

# Интеграция Python-LaTeX

с помощью пакета векторной графики PGF/TikZ

Кузнецов Максим Дмитриевич

Практикум на ЭВМ

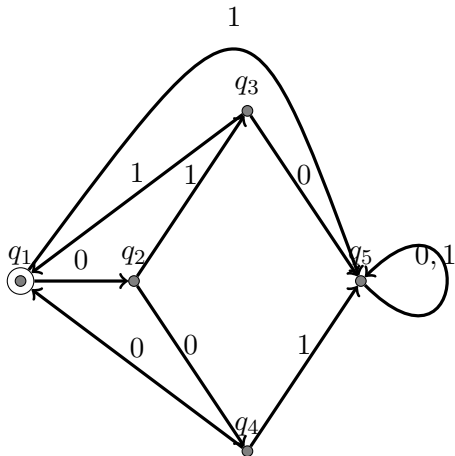
27 октября 2015

PGF/TikZ - тандем языков для создания векторной графики.

- Всегда при необходимости можно подправить визуализацию, использовать как шаблон для создания новой визуализации.
- Многие программы, работающие с векторной графикой, т.к. **Inkspace**, **GeoGebra** позволяют сохранять результаты в формате PGF/TikZ.
- Библиотеки для создания графов, деревьев, 3-d визуализаций

## Задача

Построить диаграмму Мура для конечного автомата в алфавите  $\{0, 1\}$ , допускающего все слова, состоящие из блоков 010 и 001.



## Как использовать окружение PGF/TikZ

Для использования PGF/TikZ в преамбуле документа необходимо прописать `\usepackage{tikz}`, а для вставок необходимо использовать `\begin{tikzpicture}` и `\end{tikzpicture}`.

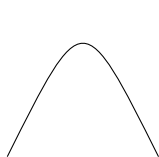
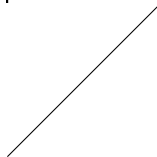
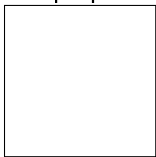
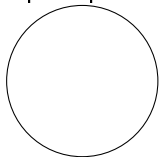
## Как использовать окружение PGF/TikZ

Для использования PGF/TikZ в преамбуле документа необходимо прописать `\usepackage{tikz}`, а для вставок необходимо использовать `\begin{tikzpicture}` и `\end{tikzpicture}`.

## Графические примитивы

Для рисования графических примитивов используется команда **draw**.

Примеры построения графических примитивов:



```
1 \begin{tikzpicture}
2   \draw (0, 0) circle [radius=1];
3   \draw (1, 1) rectangle (3, -1);
4   \draw (4, -1) -- (6, 1); %line
5   \draw (7,-1) .. controls (8, 1) .. (9, -1); %curve
6 \end{tikzpicture}
```

## Привязка метки к координате

Для привязки метки к координате можно воспользоваться `\coordinate`. Метки можно использовать как координаты во многих командах.

```
1 \coordinate (A) at (0, 0);  
2 \coordinate (B) at (1, 1);  
3 \draw (A) rectangle (B);
```

## Привязка метки к координате

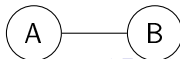
Для привязки метки к координате можно воспользоваться `\coordinate`. Метки можно использовать как координаты во многих командах.

```
1 \coordinate (A) at (0, 0);  
2 \coordinate (B) at (1, 1);  
3 \draw (A) rectangle (B);
```

## Команда node

Кроме того, есть прекрасная команда `\node`, которая позволяет делать 3 дела сразу: рисовать графический примитив, писать рядом с ним текст и привязывать координаты этого примитива к метке.

```
1 \node[draw, circle] (Node-A) at (0, 0) {A};  
2 \node[draw, circle] (Node-B) at (1, 0) {B};  
3 \draw (Node-A) -- (Node-B);
```



## Цикл

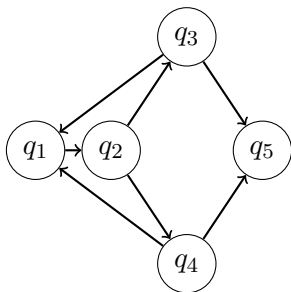
В окружении PGF/Tikz можно использовать Latex-овский цикл `foreach .. in`.



## Цикл

В окружении PGF/Tikz можно использовать Latex-овский цикл `foreach .. in`.

```
1 \foreach \name / \x / \y in {q_1/0/0, q_2/1/0, q_3/2/1.5,  
2 q_4/2/-1.5, q_5/3/0}  
3 \node[draw, circle] (\name) at (\x,\y) {$\name$};  
4  
5 \foreach \from / \to in {q_1/q_2, q_2/q_3, q_2/q_4, q_3/q_5,  
6 q_4/q_5, q_3/q_1, q_4/q_1}  
7 \draw[thick, ->] (\from) -- (\to);
```

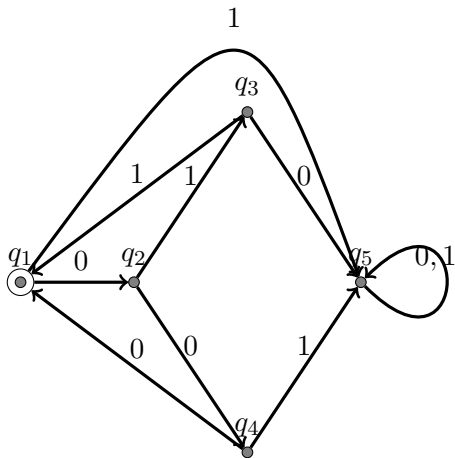


## Добавим стилей, выделение конечных состояний и загнутые рёбра.

```

1 \begin{tikzpicture}
2 %styles
3 \tikzstyle{state}=[circle, draw, fill=black!50, inner sep=0pt, minimum width=4pt,
4 label=90:\name$]
5 \tikzstyle{endstate} = [circle, draw, minimum size=10pt, inner sep=0pt]
6 \tikzstyle{edge} = [->, very thick]
7 \tikzstyle{edgelabel} = [font=\small, label={[label distance = -4pt]90:\text$}]
8
9 %draw states
10 \foreach \name / \x / \y in {q_1/0/0, q_2/1/0, q_3/2/1.5, q_4/2/-1.5, q_5/3/0}
11 \node[state] (G-\name) at (\x,\y) {};
12
13 %draw end states
14 \foreach \name / \x / \y in {q_1/0/0}
15 \node[endstate] (G-\name) at (\x,\y) {};
16
17 %draw strain edges
18 \foreach \from / \to / \text in {q_1/q_2/0, q_2/q_3/1, q_2/q_4/0, q_3/q_5/0,
19 q_4/q_5/1, q_3/q_1/1, q_4/q_1/0}
20 \draw[edge] (G-\from) -- node[edgelabel] {} (G-\to);
21
22 %draw curve edges
23 \foreach \from / \to / \text / \refpoint in {q_1/q_5/1/((2, 2.7)),
24 q_5/q_5/{0, 1}/{(4, -1) and (4, 1)}}
25 \draw[edge,auto] (G-\from) .. controls \refpoint .. node[edgelabel]{}(G-\to);
26 \end{tikzpicture}

```



## Что это?

Библиотека `matplotlib2tikz` позволяет преобразовывать графики, построенные в `matplotlib`, в код на языках PGF/TikZ.

## Как использовать?

Чтобы сохранить график в формате PGF/TikZ, до или вместо вызова `matplotlib.pyplot.show()` вставьте `matplotlib2tikz.save(filename)`. Этот файл можно вставить в `latex` командой `\include(filename)`



Пример использования:

```
1     .....
2     matplotlib2tikz.save("plot.tikz")
3     plt.show()
```

## Содержимое файла plot.tikz

```
1  \begin{axis}[
2      title={Matplotlib2tikz example},
3      xlabel={Some unit of meausure},
4      ylabel={Another unit of meausure},
5      xmin=1, xmax=4, ymin=0, ymax=16, axis on top,
6      legend style={at={(0.03,0.97)}, anchor=north west},
7      legend entries={{Curve 1},{Curve 2}}]
8      \addplot [blue]
9      coordinates {(1,1)
10     (2,2)
11     (3,9)
12     (4,16)};
13     \addplot [green!50.0!black]
14     coordinates {(1,6)
15     (2.5,8)
16     (4,5)};
17 \end{axis}
```

Сгенерированный код легкочитаем, его правка в случае необходимости потребует мало усилий.

-  Сайт TeXample с множеством готовых визуализаций (графов, деревьев и т.д.).  
<http://www.texample.net/>
-  Викиучебник PGF/TikZ.  
<https://en.wikibooks.org/wiki/LaTeX/PGF/TikZ/>