

AN OVERVIEW OF TIKZ

A LANGUAGE FOR CREATING GRAPHICS THE T_EX WAY

Till Tantau

Institute for Theoretical Computer Science
University Lübeck

GTEM Midterm Meeting 2008

OUTLINE

GOAL-ORIENTED OVERVIEW – CREATING A FIGURE

- How Do I Use TikZ?

- Recreating a Figure From a Biochemistry Textbook

- Recreating a Figure From “Die Zeit”

- Creating a Figure Depicting a Turing Machine

DESIGN-ORIENTED OVERVIEW – DESIGN PRINCIPLES

- Paths and Actions

- Special Syntax for Coordinates

- Special Syntax for Paths

- Special Syntax for Nodes

- Special Syntax for Trees

- Style Sheets

IMPLEMENTATION-ORIENTED OVERVIEW – SYSTEM STRUCTURE

- The Layers

- Top Layer: TikZ

- Middle Layer: PGF Basic Layer

- Bottom Layer: PGF System Layer

WHAT IS TIKZ?

- ▶ “TikZ ist *kein* Zeichenprogramm.”
(TikZ is not a drawing program.)
- ▶ TikZ is a T_EX macro package.
- ▶ Just as T_EX provides a special notation for formulas, TikZ provides a special notation for graphics.

FORMULAS IN T_EX – GRAPHICS IN TIKZ

In T_EX you write

Let $\int_0^1 \sqrt{x} dx$
be the integral, ...

and get

Let $\int_0^1 \sqrt{x} dx$ be the integral,
...

In TikZ you write

See $\tikz \draw[>] (0,0) -- (2ex,1ex);$
here ...

and get

See  here ...

INSTALLATION AND USAGE OF THE PACKAGE.

1. Unpack `pgf-2.00.tar.gz` in `texmf/tex/generic` and call `texhash`. (Typically already preinstalled.)
2. Add to your documents:

```
\usepackage{tikz}           % For LaTeX
\usetikzlibrary{arrows,petri,...}
```

```
\input tikz.tex             % For plain TeX
\usetikzlibrary{arrows,petri,...}
```

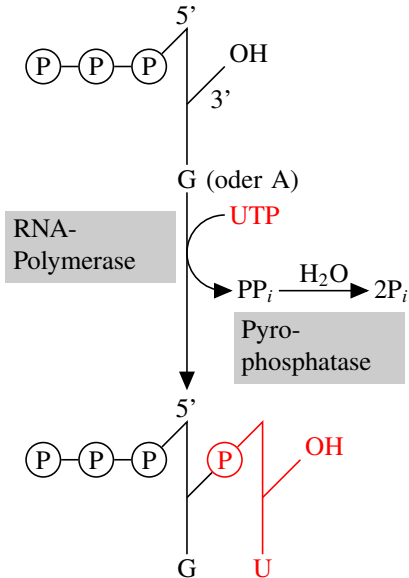
```
\usemodule[tikz]            % For ConTeXt
\usetikzlibrary[arrows,petri,...]
```

3. Process the file using one of the following:
 - ▶ `pdf(la)tex`
 - ▶ `(la)tex` and `dvips`
 - ▶ `(la)tex` and `dvipdfm`
 - ▶ `xe(la)tex` and `xdvipdfmx`
 - ▶ `vtex`
 - ▶ `textures`
 - ▶ `tex4ht`

HISTORY AND GETTING HELP

- ▶ The PGF system underlying TikZ was created for the graphics in my PhD thesis.
- ▶ The first lines of code were written around 2000.
- ▶ There are currently three core developers.
- ▶ The manual that comes with the package is around 650 pages and *very* detailed.
- ▶ There is a mailing list where people also other than myself can help you.

OUR GOAL: RECREATING THIS FIGURE.



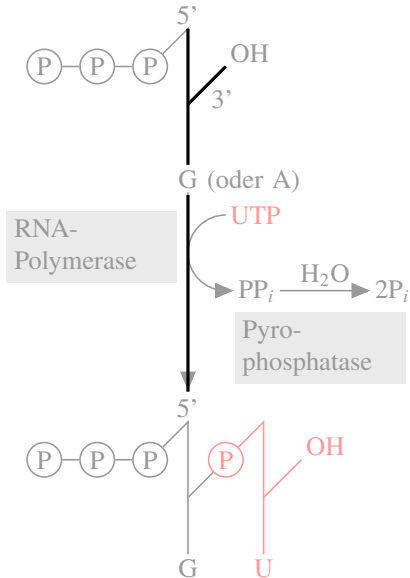
Our aim is to create this figure using TikZ.

The figure is a redrawing of the figure on page 128 of the text book



Chirsten Jaussi
Biochemie
Springer-Verlag, 2005

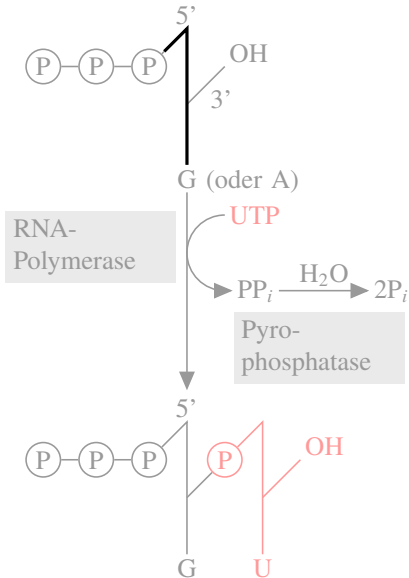
DRAWING A SIMPLE LINE.



```
\begin{tikzpicture}
  \draw (5mm,59mm) -- (5mm,41mm);
  \draw (5mm,49mm) -- (10mm,54mm);
  \draw (5mm,37mm) -- (5mm,11mm);
  ...
\end{tikzpicture}
```

- ▶ TikZ-commands have to be given in a `{tikzpicture}` environment.
- ▶ The picture size is calculated **automatically**.
- ▶ First command: `\draw`.

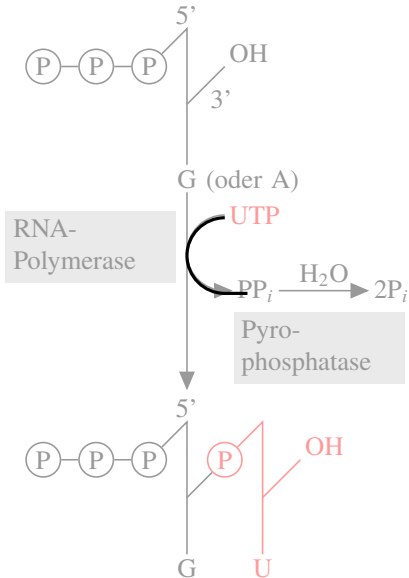
A PATH CONSISTING OF STRAIGHT LINES.



```
\begin{tikzpicture}
  \draw (0mm,54mm) -- (5mm,59mm)
    -- (5mm,41mm);
  ...
\end{tikzpicture}
```

- ▶ The `\draw` command is followed by a **path**.
- ▶ The path starts with a **coordinate**.
- ▶ The path can be continued in straight lines using `--`.

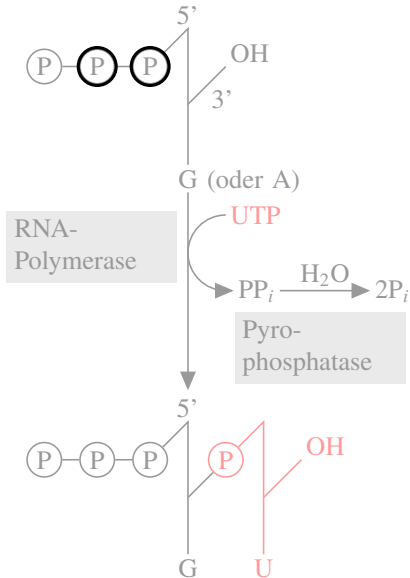
A PATH CONTAINING CURVES.



```
\begin{tikzpicture}
  \draw (10mm,34mm) arc (90:270:5mm)
    -- ++(3mm,0mm);
  ...
\end{tikzpicture}
```

- An arc can be added to a path using `arc`.
- The parameters of `arc` are
 1. start angle,
 2. end angle and
 3. radius.
- A coordinate prefixed by `++` is relative.

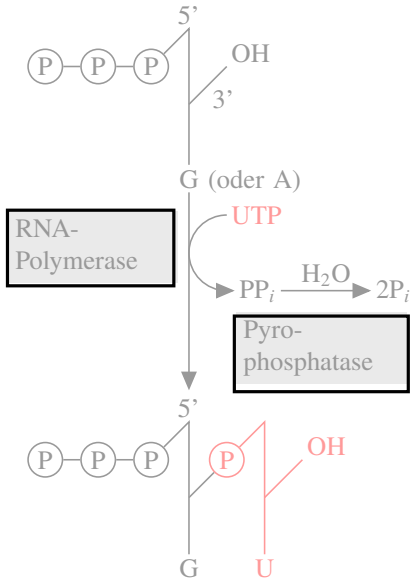
A PATH CONTAINING CIRCLES.



```
\begin{tikzpicture}
  \draw (0mm,54mm) circle (2.5mm);
  \draw (-7mm,54mm) circle (2.5mm);
  ...
\end{tikzpicture}
```

- ▶ A circle can be added to a path using `circle`.
- ▶ The radius is given in parentheses, the center is given by the previous coordinate.

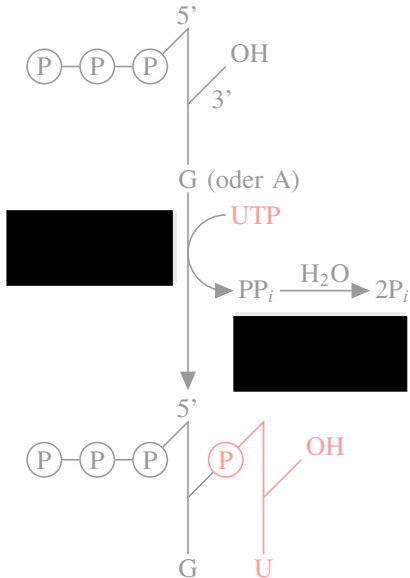
A PATH WITH TWO RECTANGLES.



```
\begin{tikzpicture}
  \draw (-19mm,25mm) -- (-19mm,35mm)
    -- (3mm,35mm) -- (3mm,25mm)
    -- cycle
    (11mm,21mm) rectangle (34mm,11mm);
  ...
\end{tikzpicture}
```

- ▶ A path may consist of several parts.
- ▶ A part can be closed using `--cycle`.
- ▶ A rectangle can be created using `rectangle`.

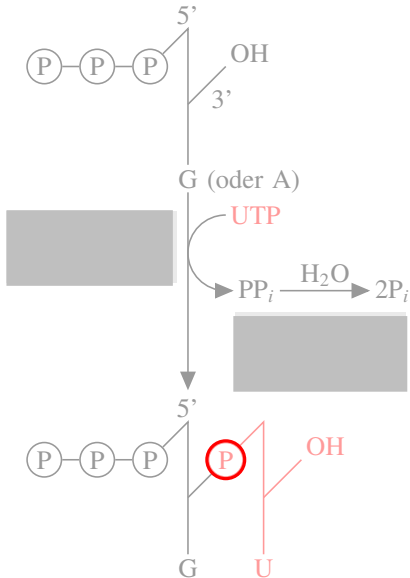
PATHS CAN BE FILLED.



```
\begin{tikzpicture}
  \fill
    (-19mm,25mm)rectangle (3mm,35mm)
    (11mm,21mm) rectangle (34mm,11mm);
  ...
\end{tikzpicture}
```

- ▶ The `\fill` command fills a path.
- ▶ It is possible to fill and draw a path.

COLORS ARE SPECIFIED USING OPTIONS.



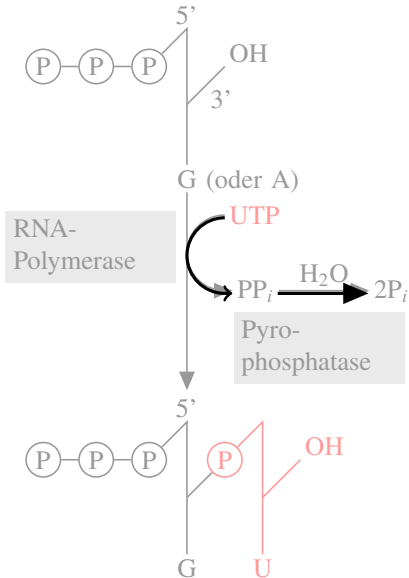
```
\fill[lightgray]
(-19mm,25mm) rectangle ++(22mm,10mm)
(11mm,21mm) rectangle ++(23mm,-10mm);
```

```
\draw[red]
(10mm,2mm) circle (2.5mm);
```

```
...
\end{tikzpicture}
```

- Colors are specified using options given in square brackets.

ARROW TIPS ARE SPECIFIED USING OPTIONS.

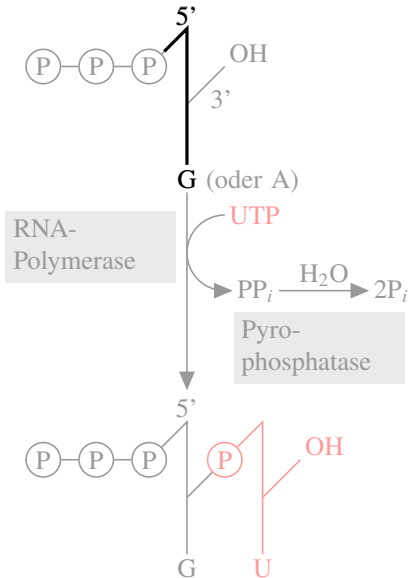


```
\begin{tikzpicture}
  \draw [->]
    (10mm,34mm) arc (90:270:5mm)
    -- (11mm,24mm);

  \draw [-triangle 45]
    (17mm,24mm) -- (27mm,24mm);
  ...
\end{tikzpicture}
```

- ▶ Arrow tips are set using an option with a hyphen in the middle.
- ▶ Whatever is left of the hyphen specifies the start arrow tip.
- ▶ Whatever is right of the hyphen specifies the end arrow tip.
- ▶ There are numerous predefined arrow tips.

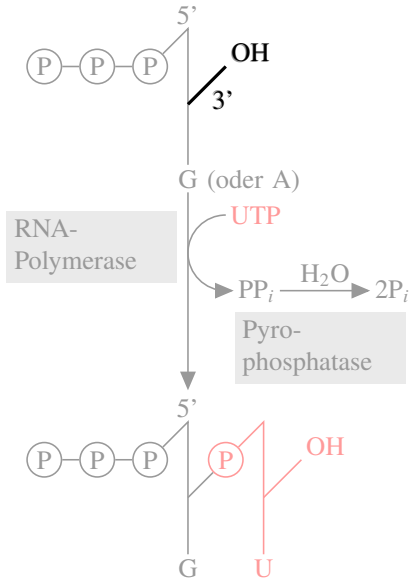
LABELS ARE ADDED USING NODES.



```
\begin{tikzpicture}
  \draw (2mm,56mm)
    -- (5mm,59mm) node [above] {5'}
    -- (5mm,41mm) node [below] {G};
  ...
\end{tikzpicture}
```

- ▶ Nodes are used for adding text.
- ▶ The preceding coordinate and options specify the exact placement.
- ▶ The node text is given in curly braces.
- ▶ Nodes are added after the path has been drawn and filled.

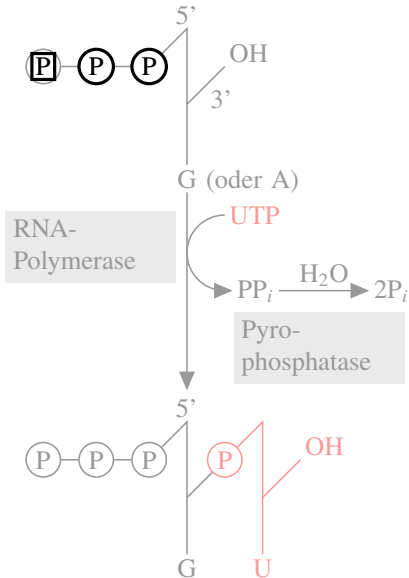
EDGE LABELS ARE ALSO ADDED USING NODES.



```
\begin{tikzpicture}
  \draw (5mm,49mm) -- (10mm,54mm)
  node [above right] {OH}
  node [midway,below right] {3'};
  ...
\end{tikzpicture}
```

- It is possible to add multiple nodes at the same place.
- The `midway` option will place a node at the middle of the previous path segment.

NODES CAN HAVE SPECIAL SHAPES.



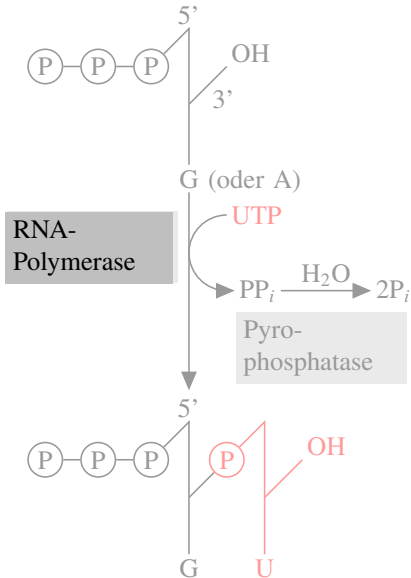
```
\begin{tikzpicture}
  \draw
    (-14mm,54mm) node [draw] {P};

  \draw
    (-7mm,54mm) node [circle,draw]{P};

  \node at (0mm,54mm)[circle,draw]{P};
  ...
\end{tikzpicture}
```

- ▶ The first path does not contain any lines. Nothing is drawn.
- ▶ The `draw` option specifies that the node's shape should be drawn.
- ▶ The `circle` specifies a circular shape.
- ▶ The `\node` command is just an abbreviation.

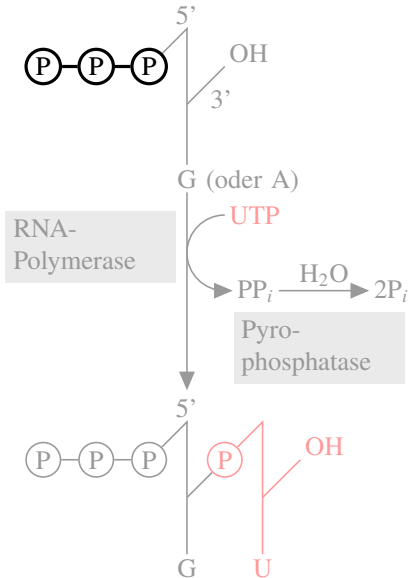
NODES CAN BE FILLED.



```
\begin{tikzpicture}
  \node at (3mm,35mm)
    [below left,
     fill=lightgray,
     text width=2cm]
    {RNA-\\Polymerase};
  ...
\end{tikzpicture}
```

- Use `text width` to specify a node's (text) width.
- Use `fill=` to specify a color for filling.

NODES CAN BE NAMED.

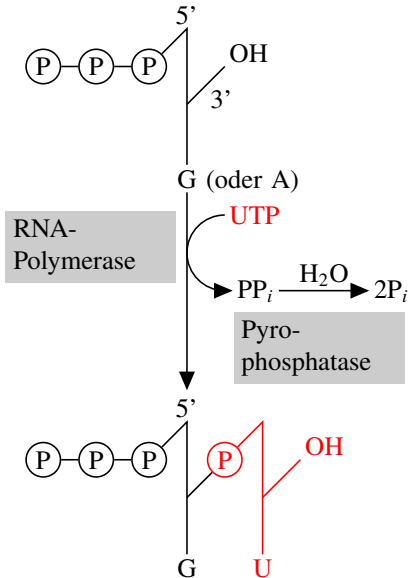


```
\begin{tikzpicture}
  \node at (-14mm,54mm)
    [circle,draw,name=p1] {P};
  \node at (-7mm,54mm)
    [circle,draw,name=p2] {P};
  \node at (0mm,54mm)
    [circle,draw,name=p3] {P};

  \draw (p1) -- (p2) -- (p3);
\end{tikzpicture}
```

- ▶ You can assign a name to a node using `name=`.
- ▶ Later, a named node can be used “like a coordinate.”

THE COMPLETE PICTURE.



The whole picture can be created using the just-described methods.

BASIC DESIGN PRINCIPLES UNDERLYING TikZ.

1. Pictures consist of **path**, to which **actions** are applied.
2. Special syntax for **coordinates**.
3. Special syntax for **paths**.
4. Special syntax for **nodes**.
5. Special syntax for **trees**.
6. **Style sheets** configure the way things look.

DESIGN PRINCIPLE: PATHS AND ACTIONS

THE CONCEPT

DESIGN PRINCIPLE

All TikZ graphics consist of **paths** to which one or more **actions** are applied.

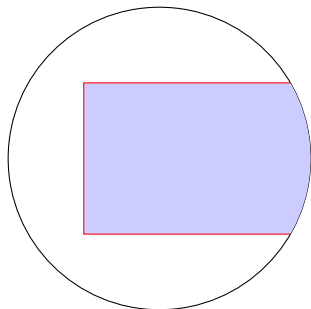
Actions are specified using options:

- ▶ **draw** will draw (stroke) a path.
- ▶ **fill** will fill a path.
- ▶ **shade** will shade the path.
- ▶ **pattern** will fill the path using a pattern.
- ▶ **clip** will clip the rest of the figure against the path.

The command **\draw** is an abbreviation for **\path[draw]**.

DESIGN PRINCIPLE: PATHS AND ACTIONS

EXAMPLES



```
\begin{tikzpicture}  
  \path[draw,clip] (0,0) circle (2cm);  
  \path[draw=red,fill=blue!20] (-1,-1) rectangle (3,1);  
\end{tikzpicture}
```

DESIGN PRINCIPLE: SYNTAX FOR COORDINATES

THE CONCEPT

DESIGN PRINCIPLE

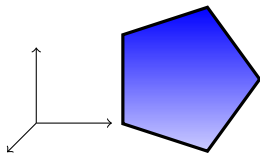
Coordinates are given in parentheses. Different coordinate systems are possible.

Supported coordinate systems:

- ▶ Cartesian
- ▶ affine
- ▶ polar 2D
- ▶ isometric 3D
- ▶ barycentric
- ▶ user defined

DESIGN PRINCIPLE: SYNTAX FOR COORDINATES

EXAMPLES



```
\begin{tikzpicture}
  \draw [->] (0,0,0) -- (1,0,0);
  \draw [->] (0,0,0) -- (0,1,0);
  \draw [->] (0,0,0) -- (0,0,1);
\end{tikzpicture}
```

```
\begin{tikzpicture}
  \draw [top color=blue,bottom color=blue!20,draw,very thick]
    (0:1cm)--(72:1cm)--(144:1cm)--(216:1cm)--(288:1cm)--cycle;
\end{tikzpicture}
```

DESIGN PRINCIPLE: SYNTAX FOR PATHS

THE CONCEPT

DESIGN PRINCIPLE

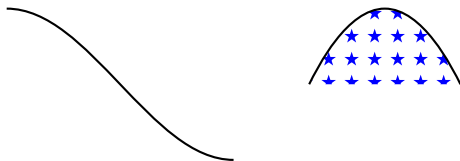
Paths are specified using a sequence of path extension operations.

Possible path operations:

- ▶ Starting a new path part.
- ▶ `--` extends the path in a straight line.
- ▶ `arc` extends the path using an arc.
- ▶ `..` extends the path using a Bézier curve.
- ▶ `parabola` extends the path using a parabola.
- ▶ `sin` extends the path using a sine curve.
- ▶ `plot` extends the path based on plot data.
- ▶ `to` extends the path using a user-defined method.
- ▶ ...

DESIGN PRINCIPLE: SYNTAX FOR PATHS

EXAMPLES



```
\begin{tikzpicture}[thick]
  \draw (0,1) cos (1.5,0) sin (3,-1);

  \draw [pattern=fivepointed stars,pattern color=blue!80]
    (4,0) parabola[parabola height=1cm] (6,0);
\end{tikzpicture}
```

DESIGN PRINCIPLE: SYNTAX FOR NODES

THE CONCEPT

DESIGN PRINCIPLE

Nodes are put at certain places along a path. Nodes have a **shape** and a **text label**.

Possible shapes:

- ▶ rectangle
- ▶ circle
- ▶ ellipse
- ▶ diamond
- ▶ breakdown diode IEC
- ▶ ...

DESIGN PRINCIPLE: SYNTAX FOR NODES

EXAMPLES



```
\begin{tikzpicture}
  \node at (0,0)
    [forbidden sign,line width=1ex,draw=red,draw opacity=.8]
    {Smoking};

  \node at (4,0)
    [ellipse,top color=white,bottom color=lightgray]
    {smoke};
\end{tikzpicture}
```

DESIGN PRINCIPLE: SYNTAX FOR TREE

THE CONCEPT

DESIGN PRINCIPLE

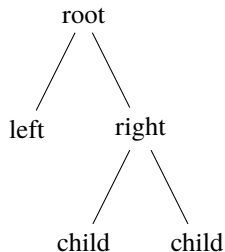
The **child** operation adds children to a node. Trees are created recursively using this operation.

The appearance of trees is governed by options:

- ▶ The sibling and parent-to-child distance.
- ▶ The child's shape.
- ▶ The appearance of the line connecting a parent and its child.

DESIGN PRINCIPLE: SYNTAX FOR TREE

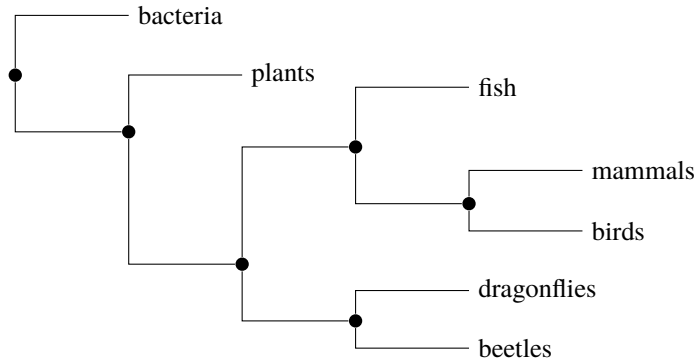
BASIC EXAMPLE



```
\begin{tikzpicture}
  \node {root}
    child {node {left}}
    child {node {right}
      child {node {child}}
      child {node {child}}
    };
\end{tikzpicture}
```

DESIGN PRINCIPLE: SYNTAX FOR TREE

COMPLEX EXAMPLE



```
...  
\node[inner node]{}  
  child { node {bacteria} }  
  child { node[inner node] {}  
    child { node {plants} }  
  }  
...
```

DESIGN PRINCIPLE: STYLE SHEETS

THE CONCEPT

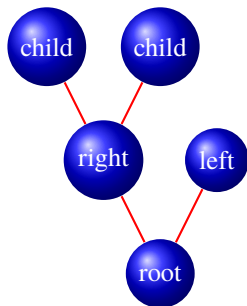
DESIGN PRINCIPLE

A **style** is a configurable set of options that are automatically or explicitly set in certain situations.

- ▶ You define a style named `foo` by saying `foo/.style=some options`.
- ▶ Using `foo` anywhere will insert `some options`.
- ▶ Styles can use other styles.
- ▶ Extensive use of styles makes code more readable and graphics more consistent (similar to HTML and CSS).

DESIGN PRINCIPLE: STYLE SHEETS

AN EXAMPLE



```
\begin{tikzpicture}
  [edge from parent/.style=
    {draw,red,thick},
   every node/.style=
    {circle,
     ball color=blue,
     text=white},
   grow=up]

  \node {root}
  child {node {left}}
  child {node {right}}
    child {node {child}}
    child {node {child}}
  };
\end{tikzpicture}
```

THE LAYERS BELOW TikZ.

TikZ is part of the **pgf package** and it just provides a “simple syntax”:

1. Top layer: **TikZ Syntax**

- ▶ Easy to use for humans.
- ▶ Succinct.
- ▶ Slow.

2. Middle layer: **pgf base layer**

- ▶ $\text{\TeX}$ macros for creating figures.
- ▶ Easy to use for other packages.
- ▶ Verbose.
- ▶ Quick.

3. Bottom layer: **pgf system layer**

- ▶ Minimalistic set of $\text{\TeX}$ macros for creating figures.
- ▶ Different implementation for each backend driver.
- ▶ Extremely difficult to use.
- ▶ Extremely fast (as fast as normal $\text{\TeX}$).

LET'S TRACE A COMMAND.

We trace the following command through the layers:

```
\draw (0,0) -- (30:10pt) -- (60:10pt) -- cycle;
```

It looks like this: 

TRANSFORMATION DONE BY TikZ.

The command

```
\draw (0,0) -- (30:10pt) -- (60:10pt) -- cycle;
```

is translated to the following PGF basic layer code by TikZ:

```
\pgfpathmoveto{\pgfpointxy{0}{0}}  
\pgfpathlineto{\pgfpointpolar{30}{10pt}}  
\pgfpathlineto{\pgfpointpolar{60}{10pt}}  
\pgfpathclose  
\pgfusepath{draw}
```

TRANSFORMATIONS DONE BY THE PGF BASIC LAYER.

The commands

```
\pgfpathmoveto{\pgfpointxy{0}{0}}  
\pgfpathlineto{\pgfpointpolar{30}{10pt}}  
\pgfpathlineto{\pgfpointpolar{60}{10pt}}  
\pgfpathclose  
\pgfusepath{draw}
```

are translated to the following PGF system layer command:

```
\pgfsys@moveto{0pt}{0pt}  
\pgfsys@lineto{8.660254pt}{5pt}  
\pgfsys@lineto{5pt}{8.660254pt}  
\pgfsys@closepath  
\pgfsys@stroke
```

TRANSFORMATIONS DONE BY THE PGF SYSTEM LAYER.

GENERATION OF SPECIAL COMMANDS FOR DVIPS.

The commands

```
\pgfsys@moveto{0pt}{0pt}  
\pgfsys@lineto{8.660254pt}{5pt}  
\pgfsys@lineto{5pt}{8.660254pt}  
\pgfsys@closepath  
\pgfsys@stroke
```

are translated to the following for dvips:

```
\special{ps:: 0 0 moveto}  
\special{ps:: 8.627899 4.98132 lineto}  
\special{ps:: 4.98132 8.627899 lineto}  
\special{ps:: closepath}  
\special{ps:: stroke}
```

TRANSFORMATIONS DONE BY THE PGF SYSTEM LAYER.

GENERATION OF SPECIAL COMMANDS FOR PDFTEX.

The commands

```
\pgfsys@moveto{0pt}{0pt}  
\pgfsys@lineto{8.660254pt}{5pt}  
\pgfsys@lineto{5pt}{8.660254pt}  
\pgfsys@closepath  
\pgfsys@stroke
```

are translated to the following for pdf_{tex}:

```
\special{pdf: 0 0 m}  
\special{pdf: 8.627899 4.98132 l}  
\special{pdf: 4.98132 8.627899 l}  
\special{pdf: h}  
\special{pdf: S}
```

TRANSFORMATIONS DONE BY THE PGF SYSTEM LAYER.

GENERATION OF SPECIAL COMMANDS FOR TEX4HT.

The commands

```
\pgfsys@moveto{0pt}{0pt}  
\pgfsys@lineto{8.660254pt}{5pt}  
\pgfsys@lineto{5pt}{8.660254pt}  
\pgfsys@closepath  
\pgfsys@stroke
```

are translated to the following for tex4ht:

```
\special{t4ht=<path d="M 0 0  
L 8.660254 5  
L 5 8.660254  
Z"  
style="stroke">}
```

TikZ COMES WITH SEVERAL LIBRARIES

- ▶ A **TikZ library** provides **additional features** or **additional options**.
- ▶ You include a library by saying `\usetikzlibrary{some lib}`.
- ▶ The list of libraries includes:
 - ▶ Additional **arrow tips**.
 - ▶ Drawing **automata**, **E/R-diagrams**, **mind maps** and **Petri nets**.
 - ▶ Adding **backgrounds** to pictures.
 - ▶ Drawing **calendars**.
 - ▶ Forming connected **chains** of nodes.
 - ▶ **Decorating** paths.
 - ▶ Predefined **transparency patterns**.
 - ▶ **Fitting** nodes around a set of coordinates.
 - ▶ Filling **patterns**.
 - ▶ Additional **shapes**.

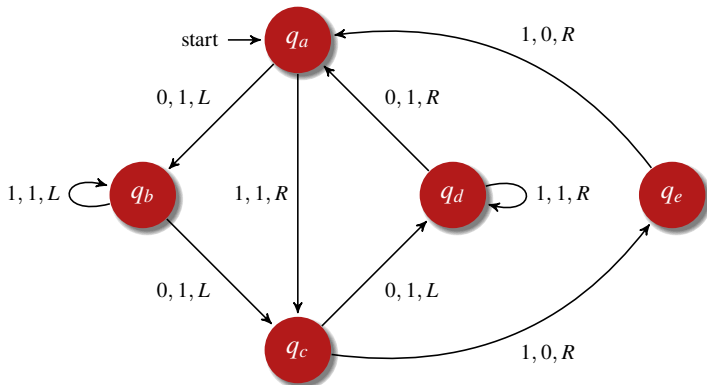
LIBRARY: ARROWS

A LIBRARY DEFINING ADDITIONAL ARROW TIPS

	<code>\usetikzlibrary{arrows}</code>	
	<code>...</code>	
	<code>\draw[-to]</code>	<code>(0,7) -- (2,7);</code>
	<code>\draw[-latex]</code>	<code>(0,6) -- (2,6);</code>
	<code>\draw[-triangle 60]</code>	<code>(0,5) -- (2,5);</code>
	<code>\draw[-angle 45]</code>	<code>(0,4) -- (2,4);</code>
	<code>\draw[-hooks]</code>	<code>(0,3) -- (2,3);</code>
	<code>\draw[-)]</code>	<code>(0,2) -- (2,2);</code>
	<code>\draw[-diamond]</code>	<code>(0,1) -- (2,1);</code>
	<code>\draw[double,-implies]</code>	<code>(0,0) -- (2,0);</code>

LIBRARY: AUTOMATA

A LIBRARY DEFINING STYLES FOR DRAWING AUTOMATA



LIBRARY: AUTOMATA

A LIBRARY DEFINING STYLES FOR DRAWING AUTOMATA

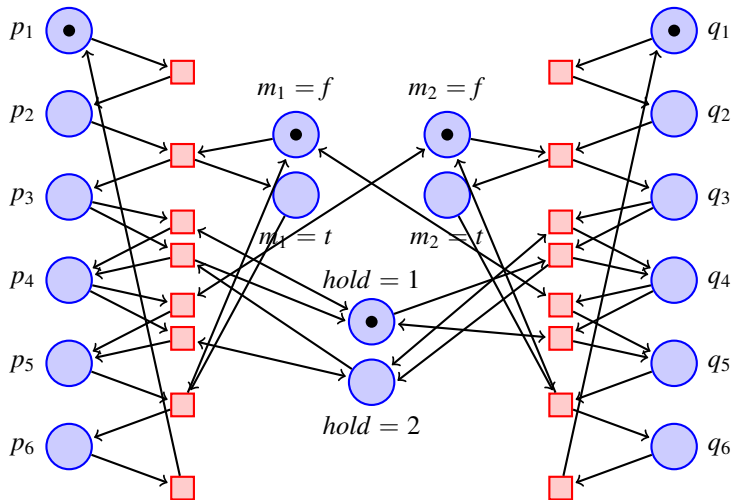
```
\usetikzlibrary{automata}
\begin{tikzpicture}
  [->,auto=right, node distance=2cm,
   >=stealth', shorten >=1pt, semithick,
   every state/.style={draw=none, fill=structure.fg,
                       text=white, circular drop shadow},
   every edge/.style={font=\footnotesize, draw}]

  \node[initial,state] (q_a)                {$q_a$};
  \node[state]          (q_b) [below left=of q_a] {$q_b$};
  \node[state]          (q_d) [below right=of q_a] {$q_d$};
  \node[state]          (q_c) [below right=of q_b] {$q_c$};
  \node[state]          (q_e) [right=of q_d]       {$q_e$};

  \draw (q_a) edge                node {$0,1,L$} (q_b)
          edge                node {$1,1,R$} (q_c)
        (q_b) edge [loop left]  node {$1,1,L$} (q_b)
          edge                node {$0,1,L$} (q_c)
        (q_c) edge                node {$0,1,L$} (q_d)
          edge [bend right] node {$1,0,R$} (q_e)
        (q_d) edge [loop right] node {$1,1,R$} (q_d)
          edge                node {$0,1,R$} (q_a)
        (q_e) edge [bend right] node {$1,0,R$} (q_a);
\end{tikzpicture}
```

LIBRARY: PETRI

A LIBRARY FOR DRAWING PETRI NETS



LIBRARIES: SHAPES

A SET OF LIBRARIES DEFINING NEW SHAPES

```
\node[draw,ellipse] {hello};
```

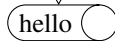
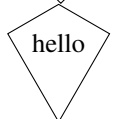
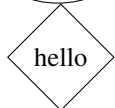
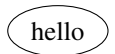
```
\node[draw,diamond] {hello};
```

```
\node[draw,kite] {hello};
```

```
\node[draw,cylinder] {hello};
```

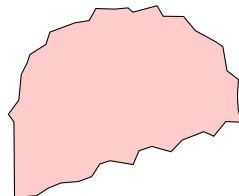
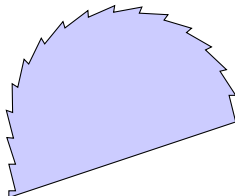
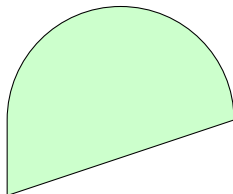
```
\node[draw,single arrow] {hello};
```

```
\node[draw,cloud callout] {hello};
```



LIBRARIES: DECORATIONS

LIBRARIES FOR “DECORATING” PATHS IN COMPLEX MANNERS.



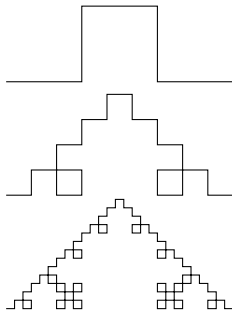
```
\begin{tikzpicture}
  \draw [fill=green!20]
    (0,0) -- (3,1) arc (0:180:1.5) -- cycle;

  \draw [fill=blue!20,xshift=3.5cm,
    decoration=saw]
    (0,0) -- (3,1) decorate { arc (0:180:1.5) -- cycle};

  \draw [fill=red!20,xshift=7cm,
    decoration={random steps,segment length=2mm}]
    decorate { (0,0) -- (3,1) arc (0:180:1.5)} -- cycle;
\end{tikzpicture}
```

LIBRARIES: DECORATIONS

LIBRARIES FOR “DECORATING” PATHS IN COMPLEX MANNERS.



```
\begin{tikzpicture}[decoration=Koch curve type 1]  
  \draw decorate{ (0,0) -- (3,0) };  
  \draw decorate{ decorate{ (0,-1.5) -- (3,-1.5) } };  
  \draw decorate{ decorate{ decorate{ (0,-3) -- (3,-3) } } };  
\end{tikzpicture}
```

SUMMARY

- ▶ TikZ provides a set of **T_EX macros** for creating figures directly inside T_EX.
- ▶ TikZ works with all **standard backend drivers and formats**.
- ▶ TikZ has a **powerful, consistent syntax**.
- ▶ TikZ is especially suited for **small or highly structured figures**.