19 Automata Drawing Library

```
\usetikzlibrary{automata} % LaTeX and plain TeX
\usetikzlibrary[automata] % ConTeXt
```

This packages provides shapes and styles for drawing finite state automata and Turing machines.

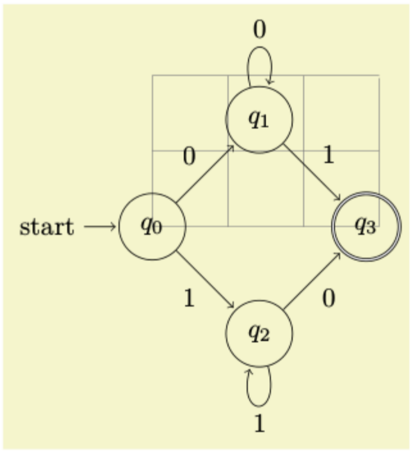
19.1 Drawing Automata

The automata drawing library is intended to make it easy to draw finite automata and Turing machines. It does not cover every situation imaginable, but most finite automata and Turing machines found in text books can be drawn in a nice and convenient fashion using this library.

To draw an automaton, proceed as follows:

1. For each state of the automaton, there should be one node with the option `state`.
2. To place the states, you can either use absolute positions or relative positions, using options like `above of` or `right of`.
3. Give a unique name to each state node.
4. Accepting and initial states are indicated by adding the options `accepting` and `initial`, respectively, to the state nodes.
5. Once the states are fixed, the edges can be added. For this, the `edge` operation is most useful. It is, however, also possible to add edges after each node has been placed.
6. For loops, use the `edge [loop]` operation.

Let us now see how this works for a real example. Let us consider a nondeterminsitic four state automaton that checks whether an contains the sequence 0*1 or the sequence 1*0.



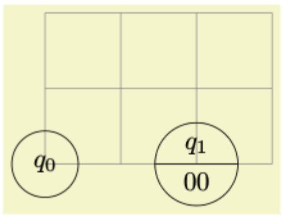
```
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,auto]
  \draw[help lines] (0,0) grid (3,2);

  \node[state,initial]  (q_0)                      {$q_0$};
  \node[state]          (q_1) [above right of=q_0] {$q_1$};
  \node[state]          (q_2) [below right of=q_0] {$q_2$};
  \node[state,accepting](q_3) [below right of=q_1] {$q_3$};

  \path[->] (q_0) edge              node        {0} (q_1)
                  edge              node [swap] {1} (q_2)
            (q_1) edge              node        {1} (q_3)
                  edge [loop above] node        {0} ()
            (q_2) edge              node [swap] {0} (q_3)
                  edge [loop below] node        {1} ();
\end{tikzpicture}
```

19.2 States With and Without Output

The state style actually just “selects” a default underlying style. Thus, you can define multiple new complicated state style and then simply set the state style to your given style to get the desired kind of styles.

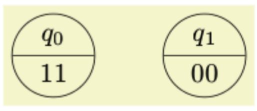
By default, the following state styles are defined:

- style=state without output This node style causes nodes to be drawn circles. Also, this style calls every state.
- style=state with output This node style causes nodes to be drawn as split circles, that is, using the circle split shape. In the upper part of the shape you have the name of the style, in the lower part the output is placed. To specify the output, use the command \nodepart{lower} inside the node. This style also calls every state.



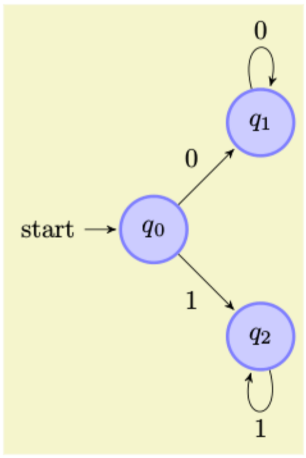
```
\begin{tikzpicture}
  \draw[help lines] (0,0) grid (3,2);

  \node[state without output] {$q_0$};

  \node[state with output] at (2,0) {$q_1$ \nodepart{lower} $00$};
\end{tikzpicture}
```

- style=state This style is set to state without output by default. You should redefine it to something else, if you wish to use states of a different nature.



```
\begin{tikzpicture}
  \tikzstyle{state}=[state with output]
  \node[state]          {$q_0$ \nodepart{lower} $11$};
  \node[state] at (2,0) {$q_1$ \nodepart{lower} $00$};
\end{tikzpicture}
```

- style=every state This style is used by state with output and also by state without output. By default, it does nothing, but you can use it to make your state look more fancy:



```
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,>=stealth']
  \tikzstyle{every state}=[draw=blue!50,very thick,fill=blue!20]

  \node[state,initial]  (q_0)                     {$q_0$};
  \node[state]          (q_1) [above right of=q_0] {$q_1$};
  \node[state]          (q_2) [below right of=q_0] {$q_2$};

  \path[->] (q_0) edge              node [above left]  {0} (q_1)
                  edge              node [below left]  {1} (q_2)
            (q_1) edge [loop above] node               {0} ()
            (q_2) edge [loop below] node               {1} ();
\end{tikzpicture}
```

19.3 Initial and Accepting States

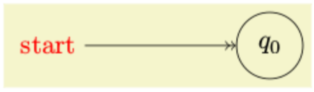
The styles initial and accepting are similar to the state style as they also just select an “underlying” style, which installs the actual settings for initial and accepting states.

For initial states, there is only one predefined way of drawing them, so the two-stage mechanism is not really necessary, but, perhaps, I will find another way of drawing them in the literature some time.

Let us start with the initial states.

- style=initial This style simply selects initial by arrow.

- style=initial by arrow This style causes an arrow and, possibly, some text to be added to the node. The arrow points from the text to the node. The node text and the direction and the distance can be set using the following options:
  - initial text=⟨*text*⟩ sets the text to be used. Use an empty text to suppress all text. The default is start.
  - initial where=⟨*direction*⟩ set the place where the text should be shown. Allowed values are above, below, left, and right.
  - intial distance=⟨*distance*⟩ is the length of the arrow leading from the text to the state node.
- style=every initial by arrow This style is executed at the beginning of every path that contains the arrow and the text. You can use it to, say, make the text red or whatever.

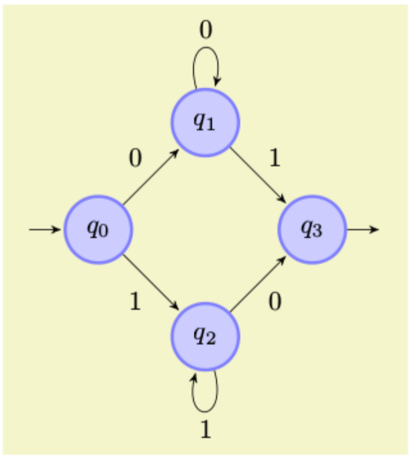


```
\begin{tikzpicture}
  \tikzstyle{every initial by arrow}=[text=red,->>]
  \node[state,initial,initial distance=2cm] {$q_0$};
\end{tikzpicture}
```

- style=initial above is a shorthand for initial by arrow,initial where=above
- style=initial below works similarly to the previous option.
- style=initial left works similarly to the previous option.
- style=initial right works similarly to the previous option.

For the accepting states, the sitation is similar: There is also an accepting style that selects the way accepting states are rendered. However, there are now two options: First, accepting by arrow, which works the same way as initial by arrow, only with the direction of arrow reversed, and accepting by double, where accepting states get a double line around them.

- style=accepting This style selects accepting by double by default. You can replace this by the style accepting by arrow to get accepting states with an arrow leaving them.
- style=accepting by double Specifies that the node should get a double line around it.
- style=accepting by arrow This style causes an arrow and, possibly, some text to be added to the node. The arrow points to the text from the node.
  - accepting text=⟨*text*⟩ sets the text to be used. Use an empty text to suppress all text. This is the default.
  - accepting where=⟨*direction*⟩ set the place where the text should be shown. Allowed values are above, below, left, and right.
  - intial distance=⟨*distance*⟩ is the length of the arrow.

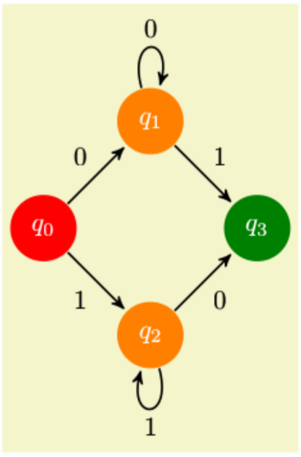


```
\begin{tikzpicture}
  [shorten >=1pt,node distance=2cm,>=stealth',initial text=]

  \tikzstyle{every state}=[draw=blue!50,very thick,fill=blue!20]
  \tikzstyle{accepting}=[accepting by arrow]

  \node[state,initial]  (q_0)                      {$q_0$};
  \node[state]          (q_1) [above right of=q_0] {$q_1$};
  \node[state]          (q_2) [below right of=q_0] {$q_2$};
  \node[state,accepting](q_3) [below right of=q_1] {$q_3$};

  \path[->] (q_0) edge              node [above left]  {0} (q_1)
                  edge              node [below left]  {1} (q_2)
            (q_1) edge              node [above right] {1} (q_3)
                  edge [loop above] node               {0} ()
            (q_2) edge              node [below right] {0} (q_3)
                  edge [loop below] node               {1} ();
\end{tikzpicture}
```

- `style=every accepting by arrow` This style is executed at the beginning of every path that contains the arrow and the text.
- `style=accepting above` is a shorthand for `accepting by arrow,accepting where=above`
- `style=accepting below` works similarly to the previous option.
- `style=accepting left` works similarly to the previous option.
- `style=accepting right` works similarly to the previous option.

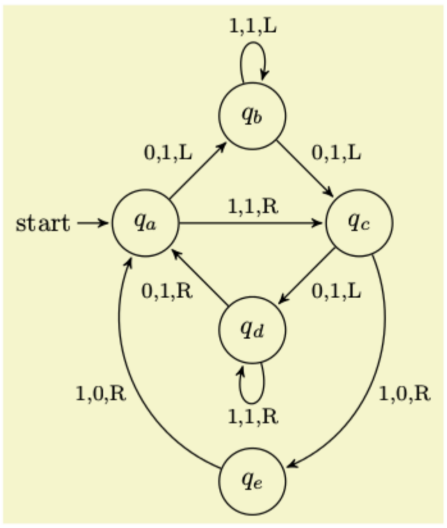
19.4 Examples

In the following example, we once more typeset the automaton presented in the previous sections. This time, we use the following rule for accepting/initial state: Initial states are red, accepting states are green, and normal states are orange. Then, we must find a path from a red state to a green state.



```
\begin{tikzpicture}[shorten >=1pt,node distance=2cm,>=stealth',thick]
  \tikzstyle{every state}=[fill,draw=none,orange,text=white]
  \tikzstyle{accepting}=[green!50!black,text=white]
  \tikzstyle{initial}=  [red,text=white]

  \node[state,initial]  (q_0)                      {$q_0$};
  \node[state]          (q_1) [above right of=q_0] {$q_1$};
  \node[state]          (q_2) [below right of=q_0] {$q_2$};
  \node[state,accepting](q_3) [below right of=q_1] {$q_3$};

  \path[->] (q_0) edge              node [above left]  {0} (q_1)
                  edge              node [below left]  {1} (q_2)
            (q_1) edge              node [above right] {1} (q_3)
                  edge [loop above] node               {0} ()
            (q_2) edge              node [below right] {0} (q_3)
                  edge [loop below] node               {1} ();
\end{tikzpicture}
```

The next example is the current candidate for the five-state busiest beaver:



```
\begin{tikzpicture}[->,>=stealth',shorten >=1pt,%
                    auto,node distance=2cm,semithick,
                    inner sep=2pt,bend angle=45]
  \node[initial,state] (A)                    {$q_a$};
  \node[state]         (B) [above right of=A] {$q_b$};
  \node[state]         (D) [below right of=A] {$q_d$};
  \node[state]         (C) [below right of=B] {$q_c$};
  \node[state]         (E) [below of=D]       {$q_e$};

  \tikzstyle{every node}=[font=\footnotesize]

  \path (A) edge              node {0,1,L} (B)
            edge              node {1,1,R} (C)
        (B) edge [loop above] node {1,1,L} (B)
            edge              node {0,1,L} (C)
        (C) edge              node {0,1,L} (D)
            edge [bend left]  node {1,0,R} (E)
        (D) edge [loop below] node {1,1,R} (D)
            edge              node {0,1,R} (A)
        (E) edge [bend left]  node {1,0,R} (A);
\end{tikzpicture}
```