LaTeX: Graphs and State Diagrams

Dr. Yihsiang Liow (August 6, 2024)

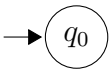
Contents

1 States

There are many drawing libraries in LaTeX. My favorite is the pgf/tikz library. For this section, I'll show you how to draw state diagrams with pgf/tikz.

Here's a single state:

Here's how to make it a start/initial state:

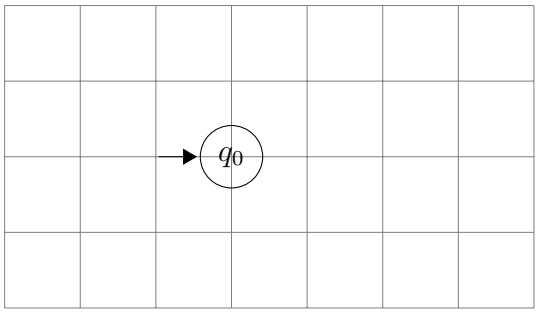


LaTeX code

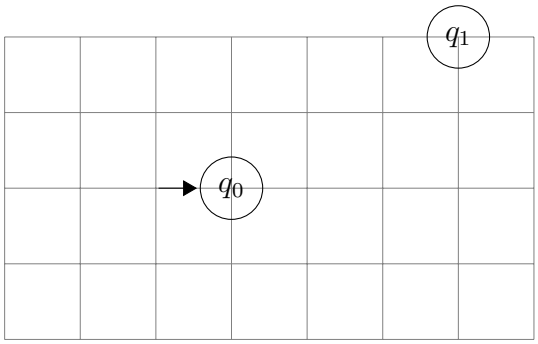
```
There are many drawing libraries in \LaTeX.
My favorite is the pgf/tikz library.
For this section, I'll show you how to draw state diagrams with pgf/tikz.

Here's a single state:
\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
\node[state]  (q_0)  {$q_0$};
;
\end{tikzpicture}
\end{center}

Here's how to make it a start/initial state:
\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
\node[state,initial]  (q_0)  {$q_0$};
;
\end{tikzpicture}
\end{center}
```

2 Grid

You can draw a grid to help you place nodes. This helper grid is from (-3, -2) to (4,2). No position is given to the initial state, so it's placed at $(0, 0)$ by default.



I'll put another state at $(3, 2)$:



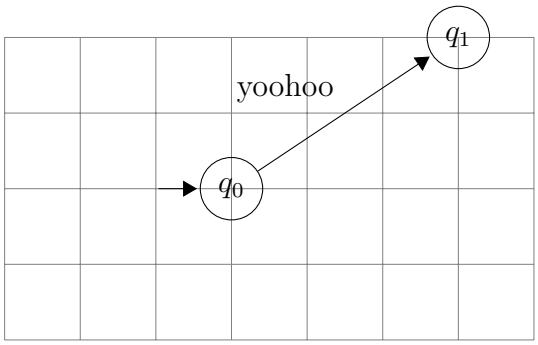
LaTeX code

```
You can draw a grid to help you place nodes.
This helper grid is from (-3, -2) to (4,2).
No position is given to the initial state, so
it's placed at $(0,0)$ by default.

\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
\draw[help lines] (-3,-2) grid (4,2);
\node[state,initial]  (q_0)  {$q_0$};
;
\end{tikzpicture}
\end{center}

I'll put another state at $(3, 2)$:
```

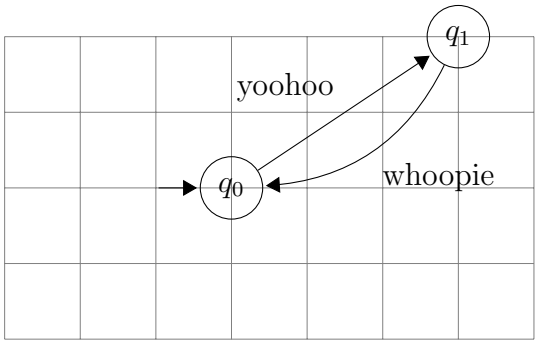
```
15
16 \begin{center}
17 \begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
18 \draw[help lines] (-3,-2) grid (4,2);
19 \node[state,initial]   (q_0)          {$q_0$};
20 \node[state]           (q_1) at (3,2) {$q_1$};
21 \end{tikzpicture}
22 \end{center}
23
```

3 Edges

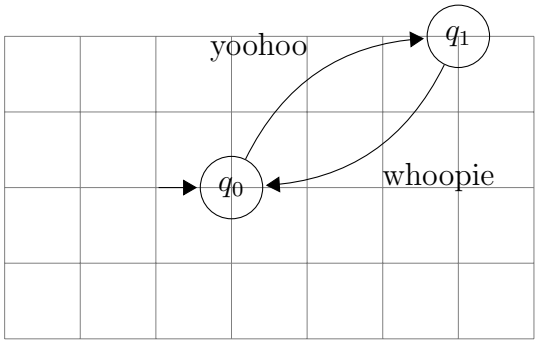
Now I'll draw an arrow:



If I want to draw an arrow from q_1 back to q_0, I had better bend it a little: If you pretend you're walking from q_1 to q_0, you basically need to bend left.



Well ... I think I'll try to make this more symmetric:



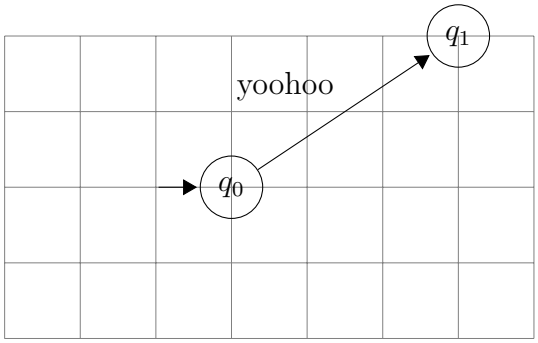
LaTeX code

```
1 Now I'll draw an arrow:
2
```

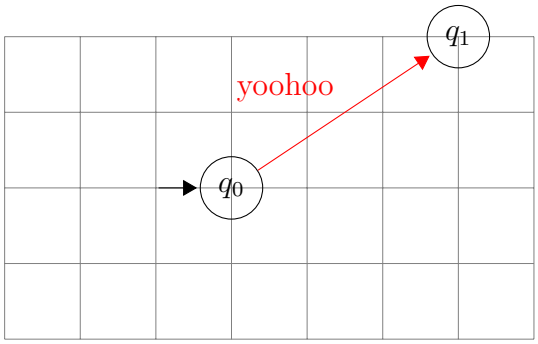
```
\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
\draw[help lines] (-3,-2) grid (4,2);
\node[state,initial]   (q_0)          {$q_0$};
\node[state]           (q_1) at (3,2) {$q_1$};

\path[->] (q_0) edge node {yoohoo} (q_1)
;
\end{tikzpicture}
\end{center}

If I want to draw an arrow from $q_1$ back to $q_0$, I had better
bend it a little:
If you pretend you're walking from $q_1$ to $q_0$, you basically
need to bend left.

\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
\draw[help lines] (-3,-2) grid (4,2);
\node[state,initial]   (q_0)          {$q_0$};
\node[state]           (q_1) at (3,2) {$q_1$};

\path[->] (q_0) edge node {yoohoo} (q_1)
          (q_1) edge [bend left] node {whoopie} (q_0)
;
\end{tikzpicture}
\end{center}

Well ... I think I'll try to make this more symmetric:

\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
\draw[help lines] (-3,-2) grid (4,2);
\node[state,initial]   (q_0)          {$q_0$};
\node[state]           (q_1) at (3,2) {$q_1$};

\path[->] (q_0) edge [bend left] node {yoohoo} (q_1)
          (q_1) edge [bend left] node {whoopie} (q_0)
;
\end{tikzpicture}
\end{center}

```

4 Modifying edges

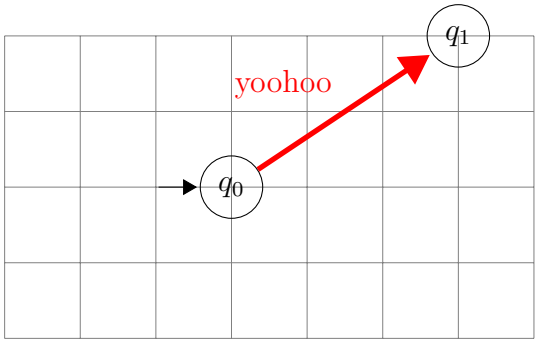
You'll note that the default arrow tip is small. You can choose a different arrow tips like this:



You can also change the color:



Note that in the above example, the change in the arrow tip affects for every edge while the change in color is for a single edge. You can also change the line width:



LaTeX code

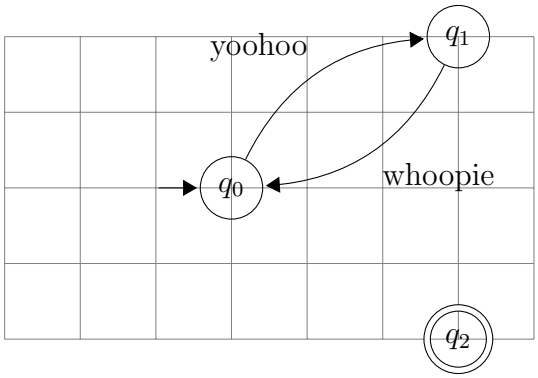
```
1 You'll note that the default arrow tip is small.
```

```
You can choose a different arrow tips like this:

\begin{center}
\begin{tikzpicture}[>=triangle 60,shorten >=1pt,node distance=2cm,
                    auto,initial text=]
\draw[help lines] (-3,-2) grid (4,2);
\node[state,initial]   (q_0)          {$q_0$};
\node[state]           (q_1) at (3,2) {$q_1$};

\path[->] (q_0) edge node {yoohoo} (q_1)
;
\end{tikzpicture}
\end{center}

You can also change the color:
\begin{center}
\begin{tikzpicture}[>=triangle 60,shorten >=1pt,node distance=2cm,
                    auto,initial text=]
\draw[help lines] (-3,-2) grid (4,2);
\node[state,initial]   (q_0)          {$q_0$};
\node[state]           (q_1) at (3,2) {$q_1$};

\path[->, red] (q_0) edge node {yoohoo} (q_1)
;
\end{tikzpicture}
\end{center}

Note that in the above example,
the change in the arrow tip affects for every edge
while the change in color is for a single edge.
You can also change the line width:
\begin{center}
\begin{tikzpicture}[>=triangle 60,
                    shorten >=1pt,node distance=2cm,
                    auto,initial text=]
\draw[help lines] (-3,-2) grid (4,2);
\node[state,initial]   (q_0)          {$q_0$};
\node[state]           (q_1) at (3,2) {$q_1$};

\path[->, red, line width=2pt] (q_0) edge node {yoohoo} (q_1)
;
\end{tikzpicture}
```

```
\end{center}

```

5 Accept state

And here's how to draw an accept state:

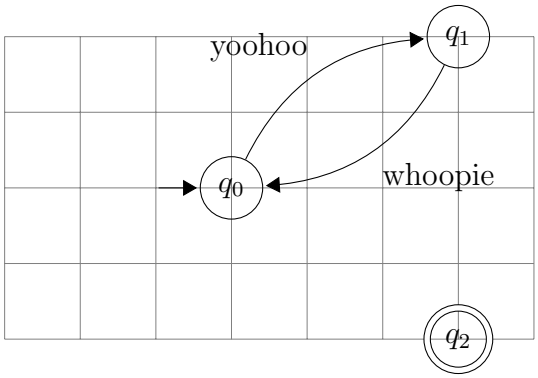


LaTeX code

```
And here's how to draw an accept state:

\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
\draw[help lines] (-3,-2) grid (4,2);
\node[state,initial]   (q_0)          {$q_0$};
\node[state]           (q_1) at (3,2) {$q_1$};
\node[state,accepting] (q_2) at (3,-2) {$q_2$};

\path[->] (q_0) edge [bend left] node {yoohoo} (q_1)
          (q_1) edge [bend left] node {whoopie} (q_0)
;
\end{tikzpicture}
\end{center}

```

6 Modifying accept states

Notice that the double-lines in the accept states are kind of close. You can change that:



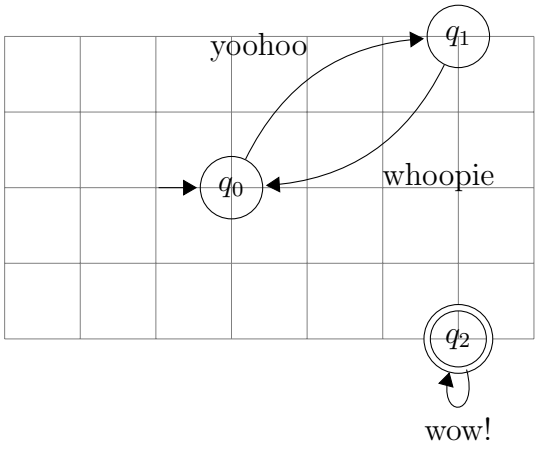
This affects all accept states.

LaTeX code

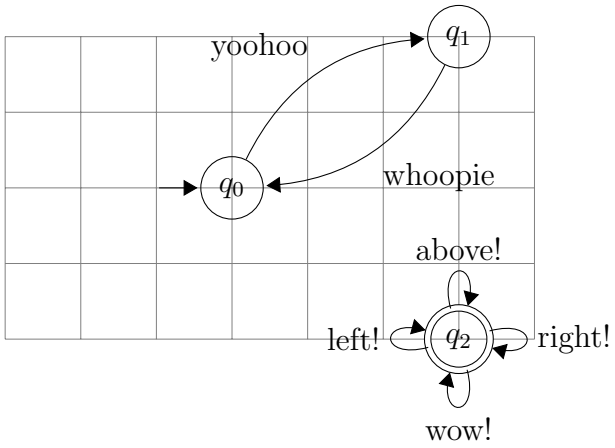
```
Notice that the double-lines in the accept states are kind of close.
You can change that:
\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=,
                    double distance=2pt]
\draw[help lines] (-3,-2) grid (4,2);
\node[state,initial]   (q_0)          {$q_0$};
\node[state]           (q_1) at (3,2) {$q_1$};
\node[state,accepting] (q_2) at (3,-2) {$q_2$};

\path[->] (q_0) edge [bend left] node {yoohoo} (q_1)
          (q_1) edge [bend left] node {whoopie} (q_0)
;
\end{tikzpicture}
\end{center}
This affects all accept states.

```

7 Loops

And here's how to add a loopy arrow (in this case it's below):



If you look at the latex code you'll see that I use loop below. You can also do loop above, loop left, and loop right.



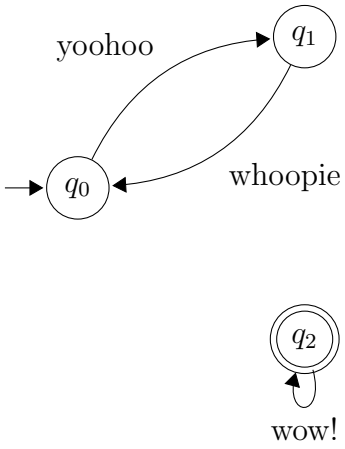
LaTeX code

```
And here's how to add a loopy arrow (in this case it's below):

\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
\draw[help lines] (-3,-2) grid (4,2);
\node[state,initial]   (q_0)          {$q_0$};
\node[state]           (q_1) at (3,2) {$q_1$};
```

```
\node[state,accepting] (q_2) at (3,-2) {$q_2$};

\path[->] (q_0) edge [bend left] node {yoohoo} (q_1)
          (q_1) edge [bend left] node {whoopie} (q_0)
          (q_2) edge [loop below] node {wow!} ()
;
\end{tikzpicture}
\end{center}

If you look at the latex code you'll see that I use loop below.
You can also do loop above, loop left, and loop right.

\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
\draw[help lines] (-3,-2) grid (4,2);
\node[state,initial]   (q_0)          {$q_0$};
\node[state]           (q_1) at (3,2) {$q_1$};
\node[state,accepting] (q_2) at (3,-2) {$q_2$};

\path[->] (q_0) edge [bend left] node {yoohoo} (q_1)
          (q_1) edge [bend left] node {whoopie} (q_0)
          (q_2) edge [loop below] node {wow!} ()
          (q_2) edge [loop above] node {above!} ()
          (q_2) edge [loop right] node {right!} ()
          (q_2) edge [loop left] node {left!} ()
;
\end{tikzpicture}
\end{center}

```

8 Remove grid

Of course once you're done drawing, you can turn off the grid:



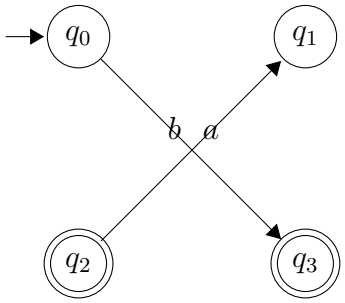
Here's an interesting web site containing some really interesting examples: http://www.texample.net/tikz/examples/

LaTeX code

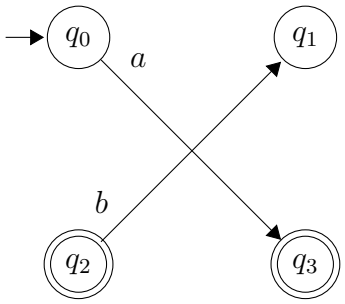
```
Of course once you're done drawing, you can turn off the grid:

\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
\node[state,initial]   (q_0)          {$q_0$};
\node[state]           (q_1) at (3,2) {$q_1$};
\node[state,accepting] (q_2) at (3,-2) {$q_2$};

\path[->] (q_0) edge [bend left] node {yoohoo} (q_1)
          (q_1) edge [bend left] node {whoopie} (q_0)
          (q_2) edge [loop below] node {wow!} ()
;
\end{tikzpicture}
\end{center}

Here's an interesting web site containing some
really interesting examples:
\url{http://www.texample.net/tikz/examples/}
```

9 Placing the Label

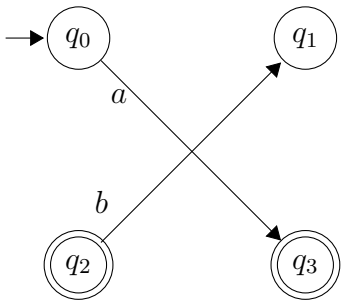
Suppose this is part of your graph:



You move the labels along the edges like this:



You can also place the label below the arrow:



LaTeX code

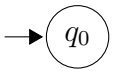
```
Suppose this is part of your graph:

\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
\node[state,initial]   (q_0)           {$q_0$};
\node[state]           (q_1) at (3,0)  {$q_1$};
```

```
\node[state,accepting] (q_2) at (0,-3) {$q_2$};
\node[state,accepting] (q_3) at (3,-3) {$q_3$};

\path[->] (q_0) edge node {$a$} (q_3)
          (q_2) edge node {$b$} (q_1)
;
\end{tikzpicture}
\end{center}

You move the labels along the edges like this:
\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
\node[state,initial]   (q_0)           {$q_0$};
\node[state]           (q_1) at (3,0)  {$q_1$};
\node[state,accepting] (q_2) at (0,-3) {$q_2$};
\node[state,accepting] (q_3) at (3,-3) {$q_3$};

\path[->] (q_0) edge node[pos=0.1] {$a$} (q_3)
          (q_2) edge node[pos=0.1] {$b$} (q_1)
;
\end{tikzpicture}
\end{center}

You can also place the label below the arrow:
\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
\node[state,initial]   (q_0)           {$q_0$};
\node[state]           (q_1) at (3,0)  {$q_1$};
\node[state,accepting] (q_2) at (0,-3) {$q_2$};
\node[state,accepting] (q_3) at (3,-3) {$q_3$};

\path[->] (q_0) edge node[below,pos=0.1] {$a$} (q_3)
          (q_2) edge node[pos=0.1] {$b$} (q_1)
;
\end{tikzpicture}
\end{center}
```

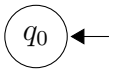
10 Start State

Here's how to change the direction of the arrow for the initial state:

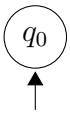
From the left (this is the default):



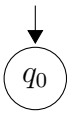
From the right:



From below:



From above:



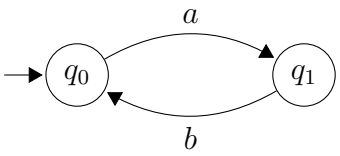
LaTeX code

```
Here's how to change the direction of the arrow for the initial state:

From the left (this is the default):
\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
\node[state,initial]   (q_0)          {$q_0$};
;
\end{tikzpicture}
\end{center}

From the right:
\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=,
                    initial where=right]
```

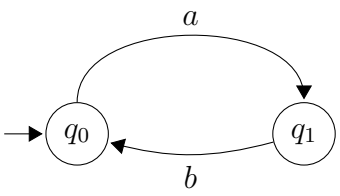
```
\node[state,initial]   (q_0)          {$q_0$};
;
\end{tikzpicture}
\end{center}

From below:
\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=,
                    initial where=below]
\node[state,initial]   (q_0)          {$q_0$};
;
\end{tikzpicture}
\end{center}

From above:
\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=,
                    initial where=above]
\node[state,initial]   (q_0)          {$q_0$};
;
\end{tikzpicture}
\end{center}
```

11 Amount of Bend

Suppose this is part of your graph:



You change the amount of bend like this:



LaTeX code

```
Suppose this is part of your graph:

\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
\node[state,initial]   (q_0)           {$q_0$};
\node[state]           (q_1) at (3,0)  {$q_1$};

\path[->] (q_0) edge [bend left] node {$a$} (q_1)
          (q_1) edge [bend left] node {$b$} (q_0)
;
\end{tikzpicture}
\end{center}

You change the amount of bend like this:
\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
\node[state,initial]   (q_0)           {$q_0$};
\node[state]           (q_1) at (3,0)  {$q_1$};

\path[->] (q_0) edge [bend left=90] node {$a$} (q_1)
          (q_1) edge [bend left=15] node {$b$} (q_0)
;
```

```
\end{tikzpicture}
\end{center}

```

12 Using `latextool_basic.py`

I have written a program to help you write the earlier LaTeX code to draw automatas. The `automata` function (in the `latextool_basic.py` module) spits out LaTeX code for state diagrams which can then be modified by hand. `latextool_basic.py` should be installed in your virtual machine.

Here are some examples.

Save this to a file say `a.py`:

```
from latextool_basic import *
print(automata(layout="""
    A
    """,
    A="initial|label=$q''_0$",
    ))
```

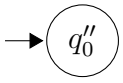
Run it like this in your bash shell:

```
python a.py
```

and you will get this LaTeX code as output:

```
\begin{center}
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto,initial text=]
\node[state,initial] (A) at (  4,  0) {$q''_0$};

\path[->]

;
\end{tikzpicture}
\end{center}
```

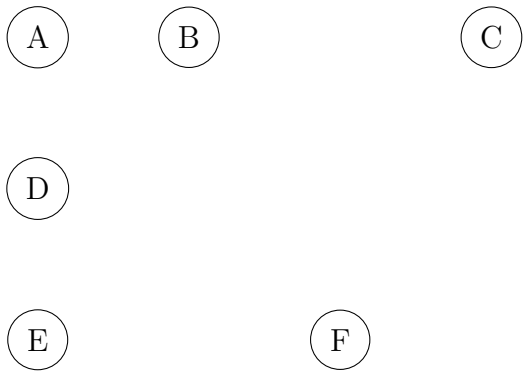
When you paste the above output into a LaTeX document (and possibly making some modifications), you will get this picture:



The `layout` parameter is a string for placing the nodes. When you execute this:

```
from latextool_basic import *
print(automata(layout="""
A B   C
D
E   F
"""
))
```

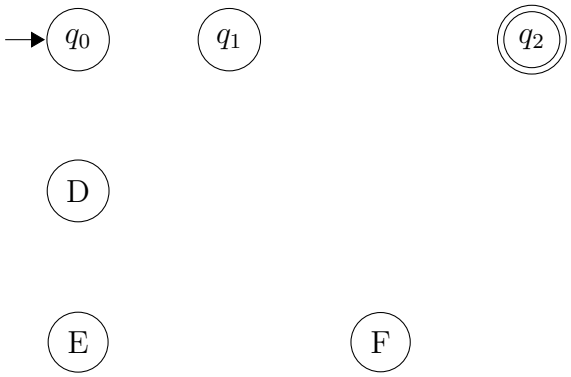
you get this output:



You can then put labels into each node.

```
from latextool_basic import *
print(automata(layout="""
A B   C
D
E   F
""",
A = "initial|label=$q_0$",
B = "label=$q_1$",
C = "accept|label=$q_2$"
))
```

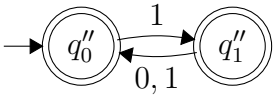
gives this picture



For the rest of the examples, I will just give you the python code and the corresponding picture.

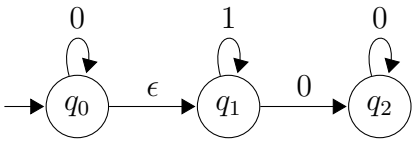
Example. Adding edges.

```
from latextool_basic import *
print(automata(layout="""
A B
""",
edges="A,$1$,B|B,$0,1$,A",
A="accept|initial|label=$q''_0$",
B="accept|label=$q''_1$",
))
```



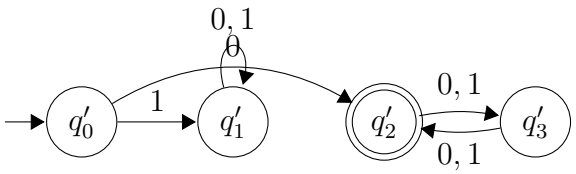
Example.

```
from latextool_basic import *
print(automata(layout="""
A B C
""",
edges="A,$\ep$,B|A,$0$,A|B,$1$,B|B,$0$,C|C,$0$,C",
A="initial|label=$q_0$",
B="label=$q_1$",
C="label=$q_2$",
))
```



Example.

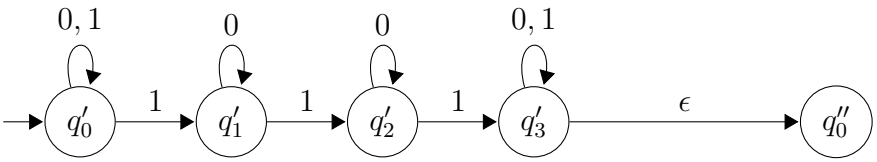
```
from latextool_basic import *
print(automata(layout="""
A D B C
""",
edges="A,$0$,B|B,$0,1$,C|C,$0,1$,B|A,$1$,D|D,$0,1$,D",
A="initial|label=$q'_0$",
B="accept|label=$q'_2$",
C="label=$q'_3$",
D="label=$q'_1$",
))
```



In this case, you have to add more bend to the edge from A to B in order to avoid the loop at D.

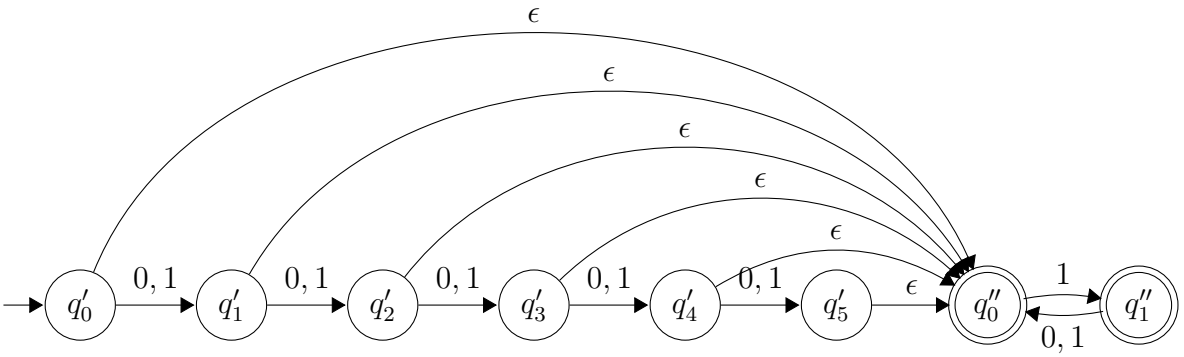
Example.

```
from latextool_basic import *
print(automata(layout="""
A B C D   E
""",
edges="A,1,B|B,1,C|C,1,D|A,$0,1$,A|D,$0,1$,D|B,0,B|C,0,C|D,$\ep$,E",
A="initial|label=$q'_0$",
B="label=$q'_1$",
C="label=$q'_2$",
D="label=$q'_3$",
E="label=$q''_0$"
))
```



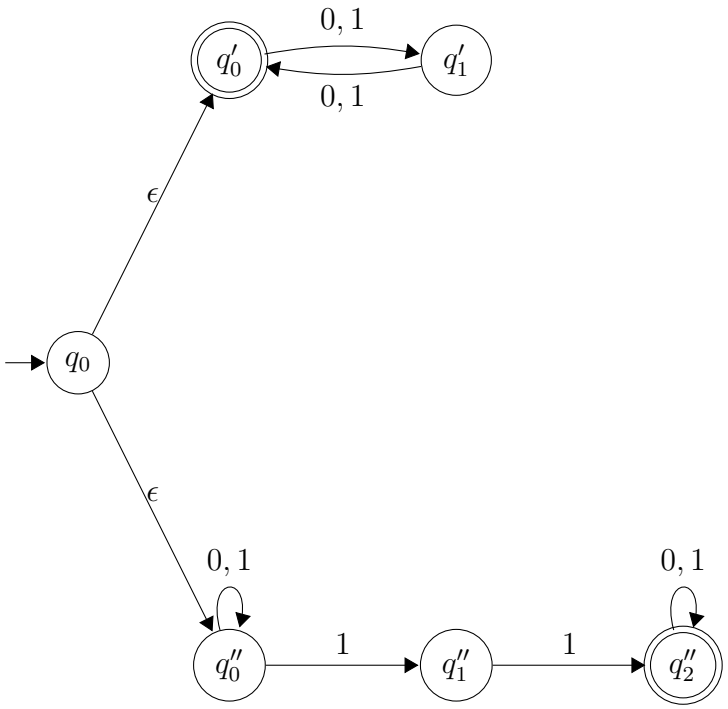
Example. You can do the following when the edge string is too long:

```
from latextool_basic import *
print(automata(layout="""
A B C D E F G H
""",
edges="A,$0,1$,B|B,$0,1$,C|C,$0,1$,D|D,$0,1$,E|E,$0,1$,F"+\
 "|F,$\epsilon$,G"+\
 "|A,$\epsilon$,G"+\
 "|B,$\epsilon$,G"+\
 "|C,$\epsilon$,G"+\
 "|D,$\epsilon$,G"+\
 "|E,$\epsilon$,G"+\
 "|G,$1$,H|H,$0,1$,G",
A="initial|label=$q'_0$",
B="label=$q'_1$",
C="label=$q'_2$",
D="label=$q'_3$",
E="label=$q'_4$",
F="label=$q'_5$",
G="accept|label=$q''_0$",
H="accept|label=$q''_1$",
))
```



Example.

```
from latextool_basic import *
print(automata(layout="""
  B  D

A

  C  E  F
""",
edges="A,$\ep$,B|B,$0,1$,D|D,$0,1$,B|A,$\ep$,C|C,$1$,E|E,$1$,F|C,$0,1$,C|F,
$0,1$,F",
A="initial|label=$q_0$",
B="accept|label=$q'_0$",
C="label=$q''_0$",
D="label=$q'_1$",
E="label=$q''_1$",
F="label=$q''_2$|accept",
))
```



13 Using dot2tex and graphviz

There's a program that can be used to automatically place states and transitions in order to avoid overlaps. This is actually a pretty difficult problem in CS. The most famous software for doing this is graphviz, a research project started at AT&T Labs.

The software dot2tex is a python program that builds on top of graphviz so that LaTeX can be added to graphviz output so that for instance nodes/states and edges/transitions can be labeled using LaTeX notation. Nodes and edges can be drawn and colored using LaTeX.

Run the following as root to install the necessary libraries:

```
dnf -y install tex-preview
dnf -y install graphviz
dnf -y install python-pip
```

Exit root. As a regular user, do

```
pip install pyparsing --user
pip install pydot --user
pip install dot2tex --user
```

Now to test dot2tex.

First, write this to file `a.dot`:

```
digraph G {
  A [label="", shape="plaintext"];     /* dummy node */
  q0 [texlbl="$q_0$", shape="ellipse"];
  q1 [texlbl="$q_1$", shape="circle"];
  q2 [texlbl="$q_2$", shape="doublecircle"];

  A -> q0;
  q0 -> q1 [texlbl="$\epsilon$"];
  q1 -> q2 [texlbl="$a$"];
  q0 -> q2 [texlbl="$b$"];
}
```

Write and save this file `mydot2tex.py`:

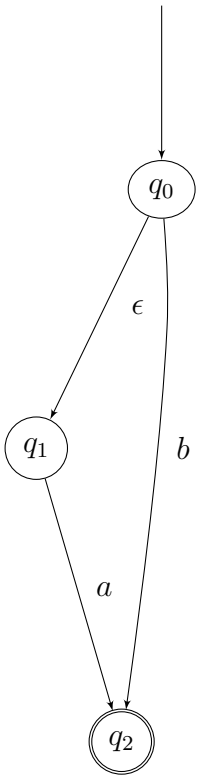
```
from dot2tex import dot2tex
import sys
filename = sys.argv[1]
s = open(filename, 'r').read()
t = dot2tex(s, preproc=True)
u = dot2tex(t, format='tikz')
print(u)
```

Then in your bash shell run

```
python mydot2tex.py a.dot > a.tex
```

This will produce `a.tex`, a LaTeX file.

You can be copy the tikz code in `a.tex` to your main LaTeX document, perhaps with minor modification. The above `a.tex` when compiled will produce this picture:



Note that the `a.dot` file only specifies the transition diagram. It does not specifies the position of the nodes or the amount of bend of the edges. Graphviz takes care of all that for you. This is not entirely trivial. Like I said this was a research project at AT&T Labs.

For more information on graphviz, just google. Here's the homepage of graphviz: https://graphviz.org/. Here's a useful website for dot2tex: https://dot2tex.readthedocs.io/en/latest/index.html.

Wait ... there's something even better ...

14 Using `dfa.py`

The file `dfa.py` contains the code to create DFAs, test run DFAs, draw DFA diagrams in LaTeX, and print formal definition of DFAs in LaTeX. `dfa.py` uses dot2tex.

I have an example folder on how to use `dfa.py`.

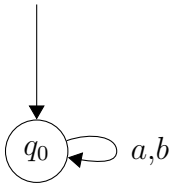
For instance here's an example. Create a file say `M0.py` with the following content:

```
from dfa import *

M0 = DFA(Q=['q0'],
         F=[],
         S=['a', 'b'],
         d={('q0','a'):'q0',
            ('q0','b'):'q0',
            },
         start='q0',
         )
M0.diagram(filename="M0")
M0.formal(filename="M0-formal")
```

When you execute this in your bash shell:

```
python M0.py
```

a file `M0-tikz.tex` is produced that can be copied into your main LaTeX file. For this example, `M0-tikz.tex` produces this diagram:



Another file created is `M0-formal.tex`. For this example, `M0-formal.tex` produces this:

$$\begin{aligned} \Sigma &= \{a, b\} \\ Q &= \{q_0\} \\ \text{start state} &= q_0 \\ F &= \{\} \end{aligned}$$

q	c	$\delta(q,c)$
q_0	a	q_0
q_0	b	q_0

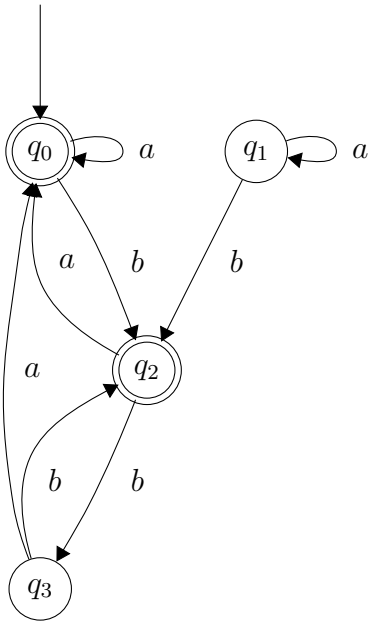
Here's another example. Here's `M1.py`:

```
from dfa import *

M1 = DFA(Q=['q0', 'q1', 'q2', 'q3'],
         F=['q0', 'q2'],
         S=['a', 'b'],
         d={('q0','a'):'q0',
            ('q0','b'):'q2',
            ('q1','a'):'q1',
            ('q1','b'):'q2',
            ('q2','a'):'q0',
            ('q2','b'):'q3',
            ('q3','a'):'q0',
            ('q3','b'):'q2',
            },
         start='q0',
         )
M1.diagram(filename="M1")
M1.formal(filename="M1-formal")
```

After executing

```
python M1.py
```

I get `M1-tikz.tex` which gives this diagram



and `M1-formal.tex` which gives this:

$$\begin{aligned}\Sigma &= \{a, b\}\\ Q &= \{q_0, q_1, q_2, q_3\}\\ \text{start state} &= q_0\\ F &= \{q_0, q_2\}\end{aligned}$$

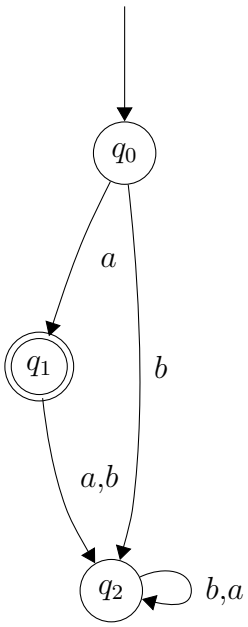
q	c	$\delta(q, c)$
q_0	a	q_0
q_0	b	q_2
q_1	a	q_1
q_1	b	q_2
q_2	a	q_0
q_2	b	q_3
q_3	a	q_0
q_3	b	q_2

Here's another example. Here's `M2.py`:

```
from dfa import *

M2 = DFA(Q=['q0', 'q1', 'q2'],
```

```
        F=['q1'],
        S=['a', 'b'],
        d={('q0','a'):'q1',
           ('q0','b'):'q2',
           ('q1','a'):'q2',
           ('q1','b'):'q2',
           ('q2','a'):'q2',
           ('q2','b'):'q2',
           },
        start='q0',
        )
M2.diagram(filename="M2")
M2.formal(filename="M2-formal")
```

which gives me

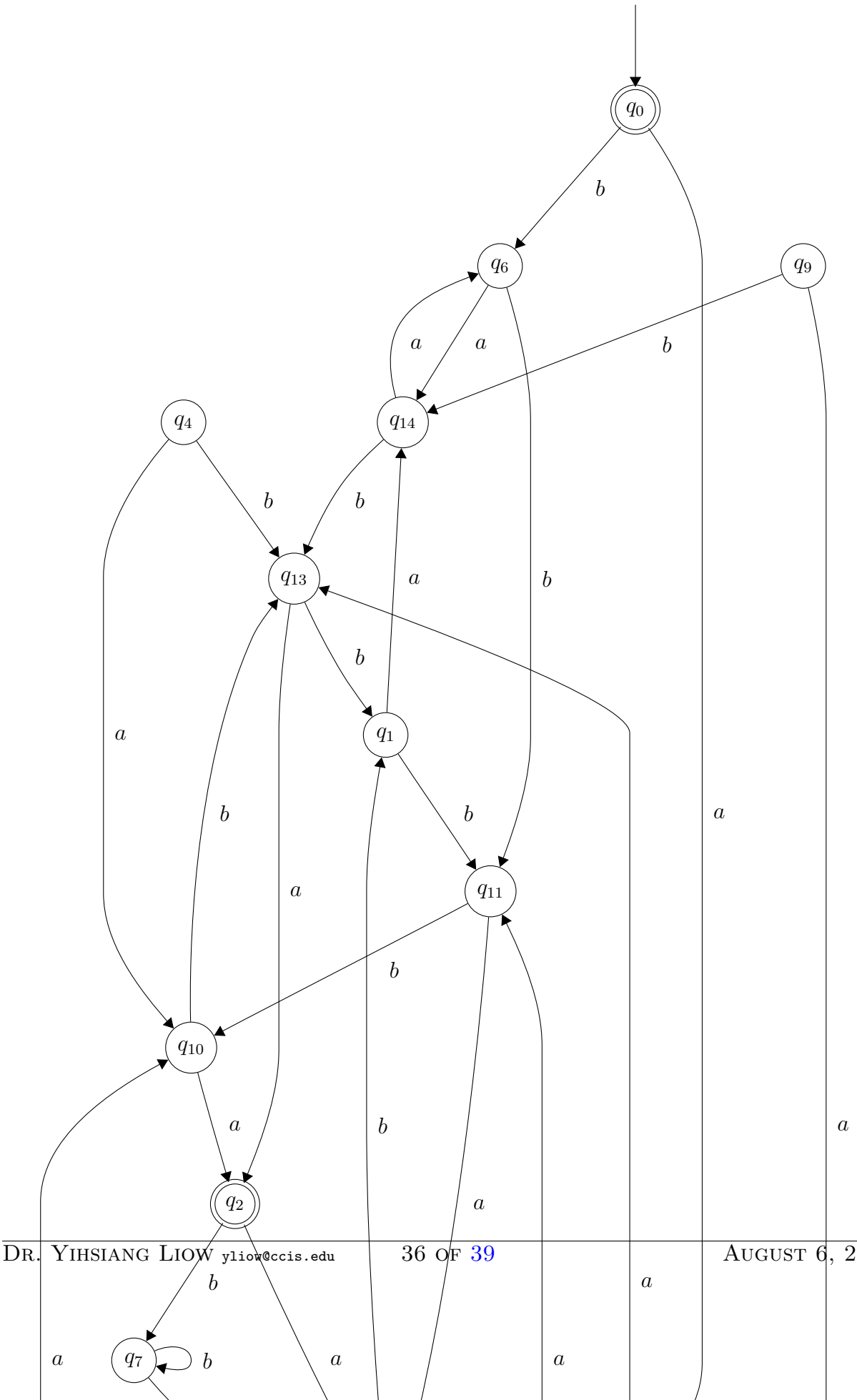


$$\Sigma = \{a, b\}$$
$$Q = \{q_0, q_1, q_2\}$$
$$\text{start state} = q_0$$
$$F = \{q_1\}$$

q	c	$\delta(q, c)$

q_0	a	q_1
q_0	b	q_2
q_1	a	q_2
q_1	b	q_2
q_2	a	q_2
q_2	b	q_2

Here's a random DFA that's too big:

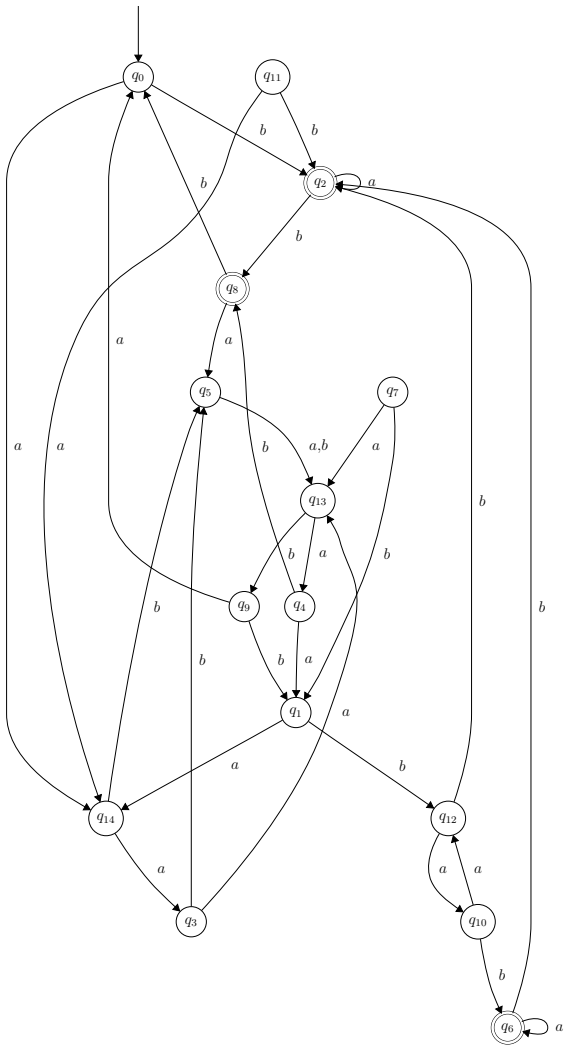

q_0
q_6
q_9
q_4
q_{14}
q_{13}
q_1
q_{11}
q_{10}
q_2
q_7
a
b

$$\Sigma = \{a, b\}$$
$$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}\}$$
$$\text{start state} = q_0$$
$$F = \{q_0, q_2, q_5\}$$

q	c	$\delta(q, c)$
q_0	a	q_{12}
q_0	b	q_6
q_1	a	q_{14}
q_1	b	q_{11}
q_2	a	q_{12}
q_2	b	q_7
q_3	a	q_{10}
q_3	b	q_8
q_4	a	q_{10}
q_4	b	q_{13}
q_5	a	q_{13}
q_5	b	q_{12}
q_6	a	q_{14}
q_6	b	q_{11}
q_7	a	q_{12}
q_7	b	q_7
q_8	a	q_{11}
q_8	b	q_5
q_9	a	q_5
q_9	b	q_{14}
q_{10}	a	q_2
q_{10}	b	q_{13}
q_{11}	a	q_{12}
q_{11}	b	q_{10}
q_{12}	a	q_3
q_{12}	b	q_1
q_{13}	a	q_2
q_{13}	b	q_1
q_{14}	a	q_6
q_{14}	b	q_{13}

When a DFA diagram is too big, do this (specify the width):

```
from dfa import *
M4 = DFA.rand(Ssize=2, Qsize=15, Fsize=3)
M4.diagram(filename="M4", width="3in")
M4.formal(filename="M4-formal")
```



$$\Sigma = \{a, b\}$$
$$Q = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}\}$$
$$\text{start state} = q_0$$
$$F = \{q_2, q_6, q_8\}$$

q	c	$\delta(q, c)$
q_0	a	q_{14}
q_0	b	q_2
q_1	a	q_{14}
q_1	b	q_{12}
q_2	a	q_2
q_2	b	q_8
q_3	a	q_{13}
q_3	b	q_5
q_4	a	q_1
q_4	b	q_8
q_5	a	q_{13}
q_5	b	q_{13}
q_6	a	q_6
q_6	b	q_2
q_7	a	q_{13}
q_7	b	q_1
q_8	a	q_5
q_8	b	q_0
q_9	a	q_0
q_9	b	q_1
q_{10}	a	q_{12}
q_{10}	b	q_6
q_{11}	a	q_{14}
q_{11}	b	q_2
q_{12}	a	q_{10}
q_{12}	b	q_2
q_{13}	a	q_4
q_{13}	b	q_9
q_{14}	a	q_3
q_{14}	b	q_5