting up LaTeX

To proceed with the tutorial, a working LaTeX setup is necessary. You may choose to install it locally on your machine, or use an online service like ShareLaTeX[3] or Overleaf[4]. For further information regarding setup, visit `http://www.latex-project.org/get/`.

While there exists a myriad of tutorials online, [beg14] and `https://www.latex-tutorial.com/` are suggested to make you familiar with LaTeX.

3 The `tikz-automata` library

The `tikz` library is not imported by default. So, it must be imported manually in the preamble of the document (after `documentclass`, but before the `begin document` statements). The `automata` library is needed to draw the state diagrams of FSMs. The `positioning` and `arrows` libraries are imported to help position the nodes and customize the arrows respectively.

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

Each object in a `tikz` picture is called a *node*, and each node can be customized as per requirement. Edges or paths can span between nodes.

Inside the document, each `tikz` diagram must reside in the `tikzpicture` environment. It's advisable to put the `tikzpicture` environment inside the `figure` environment, as shown below.

[1] The Comprehensive TeX Archive Network (CTAN) is the central place for all kinds of material around TeX. `https://www.ctan.org/?lang=en`

[2] `https://www.ctan.org/pkg/pgf?lang=en`

[3] `https://www.sharelatex.com/`

[4] `https://www.overleaf.com/`

```
\begin{figure}[ht] % 'ht' tells LaTeX to place the figure 'here' or at the top of the page
    \centering % centers the figure
    \begin{tikzpicture}
        % tikz code goes here
    \end{tikzpicture}
    \caption{Caption of the FSM}
    \label{fig:my_label}
\end{figure}
```

The attributes of each `tikzpicture` can be declared either globally, or in the environment declaration for each individual `tikzpicture`.

Setting the attributes of the pictures globally allows for all the figures to be consistent. To make the FSMs consistent to those in the [Sip12], use the following code snippet in the document preamble *after* importing the `tikz` library.

```
\tikzset{
        ->,  % makes the edges directed
        >=stealth', % makes the arrow heads bold
        node distance=3cm, % specifies the minimum distance between two nodes. Change if necessary.
        every state/.style={thick, fill=gray!10}, % sets the properties for each 'state' node
        initial text=$ $, % sets the text that appears on the start arrow
        }
```

3.1 Nodes

Let's start off by drawing nodes. Nodes can be positioned either manually or relative to other nodes. Relative placement is often much easier. Note that the following two subsections is a slightly modified version of the corresponding subsections in section 4 of [Sti15].

```
\node[<options>] (name) {text label};
```

3.1.1 Options

The options (in the context of FSMs) are:

- `state`: **Must** be specified to create a state
- `initial`: Specifies a start-state
- `accepting`: Indicates the final state of the machine. Draws a double circle around the state.

See fig 1 to see how the different nodes look. Note that the size of a node depends on the length of its text label.

3.1.2 Positioning

The position of the states can easily be specified by using the `positioning` library which we have already imported.

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

- `of = <node>`: Specifies the node use mean by using `right, left, ....` Eg:

```
\node[state, right of=q1] (q2) {$q_2$};
```

- `xshift=`x`, yshift=`y: Gives manual control of the node positions after relative placement. Eg:

```
\node[state, right of=q1, xshift=1cm] (q2) {$q_2$};
```

- `at (x, y)`: Forces `tikz` to place the node at (x, y) in the figure. The origin $(0, 0)$ is at the center of the figure. Eg:

```
\node[state] (q) at (2, 3) {$q$};
```

Here's the `tikz` code for figure 1.

```
\begin{tikzpicture}
    \node[state] (q1) {normal};
    \node[state, initial, right of=q1] (q2) {start};
    \node[state, accepting] at (1.5, 2) (q3) {accept};
\end{tikzpicture}
```

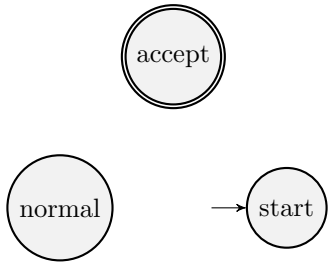

Figure 1: Different node types

3.2 Edges

Once the states are all in place, let's start adding the transitions, i.e. the edges between the states.

The `draw` command can be used to draw the edges between the already created nodes (states). The syntax is as follows

```
\draw (<source node>) edge[<edge options>] node{<edge label>} (<dest node>);
```

Note:

- `<source node>` and `<dest node>` are the names of the nodes and NOT the labels.
- `<edge properties>` modify the apperance of the edge. For edges that start and end in the same node, i.e. the loops, the `loop <direction>` property must be used. The direction determines the direction of the loop. It can be `above, below, left` and `right`.
- By default, the edges are straight, so to prevent overlaps, they can be *bent* either `left` or `right`.
- The edge labels can be positioned either above or below the edge, with the keywords `above` and `below` in the edge options.

Multiple edges can be drawn with the same `draw` command, in which case put each edge in it's own line, and on the final line end with the `;`.

```
\draw   (<source node1>) edge[<edge options1>] node{<edge label1>} (<dest node1>)
        (<source node2>) edge[<edge options2>] node{<edge label2>} (<dest node2>)
        ...
        ...
        ...
        (<source node k>) edge[<edge options k>] node{<edge label k>} (<dest nodek>);
```

It's advisable to draw the edges in order of the source nodes.
Now, with everything in place, let's start drawing some Finite State Machines, starting with DFAs.

3.3 Drawing DFAs

Let's start off with a simple DFA from [Sip12]. The DFA is described as follows

$$D_1 = (\{q_1, q_2, q_3\},\ \{0,1\},\ \delta,\ q_1,\ \{q_2\}),$$

where δ is given by:

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

Below is the code that generates the state diagram of the D_1. The output is shown in figure 2.

```
\begin{tikzpicture}
    \node[state, initial] (q1) {$q_1$};
    \node[state, accepting, right of=q1] (q2) {$q_2$};
    \node[state, right of=q2] (q3) {$q_3$};

    \draw   (q1) edge[loop above] node{0} (q1)
            (q1) edge[above] node{1} (q2)
            (q2) edge[loop above] node{1} (q2)
            (q2) edge[bend left, above] node{0} (q3)
            (q3) edge[bend left, below] node{0, 1} (q2);
\end{tikzpicture}
```

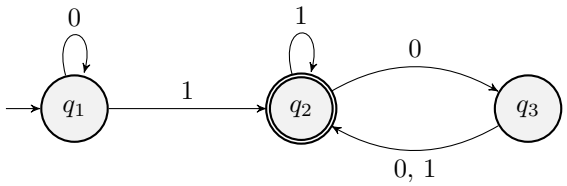

Figure 2: DFA example ([Sip12], page 36, fig 1.6)

3.4 Drawing NFAs

Drawing NFAs is very similar to drawing to DFAs. Let's see an example from [Sip12]. The NFA is described as follows

$$N = (\{0,1,2\},\ \{a,b\},\ \delta,\ 1,\ \{1\}),$$

where δ is given by:

	a	b	ϵ
1	{}	{2}	{3}
2	{2,3}	{3}	{}
3	{1}	{}	{}

Below is the code that generates the state diagram of the N. The output is shown in figure 3.

```
\begin{tikzpicture}
    \node[state, initial, accepting] (1) {$1$};
    \node[state, below left of=1] (2) {$2$};
    \node[state, right of=2] (3) {$3$};

    \draw   (1) edge[above] node{$b$} (2)
            (1) edge[below, bend right, left=0.3] node{$\epsilon$} (3)
            (2) edge[loop left] node{$a$} (2)
            (2) edge[below] node{$a, b$} (3)
            (3) edge[above, bend right, right=0.3] node{$a$} (1);
\end{tikzpicture}
```

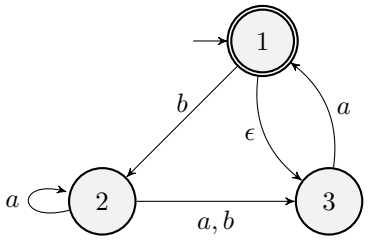

Figure 3: NFA example ([Sip12], page 57, fig 1.42)

3.5 Exponential Blow ups

Every NFA can be converted to an equivalent DFA. Prior to minimization, the DFAs have exponentially many states compared to the corresponding NFA. Drawing those FSMs can get messy. So, here's the final example showing the state diagram of the DFA equivalent to the NFA N (prior to minimization). The DFA D_2 can be described as:

$$D_2 = (\{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\},\ \{a, b\},\ \delta, \{1,3\}, \{\{1\}, \{1,2\}, \{1,3\}, \{1,2,3\}\})\,,$$

where δ is given by

	a	b
$\emptyset$	$\emptyset$	$\emptyset$
$\{1\}$	$\emptyset$	$\{2\}$
$\{2\}$	$\{2,3\}$	$\{3\}$
$\{3\}$	$\{1,3\}$	$\emptyset$
$\{1,2\}$	$\{2,3\}$	$\{2,3\}$
$\{1,3\}$	$\{1,3\}$	$\{2\}$
$\{2,3\}$	$\{1,2,3\}$	$\{3\}$
$\{1,2,3\}$	$\{1,2,3\}$	$\{2,3\}$

.

Below is the code that generates the state diagram of the D_2. The output is shown in figure 4.

```
\begin{tikzpicture}
    \node[state] (phi) {$\emptyset$};
    \node[state, accepting, right of=phi] (1) {$\{1\}$};
    \node[state, right of=1] (2) {$\{2\}$};
    \node[state, accepting, right of=2] (12) {$\{1, 2\}$};
    \node[state, below of=phi] (3) {$\{3\}$};
    \node[state, initial, initial where=above, accepting, right of=3] (13) {$\{1, 3\}$};
    \node[state, right of=13] (23) {$\{2, 3\}$};
    \node[state, accepting, right of=23] (123) {$\{1, 2, 3\}$};
```

```
    \draw (phi) edge[loop left] node{$a, b$} (phi)
          (1)   edge[above] node{$a$} (phi)
          (1)   edge[above] node{$b$} (2)
          (2)   edge[right] node{$a$} (23)
          (2)   edge[above] node{$b$} (3)
          (12)  edge[above, pos=.3, left=2pt] node{$a, b$} (23)
          (3)   edge[left] node{$b$} (phi)
          (3)   edge[below] node{$a$} (13)
          (13)  edge[loop right] node{$a$} (13)
          (13)  edge[above] node{$b$} (2)
          (23)  edge[bend left, above] node{$a$} (123)
          (23)  edge[bend left, below] node{$b$} (3)
          (123) edge[loop above] node{$a$} (123)
          (123) edge[bend left, below] node{$b$} (23);
\end{tikzpicture}
```

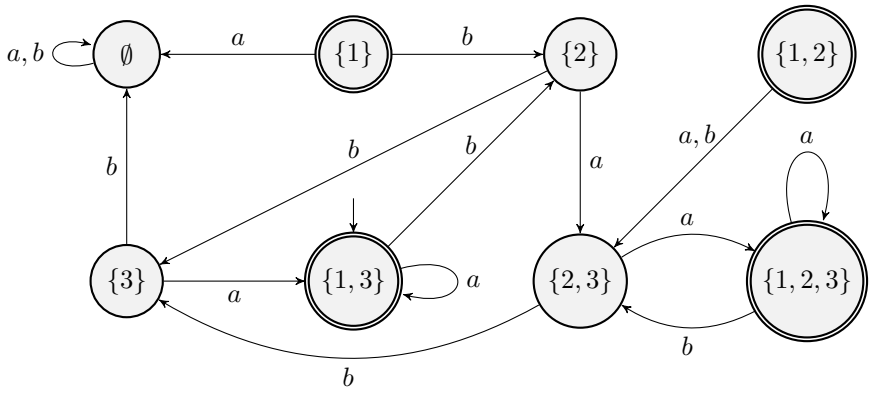

Figure 4: DFA example ([Sip12], page 58, fig 1.43)

4 Wrapping up

By this point of the tutorial, the reader should be able to draw state diagrams of arbitrary DFAs and NFAs. If needed, this tutorial will be extended to handle the state diagrams of PDAs, Turing Machines, Moore and Mealy machines.

References

[beg14] Latex for beginners. `http://www.docs.is.ed.ac.uk/skills/documents/3722/3722-2014.pdf`, 2014.

[Sip12] M. Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012. Third edition.

[Sti15] Hauke Stieler. Tikz for state-machines. `https://hauke-stieler.de/public/tikz-for-state-machines_en.pdf`, 2015.