LaTeX, beamer, tikz und Co.

14 Tikz: advanced topics

Thomas Worsch

Fakultät für Informatik
Karlsruher Institut für Technologie

Wintersemester 2016/2017

Überblick

Nodes

- Nodes: drei Konzepte auf einmal
- Knotenformen
- Anker
- Inhalt von Knoten
- Positionierung von Knoten „auf“ „Kanten“
- Optionen `remember picture` und `overlay`

Matrices

Edges

Automata

Überblick

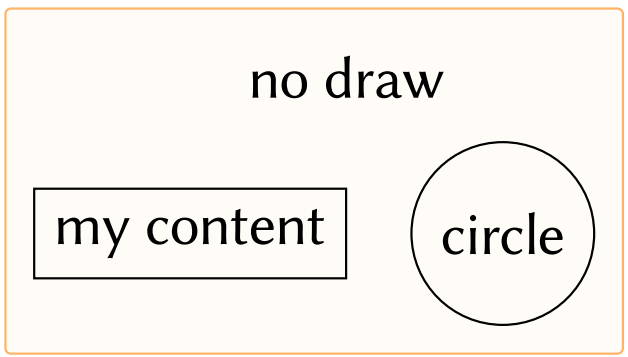
Die `node` Operation

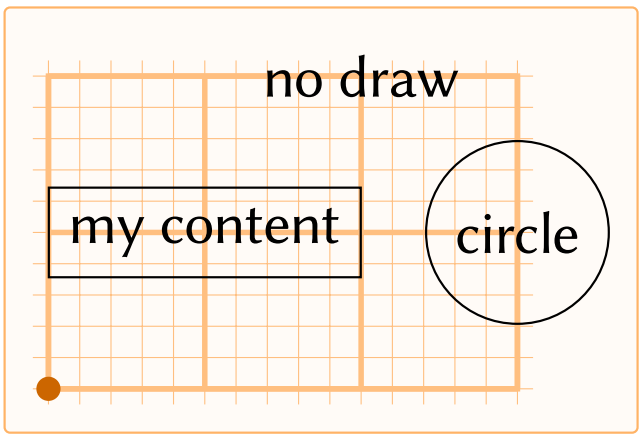
Ein Knoten

- erzeugt durch Pfadoperation `node`
- umfasst (meist) drei Konzepte:
  - eine äußere *Form* (z. B. Rechteck, Kreis, …)
  - einen *Inhalt* (Text)
  - einen symbolischen *Namen*
    - und mindestens einen *Anker*
- man muss nicht alle Konzepte gleichzeitig benutzen

Die node Operation: Beispiel

```
\begin{tikzpicture}
  \draw (2,2) node[rectangle] (myname) {no draw};
  \draw (1,1) node[draw,rectangle] (myname) {my content};
  \draw (3,1) node[draw,circle] (myname) {circle};
\end{tikzpicture}
```





Syntax

- Syntax (einfache Variante; es geht noch komplexer):
  `node` `[` ⟨*Optionen*⟩ `]` `(` ⟨*Name*⟩ `)` `{` ⟨*Inhalt*⟩ `}`
- beachte:
  - `[` ⟨*Optionen*⟩ `]` sind optional ;-)
  - `(` ⟨*Name*⟩ `)` darf auch vollständig fehlen
  - Inhalt darf leer sein,
    - *aber die geschweiften Klammern müssen stehen*
    - außer bei Knoten der Form `coordinate` („0-dimensional“): dort ist `{` ⟨*Inhalt*⟩ `}` gänzlich verboten
- In den Optionen kann man z. B. spezifizieren
  - die Form des Knotens
  - ob der Knoten(rahmen) gezeichnet, gefüllt, etc. werden soll
  - welcher *Anker* des Knotens als Referenzpunkt genommen werden soll

Überblick

Knotenformen

- ohne extra tikzlibrary:
  - `rectangle`, `coordinate`, `circle`
  - default, falls keine Angabe: `rectangle`
- `\usetikzlibrary{shapes.geometric}`:
  - `diamond`,`regular polygon`,`star`,...
- `\usetikzlibrary{shapes.symbols}`:
  - `cloud`, `signal`, `tape`
- `\usetikzlibrary{shapes.multipart}`:
  - `circle split`, `rectangle split`
  - (siehe auch `automata`)
- `\usetikzlibrary{shapes.arrows}`
- `\usetikzlibrary{shapes.misc}`

Überblick

Anker

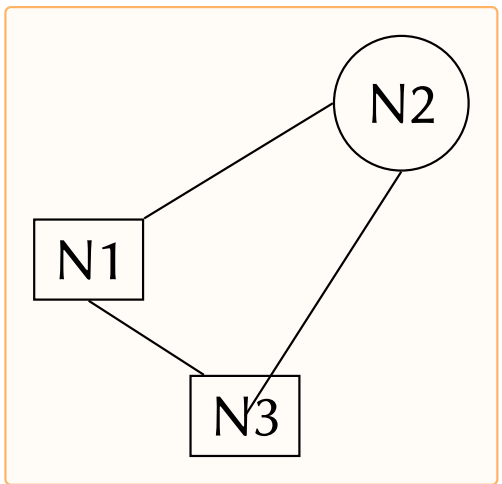
- der Referenzpunkt des Knotens, der an die angegeben Stelle gesetzt wird
  - Defaultwert `center` (existiert immer)
  - sonst spezifiziert mittels der `anchor=` Option
- für rechteckige Knoten:
  - symbolische Namen für einige Anker auf dem Rand: `north`, `north east`, …
  - `center` „in der Mitte“.
  - Gradangaben sind als Anker ebenfalls zulässig.
- für andere Knotenformen lese man die Doku
  - Abschnitt „Shape Library“

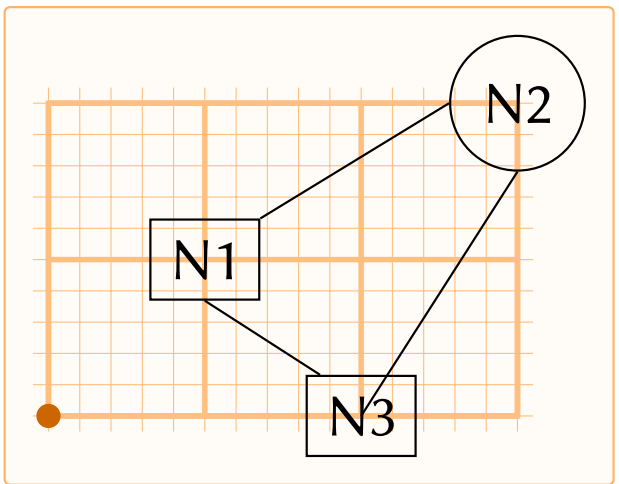
Symbolische Punktangaben

Punkte in einem Bild kann man auch so spezifizieren:

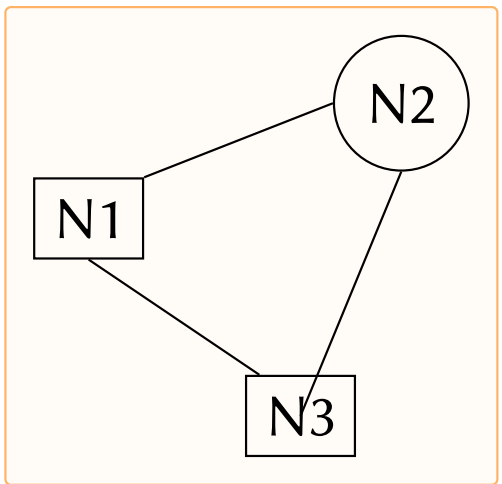
- (⟨*Knotenname*⟩.⟨*Anker*⟩) und
- (⟨*Knotenname*⟩)

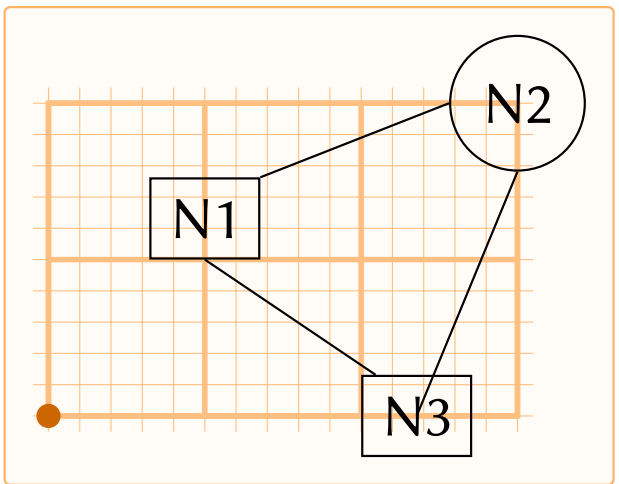
```
\begin{tikzpicture}
  \draw (1,1) node[draw]          (n1) {N1};
  \draw (3,2) node[draw,circle] (n2) {N2};
  \draw (2,0) node[draw]          (n3) {N3};

  \draw (n1.north east) -- (n2.west);
  \draw (n2.south) -- (n3.center);
  \draw (n3.135) -- (n1.270);
\end{tikzpicture}
```





```
\begin{tikzpicture}
  \draw (1,1) node[draw,anchor=south] (n1) {N1};
  \draw (3,2) node[draw,circle]       (n2) {N2};
  \draw (2,0) node[draw,anchor=west]  (n3) {N3};

  \draw (n1.north east) -- (n2.west);
  \draw (n2.south) -- (n3.center);
  \draw (n3.135) -- (n1.270);
\end{tikzpicture}
```



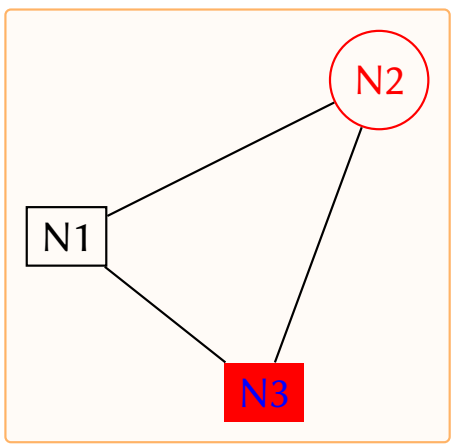


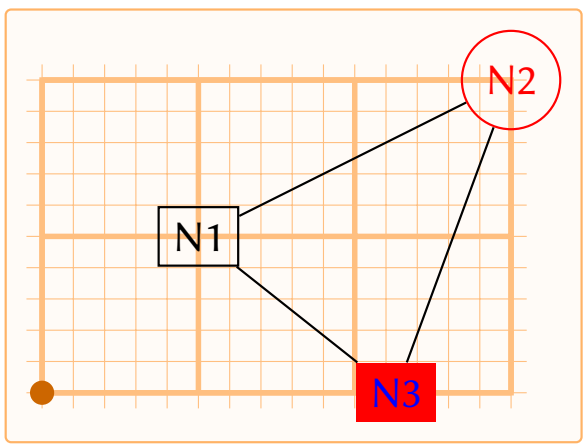
Besonderheiten

- Knotennamen *ohne* Ankerangabe als Koordinate:
  - *line to* Operationen sind «schlau» und enden *am Rand des Knotens*.
- angegebene ⟨*options*⟩ betreffen nur den Knoten, nichts «außen herum»
  - *manche* der Optionen von «außen» betreffen auch Knoten,
  - aber z. B. *nicht* `fill`, `draw`, Drehungen, …
- Knoten werden erst *ganz am Ende* gezeichnet, wenn alle *line to* Operationen etc. schon erledigt sind

Besonderheiten: Beispiel

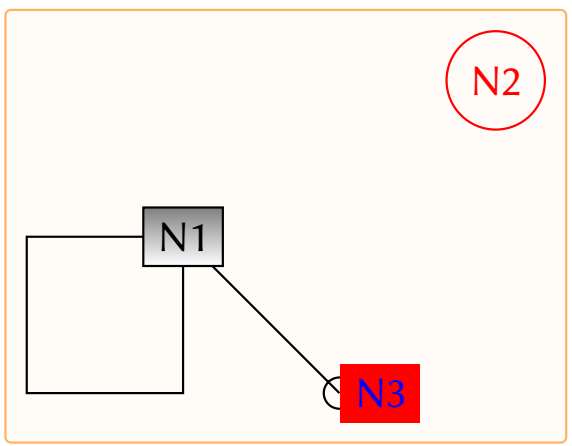
```
\begin{tikzpicture}
  \draw (1,1) node[draw]                                  (n1) {N1};
  \draw (3,2) node[draw,red,circle]                       (n2) {N2};
  \draw (2,0) node[fill=red,text=blue,anchor=west] (n3) {N3};

  \draw (n1) -- (n2);
  \draw (n2) -- (n3);
  \draw (n3) -- (n1);
\end{tikzpicture}
```

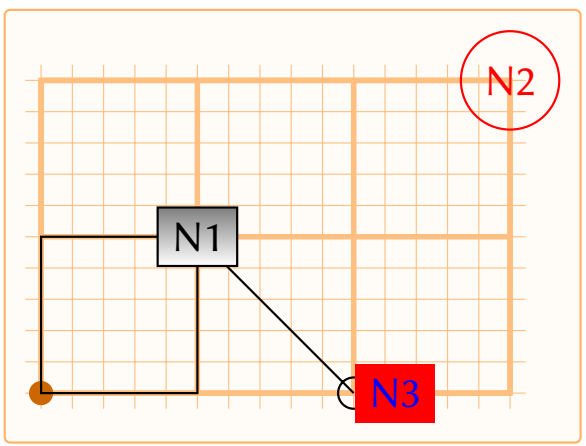




Besonderheiten: Beispiel (2)

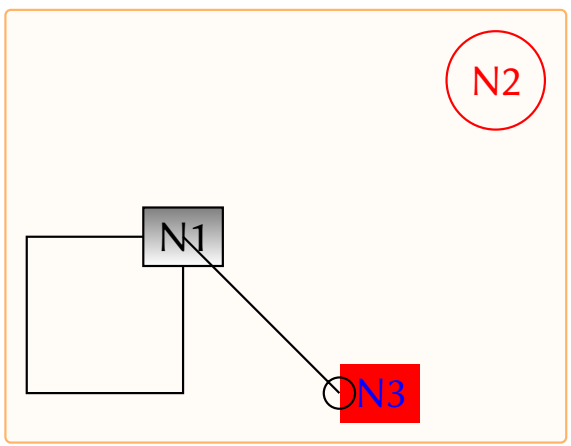
```
\begin{tikzpicture}
  \draw (1,1) node[draw,shade] (n1) {N1}
        (3,2) node[draw,red,circle] (n2) {N2}
        (2,0) node[fill=red,text=blue,anchor=west] (n3) {N3}
        (0,0) rectangle +(1,1)
        (1,1) -- (2,0) circle[radius=1mm];
\end{tikzpicture}
```

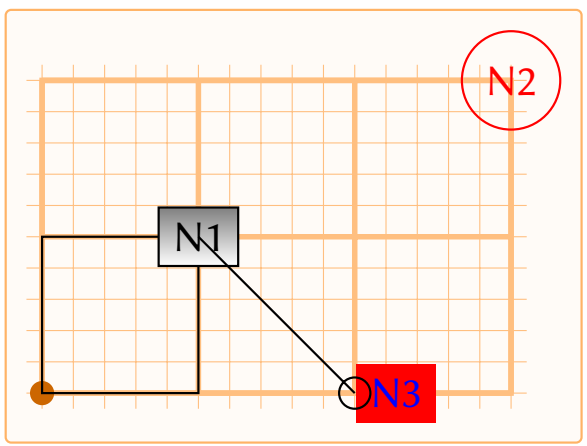




Besonderheiten: Beispiel (3)

```
\begin{tikzpicture}
  \draw (1,1) node[draw,shade] (n1) {N1}
        (3,2) node[draw,red,circle] (n2) {N2}
        (2,0) node[fill=red,text=blue,anchor=west] (n3) {N3}
        (0,0) rectangle +(1,1);
  \draw (1,1) -- (2,0) circle[radius=1mm];
\end{tikzpicture}
```





Überblick

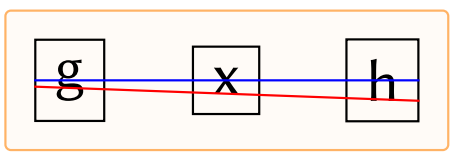
Textinhalt von Knoten

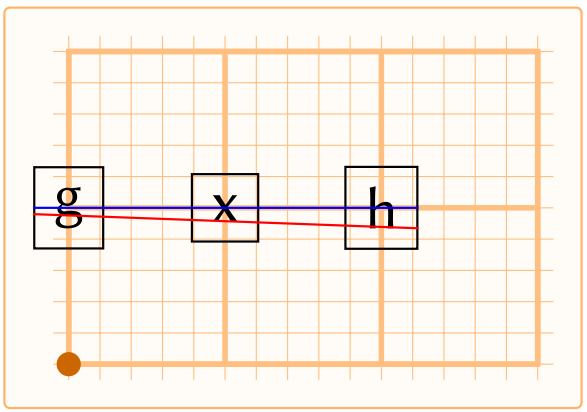
Der Inhalt eines Knotens kann gesetzt werden als

- einzeiliger Text: default (Inhalt in einer `\hbox`)
- mehrzeiliger Text
  - im Inhalt Umgebung, die mehrzeiligen Text „macht“
    - z. B. `tabular`
  - Knoten-Option `align=` verwenden und im Inhalt `\\`
    - `left`, `center`, `right`
  - Knoten-Option `text width=`
    - dann Inhalt automatisch umbrochen
    - `\\` auch erlaubt
    - so kann man die Breite natürlich auch für einzeilige Knoten setzen

Ausrichtung von Knoten untereinander: ein Problem

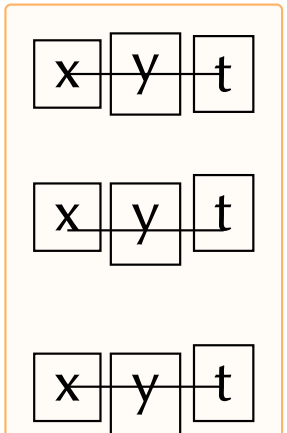
```
\begin{tikzpicture}
  \draw (0,1) node[draw] (n1) {g};
  \draw (1,1) node[draw] (n2) {x};
  \draw (2,1) node[draw] (n3) {h};

  \draw[blue] (n1.west)  -- (n3.east);
  \draw[red]  (n1.base west)  -- (n3.base east);
\end{tikzpicture}
```

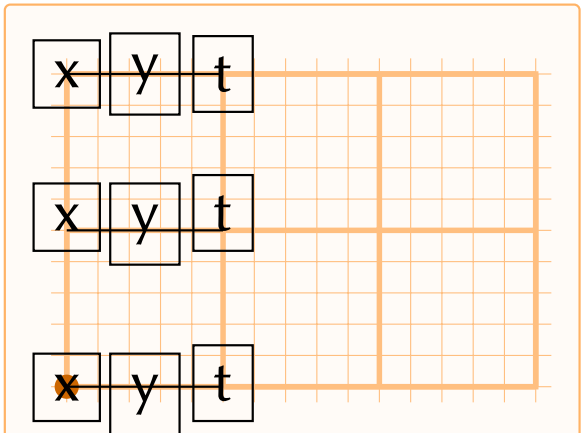




Ausrichtung von Knoten untereinander

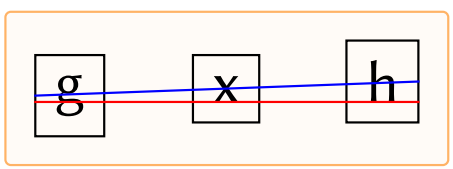
```
\begin{tikzpicture}[every node/.style={draw}]
  \draw[anchor=center] (0,2)
     node{x} -- ++(0.5,0) node{y} -- ++(0.5,0) node{t};
  \draw[anchor=base]   (0,1)
     node{x} -- ++(0.5,0) node{y} -- ++(0.5,0) node{t};
  \draw[anchor=mid]    (0,0)
     node{x} -- ++(0.5,0) node{y} -- ++(0.5,0) node{t};
\end{tikzpicture}
```

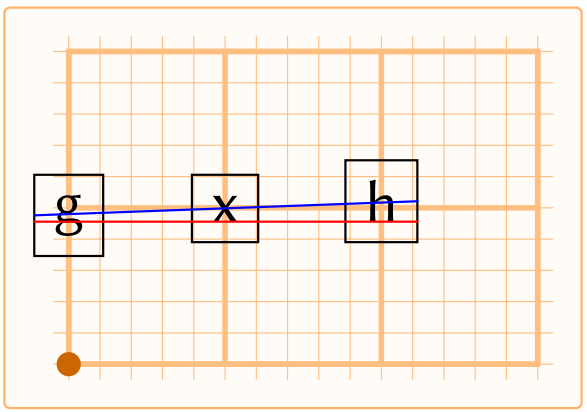




Ausrichtung von Knoten untereinander

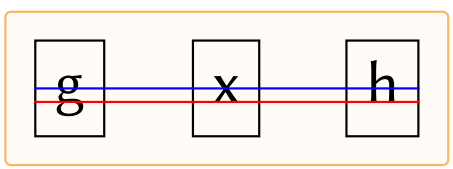
```
\begin{tikzpicture}
  \draw (0,1) node[draw,anchor=mid] (n1) {g};
  \draw (1,1) node[draw,anchor=mid] (n2) {x};
  \draw (2,1) node[draw,anchor=mid] (n3) {h};

  \draw[blue] (n1.west)  -- (n3.east);
  \draw[red]  (n1.base west)  -- (n3.base east);
\end{tikzpicture}
```

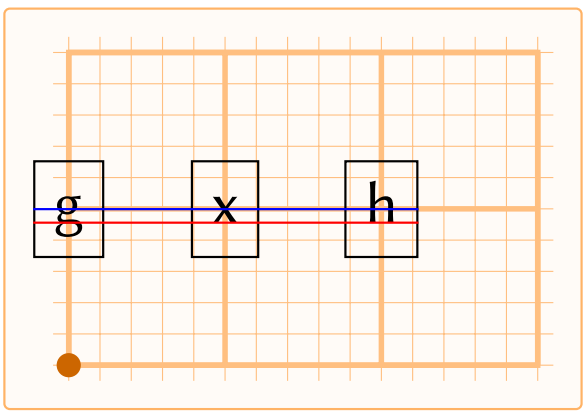




Höhe von Knoten (1)

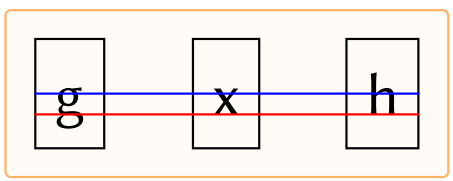
```
\def\foo{\vphantom{fy}}
\begin{tikzpicture}[anchor=mid]
  \draw (0,1) node[draw] (n1) {g\foo};
  \draw (1,1) node[draw] (n2) {x\foo};
  \draw (2,1) node[draw] (n3) {h\foo};
  \draw[blue] (n1.west)  -- (n3.east);
  \draw[red]  (n1.base west)  -- (n3.base east);
\end{tikzpicture}
```

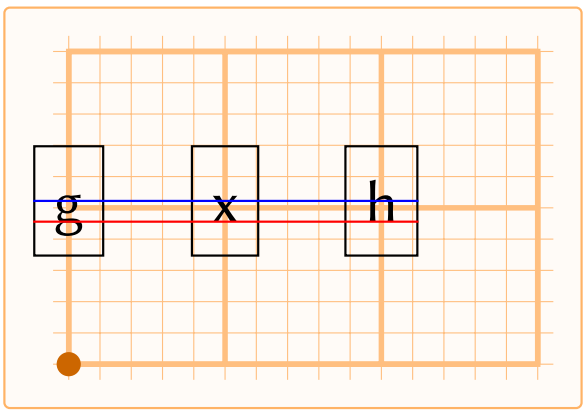




Höhe von Knoten (2)

```
\begin{tikzpicture}[every node/.style=
  {anchor=mid,text height=2ex, text depth=0.5ex}]
  \draw (0,1) node[draw] (n1) {g};
  \draw (1,1) node[draw] (n2) {x};
  \draw (2,1) node[draw] (n3) {h};
  \draw[blue] (n1.west)  -- (n3.east);
  \draw[red]  (n1.base west)  -- (n3.base east);
\end{tikzpicture}
```



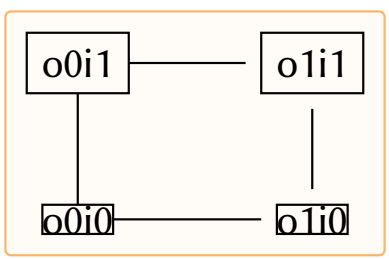


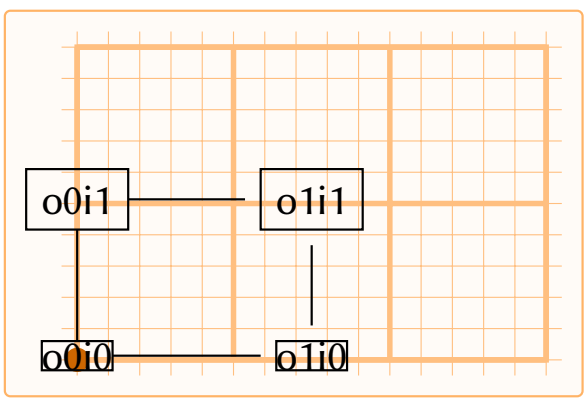
Abstände zwischen Inhalt, „Rahmen“ und Umgebung

- Abstand zwischen Inhalt und „Rahmen“
  - Option `inner sep`
  - bzw `inner xsep` und `inner ysep`
- Abstand zwischen „Rahmen“ und Umgebung
  - Option `outer sep`
  - bzw `outer xsep` und `outer ysep`
- weitere Optionen
  - `minimum height`
  - `minimum width`

„Knotenrahmen“

```
\begin{tikzpicture}[anchor=mid]
  \draw (0  ,0) node[draw,outer sep=0mm,inner sep=0mm] (n1) {o0i0};
  \draw (0  ,1) node[draw,outer sep=0mm,inner sep=1mm] (n2) {o0i1};
  \draw (1.5,0) node[draw,outer sep=1mm,inner sep=0mm] (n3) {o1i0};
  \draw (1.5,1) node[draw,outer sep=1mm,inner sep=1mm] (n4) {o1i1};
  \draw (n1)  -- (n2) -- (n4) -- (n3) -- (n1);
\end{tikzpicture}
```





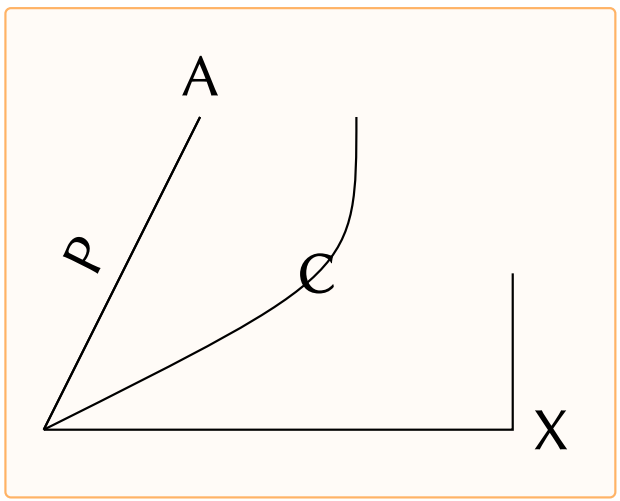
Überblick

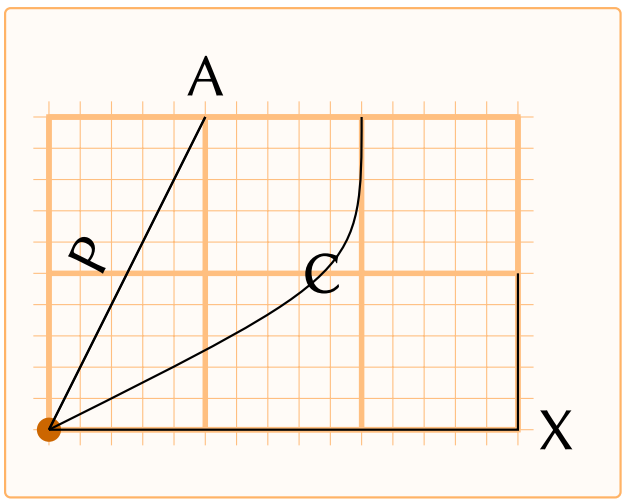
Explizite Knotenpositionierung auf „Kanten“

- bisher: Knoten dort, wo ein Punkt stehen könnte
- nun: irgendwo entlang des zuletzt gezeichneten Linien- oder Kurvenstücks
  - explizite Positionierung: `[pos=`⟨*rel.Pos.*⟩`]`
  - implizite Positionierung: Knotenangabe unmittelbar nach `--` oder `..`

Explizite Knotenpositionierung auf „Kanten“: Beispiele

```
\begin{tikzpicture}
  \draw (0,0) --  (1,2) node[above] {A};
  \draw (0,0) --  (1,2) node[sloped,above,pos=0.5] {P};
  \draw (0,0) -| (3,1) node[pos=0.5,right] {X};
  \draw (0,0) ..controls(2,1).. (2,2) node[pos=0.5] {C};
\end{tikzpicture}
```



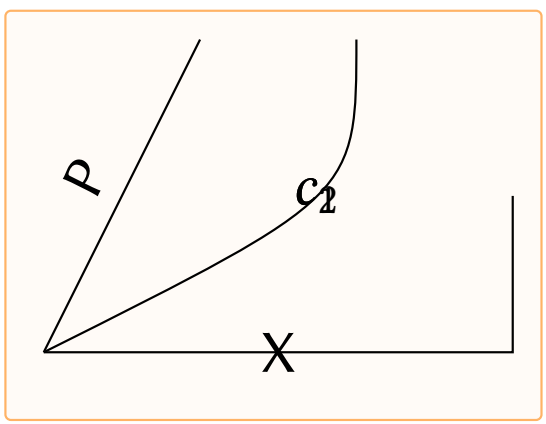


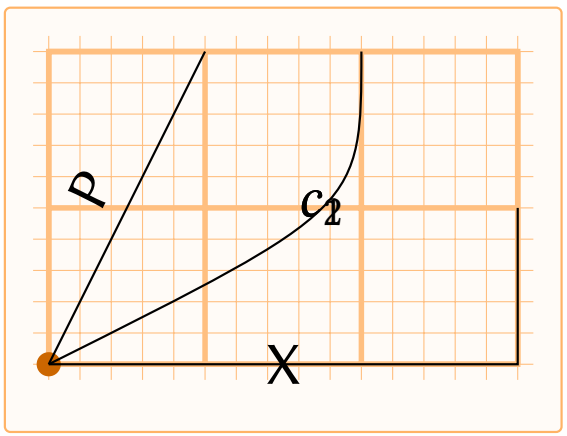
Implizite Knotenpositionierung auf „Kanten“

- bisher: Knoten dort, wo ein Punkt stehen könnte
- nun: Knoten nach „Linienangaben“

Implizite Knotenpositionierung auf „Kanten“: Beispiel

```
\begin{tikzpicture}
  \draw (0,0) -- node[sloped,above] {P} (1,2);
  \draw (0,0) .. node {$c_1$} controls(2,1)
              .. node {$c_2$} (2,2);
  \draw (0,0) -| node[pos=0.25] {X} (3,1);
\end{tikzpicture}
```





Knoten der „Form“ `coordinate`

- `coordinate[`⟨*options*⟩`](`⟨*name*⟩`)at(`⟨*point*⟩`)`
  - Optionen sind optional
  - `at(`⟨*point*⟩`)` ist optional (geht auch bei anderen Knoten)
- solche Knoten haben *keinen Inhalt*
  - `(`⟨*coordinate name*⟩`)` wird behandelt wie `(`⟨*coordinate name*⟩`.center)`
- äquivalent sind
  - `coordinate`
  - `node[shape=coordinate]`
- `\coordinate` ist Abkürzung für `\path coordinate`
- und übrigens `\node` ist Abkürzung für `\path node`

Überblick

remember picture und overlay

- remember picture für tikzpicture Umgebungen
  - sorgt dafür, dass Knotennamen und Positionen auch in anderen Bildern benutzt werden können
- overlay für (Teil-)Pfade
  - sorgt dafür, dass die Teile «*keinen Platz*» benötigen (in der bounding box)
- so auch Verwendung des «virtuellen Knotens» current page
- siehe demo-remember-overlay.tex

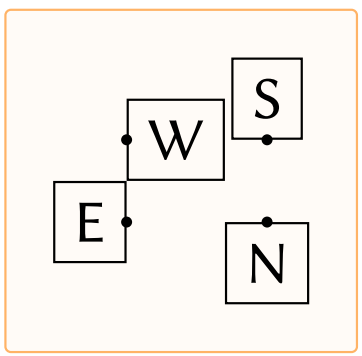
Überblick

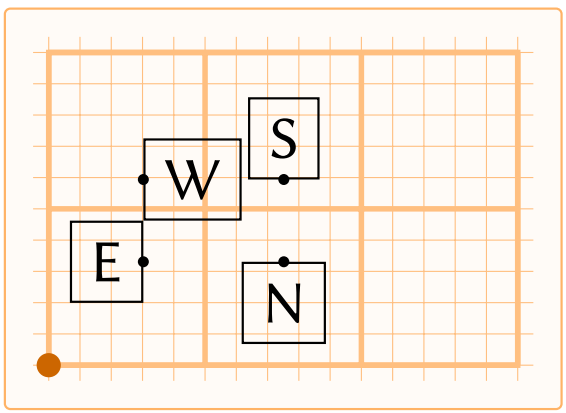
matrix

- zur Ausrichtung von Bildteilen
  - horizontal und vertikal
  - ähnlich `tabular` und `array`
- eine Matrix ist ein Knoten:
  - `\matrix` Abkürzung für `\path node[matrix]`
  - Grobsyntax: `\matrix [`⟨*Optionen*⟩`] (`⟨*Name*⟩`) {`⟨*Zeilen*⟩`};`
  - Zeilen
    - ⟨*Zelle*⟩ `&` ··· `&` ⟨*Zelle*⟩ `\\`
    - Achtung: auch am Ende der letzten Zeile muss `\\` stehen
    - dürfen unterschiedlich viele Einträge haben
  - Zelle: `tikz`-Kommandos für ein Zell-Bild
- Ausrichtung der Zell-Bilder: anhand ihrer Anker

matrix: Beispiel zur Ausrichtung

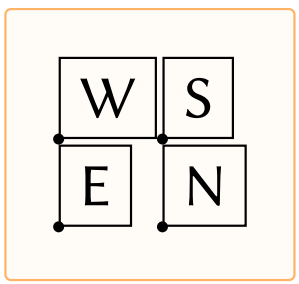
```
\begin{tikzpicture}
  \matrix[every node/.style={draw}, anchor=south west,
     execute at end cell={\fill (0,0) circle[radius=1pt];},
   ] {
    \node[anchor=west] {W}; & \node[anchor=south] {S}; \\
    \node[anchor=east] {E}; & \node[anchor=north] {N}; \\
  };
\end{tikzpicture}
```

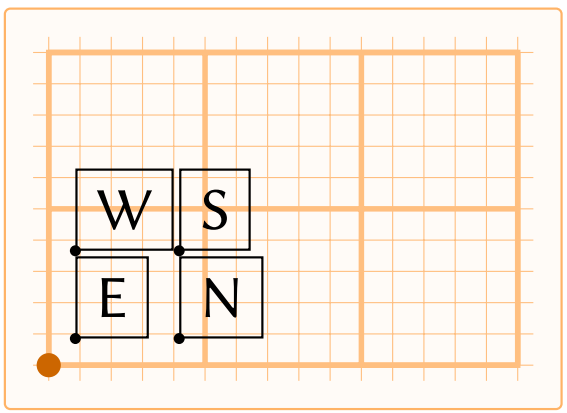




matrix: Beispiel zu Abständen

```
\begin{tikzpicture}
  \matrix[every node/.style={draw}, anchor=south west,
     execute at end cell={\fill (0,0) circle[radius=1pt];},
   ] {
    \node {W}; & \node {S}; \\
    \node {E}; & \node {N}; \\
  };
\end{tikzpicture}
```





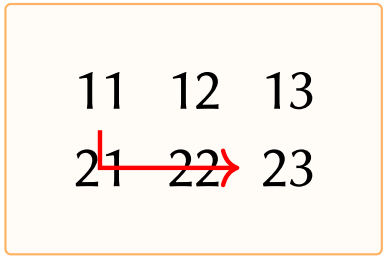
`\usetikzlibrary{matrix}`

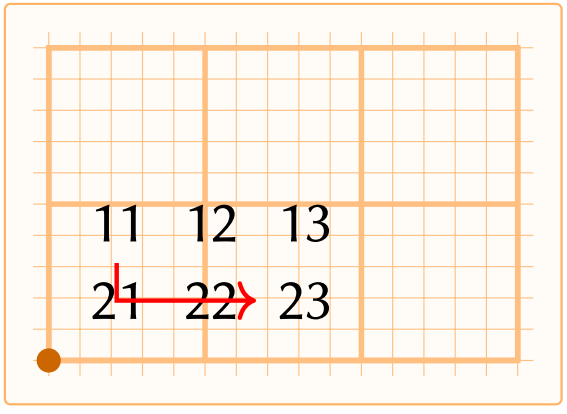
zusätzliche Abkürzungen

- Option `matrix of nodes`
- Option `matrix of math nodes`
- Option `left delimiter=`⟨*Klammer*⟩
- Option `right delimiter=`⟨*Klammer*⟩
- Option `above delimiter=`⟨*Klammer*⟩
- Option `below delimiter=`⟨*Klammer*⟩

matrix of nodes: Beispiel für Zellnamen

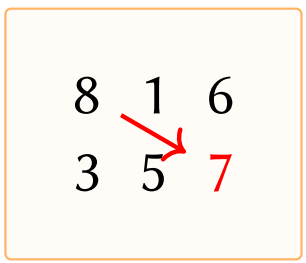
```
\begin{tikzpicture}
  \matrix (M) [matrix of nodes,anchor=south west] {
    11 & 12 & 13  \\
    21 & 22 & 23  \\
  };
  \draw[thick,red,->] (M-1-1) |- (M-2-3);
\end{tikzpicture}
```

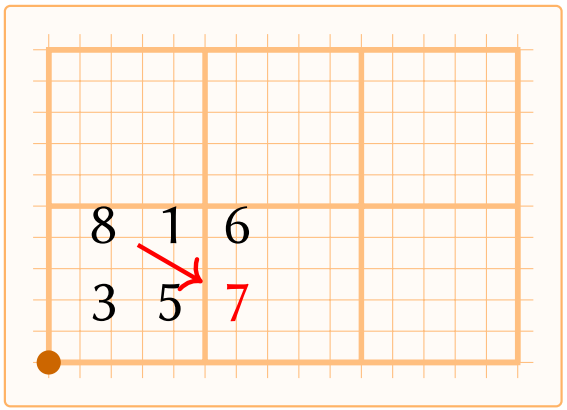




matrix of nodes: Beispiel für Knotenoptionen

```
\begin{tikzpicture}
  \matrix (M) [matrix of nodes,anchor=south west] {
    8 & 1 &               6 \\
    3 & 5 & |[red] (S)| 7 \\
  };
  \draw[thick,red,->] (M-1-1) -- (S);
\end{tikzpicture}
```





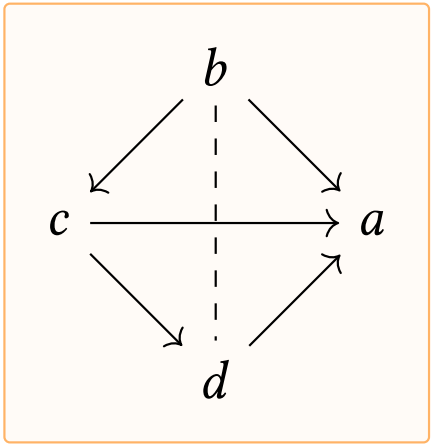
Überblick

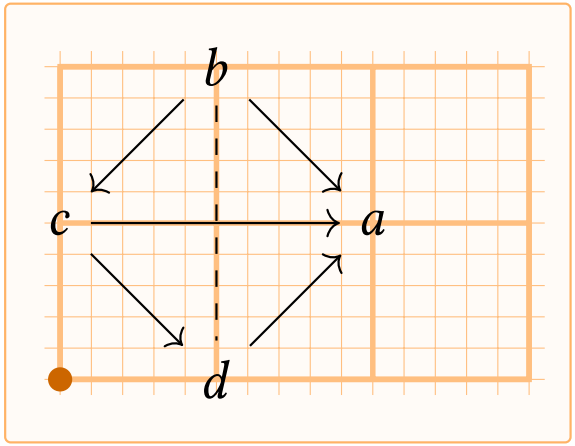
Edges

- Syntax: `edge (`⟨*Punkt*⟩`)` oder allgemeiner
  `edge [`⟨*Optionen*⟩`]` ⟨*Knoten*⟩ `(`⟨*Punkt*⟩`)`
- erzeugt ein Linienstück (wie Pfadoperation `to`)
- aber:
  - das Linienstück wird erst gezeichnet, nachdem der „Hauptpfad“ fertig ist
  - für das Linienstück wird ein neuer Pfad konstruiert
    - kann also z. B. eine andere Farbe haben
  - der „Hauptpfad“ wird nicht verändert, d. h. insbesondere:
  - nach Ende der `edge` ist der Startpunkt für die nächste Pfadoperation der gleiche wie vorher

Edges: Beispiel aus `tikz` Doku

```
\begin{tikzpicture}
  \foreach \nam/\ang in {a/0,b/90,c/180,d/270}
    { \node[shift={(1,1)}] (\nam) at (\ang:1) {$\nam$}; }
  \path[->] (b) edge (a)
                edge (c)
                edge [-,dashed] (d)
            (c) edge (a)  edge (d)
            (d) edge (a)  ;
\end{tikzpicture}
```



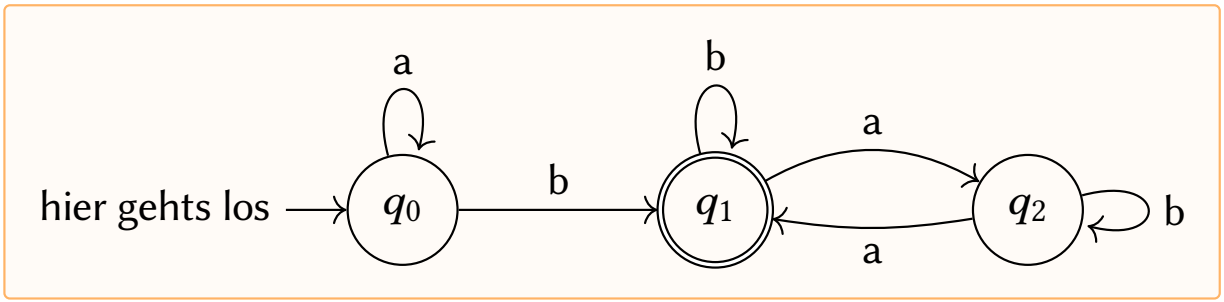


Überblick

- `\usetikzlibrary{automata}`
- bequeme(re) Positionierung von Zuständen (nodes)

automata: Beispiel (ähnlich tikz Doku)

```
\begin{tikzpicture}[node distance=2cm,auto]
  \node[state,initial,initial text=hier gehts los] (q0) {$q_0$};
  \node[state,accepting] (q1) [right of=q0] {$q_1$};
  \node[state] (q2) [right of=q1] {$q_2$};

  \path[->] (q0) edge [loop above]  node {a} ()
                 edge                node {b} (q1)
            (q1) edge [loop above]  node {b} ()
                 edge [bend left]   node {a} (q2)
            (q2) edge [loop right]  node {b} ()
                 edge [bend left=9] node {a} (q1)
  ;
\end{tikzpicture}
```



Überblick

es gibt viele Möglichkeiten …

… für so etwas wie „Plots“ von Funktionen bzw. Datenbeständen

- im folgenden nur die einfachste Variante
- siehe auch:
  - LaTeX-Paket `pgfplots`
    - Diagramme mit Achsenbeschriftungen, …
  - `\datavisualization` in `tikz` Version 3.0.0

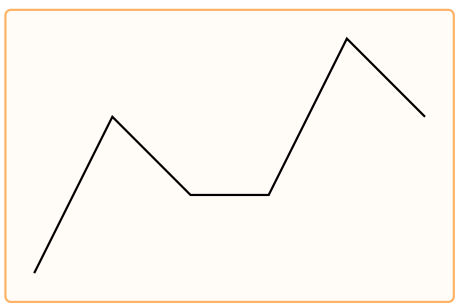
Verschiedene Arten von Plots

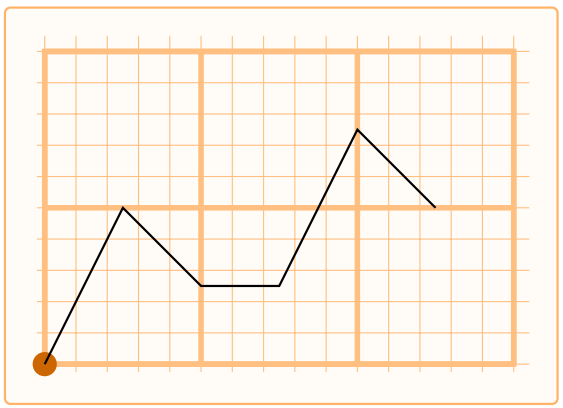
- verschiedene Arten von „Eingabe“
  - explizite Liste von Koordinaten
  - Liste von Koordinaten aus einer Datei
  - Formeln a la „$y = x^2$“
- verschiedene Arten von „Ausgaben“
  - Geradenstücke
  - geglätte Kurve
  - Treppenfunktionen
  - Balkendiagramme
- Syntax im wesentlichen:
  ⟨*opt.* `--`⟩ `plot [` ⟨*Ausgabeart*⟩ `]` ⟨*Eingabeart*⟩ ⟨*Eingabe*⟩

Plots expliziter Punkte: Geradenstücke

- Syntax: `plot coordinates { ⟨Punkte⟩ }`

```
\begin{tikzpicture}[x={(5mm,0mm)},y={(0mm,5mm)}]
  \draw (0,0) plot coordinates
     {(0,0) (1,2) (2,1) (3,1) (4,3) (5,2)};
\end{tikzpicture}
```

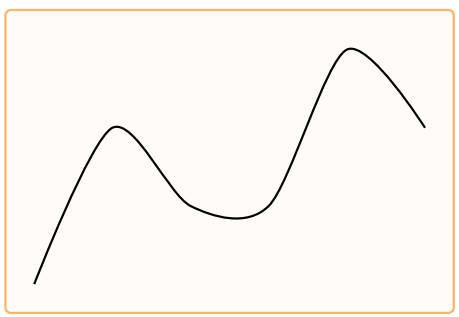


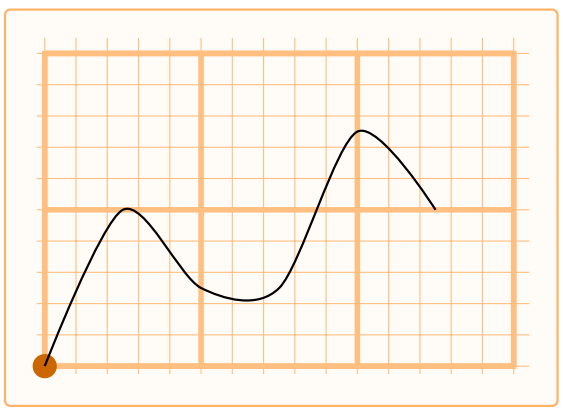


Plots expliziter Punkte: geglättete Kurve

- Syntax: `plot [ smooth ] coordinates { ⟨Punkte⟩ }`

```
\begin{tikzpicture}[x={(5mm,0mm)},y={(0mm,5mm)}]
  \draw (0,0) plot [smooth] coordinates
     {(0,0) (1,2) (2,1) (3,1) (4,3) (5,2)};
\end{tikzpicture}
```

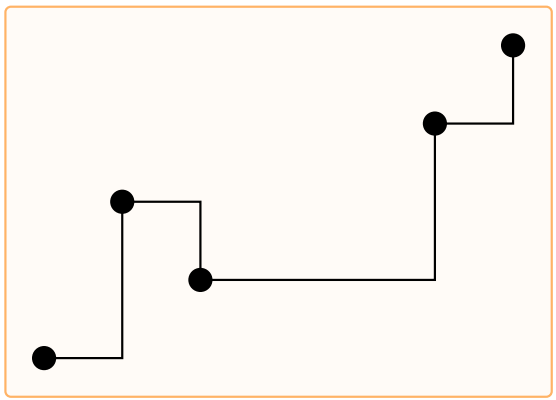


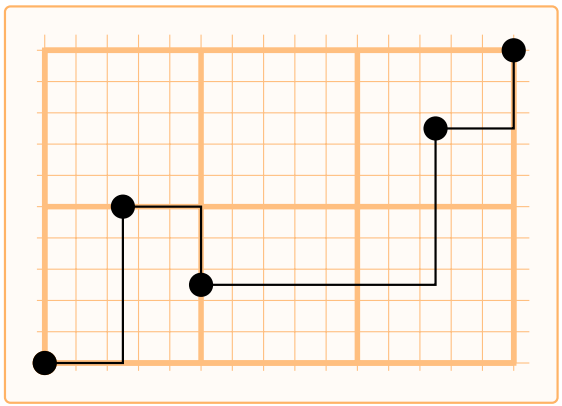


Plots expliziter Punkte: Treppenfunktionen

- Syntax: `plot [ const plot ] coordinates { ⟨Punkte⟩ }`
- statt `const plot` auch möglich
  - `const plot mark left` (Alias für `const plot`)
  - `const plot mark right`

```
\begin{tikzpicture}[x={(5mm,0mm)},y={(0mm,5mm)}]
  \draw (0,0) plot [const plot,mark=*] coordinates
    {(0,0) (1,2) (2,1) (5,3) (6,4)};
\end{tikzpicture}
```

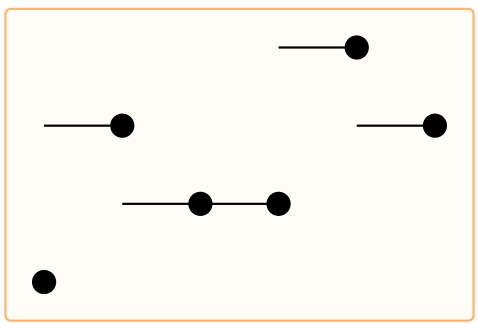


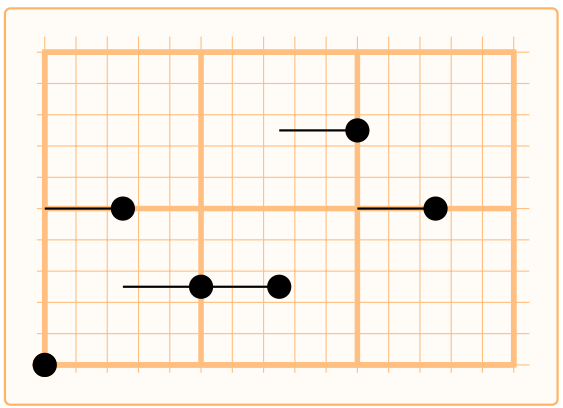


Plots expliziter Punkte: Treppenfunktionen (2)

- Syntax der Pfadoperation:
  `plot [ jump mark left  ] coordinates { ⟨Punkte⟩ }`
  `plot [ jump mark right ] coordinates { ⟨Punkte⟩ }`

```
\begin{tikzpicture}[x={(5mm,0mm)},y={(0mm,5mm)}]
  \draw (0,0) plot [jump mark right,mark=*] coordinates
    {(0,0) (1,2) (2,1) (3,1) (4,3) (5,2)};
\end{tikzpicture}
```

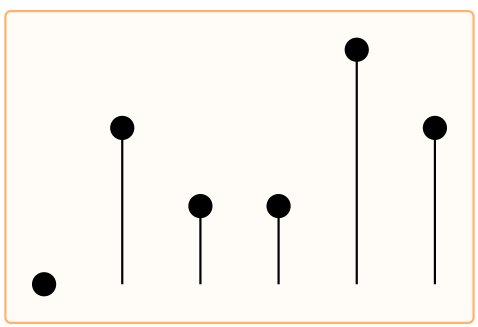


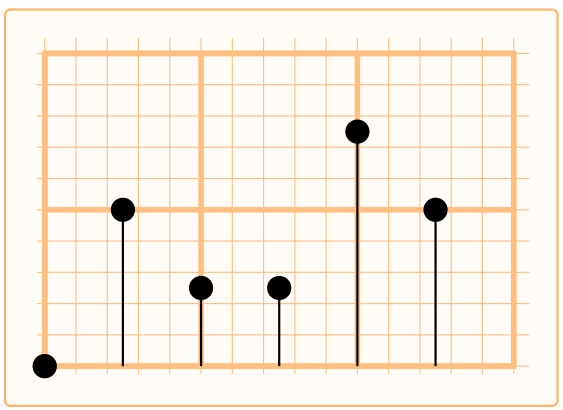


Plots expliziter Punkte: Balkendiagramme

- Syntax: `plot [ ycomb ] coordinates { ⟨Punkte⟩ }`

```
\begin{tikzpicture}[x={(5mm,0mm)},y={(0mm,5mm)}]
  \draw (0,0) plot [ycomb,mark=*] coordinates
     {(0,0) (1,2) (2,1) (3,1) (4,3) (5,2)};
\end{tikzpicture}
```

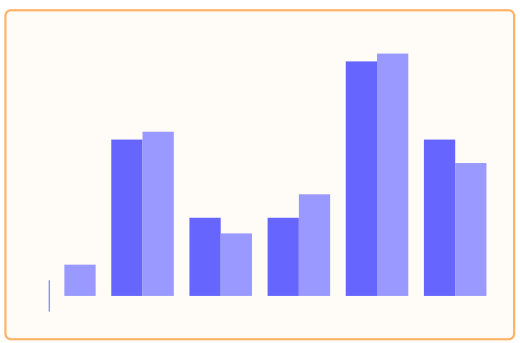


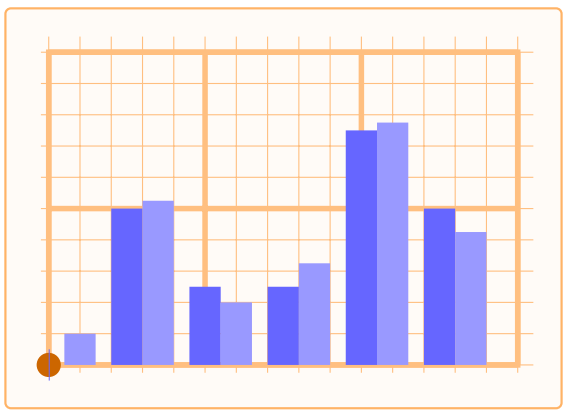


Plots expliziter Punkte: Balkendiagramme Beispiel

- Syntax: `plot [ ycomb ] coordinates { ⟨Punkte⟩ }`

```
\begin{tikzpicture}[x={(5mm,0mm)},y={(0mm,5mm)}]
  \draw[line width=2mm,blue!60] (0,0) plot [ycomb]
     coordinates {(0,0) (1,2) (2,1) (3,1) (4,3) (5,2)};
  \draw[line width=2mm,blue!40,xshift=2mm] (0,0) plot [ycomb]
     coordinates {(0,0.4) (1,2.1) (2,0.8) (3,1.3) (4,3.1) (5,1
\end{tikzpicture}
```





Weiteres zu Balkendiagrammen

- es gibt auch `xcomb` und `polar comb`
- `ybar`, `xbar`: statt Strichen werden Rechtecke gezeichnet
  - füllbar
  - verschiedene Farben für `draw` und `fill`

Eingabeart Datei

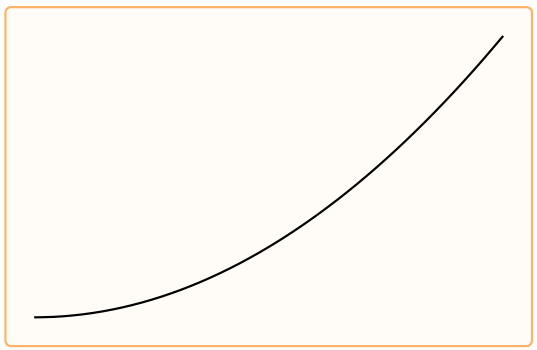
- Syntax `plot [` ⟨*Optionen*⟩ `] file {` ⟨*Dateiname*⟩ `}`
- Syntax der Eingabedatei
  - vollständig ignoriert werden Zeilen
    - die leer sind
    - die mit `%` beginnen
    - die mit `#` beginnen
  - die anderen
    - müssen mit zwei Zahlen beginnen
    - durch Leerzeichen getrennt
    - der Rest wird ignoriert

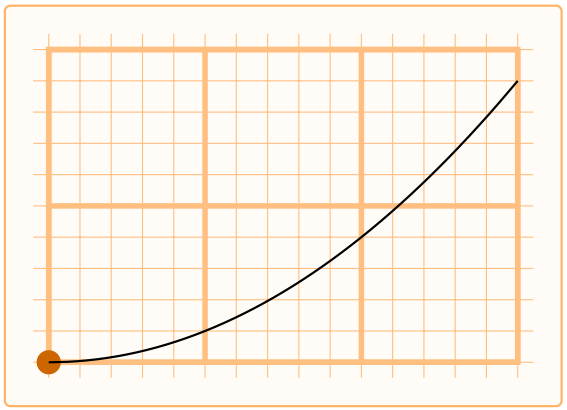
Eingabeart Funktionsausdruck

- Syntax `plot [` ⟨*Optionen*⟩ `]` ⟨*Funktionsausdruck*⟩
- in Optionen:
  - Spezifikation der Samples
    - `domain=` ⟨*Start*⟩ `:` ⟨*End*⟩
    - `samples=` ⟨*Anzahl*⟩ oder
    - `samples at={` ⟨*Liste*⟩ `}`
  - Name der Variablen (default `\x`)

Eingabeart Funktionsausdruck: Beispiel

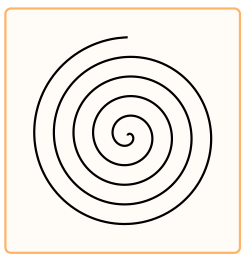
```
\begin{tikzpicture}
  \draw plot [smooth,domain=0:3] (\x,0.2*\x^2);
\end{tikzpicture}
```

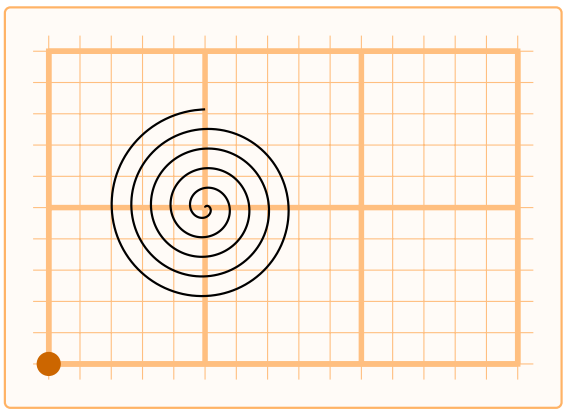




Eingabeart Funktionsausdruck: Beispiel (2)

```
\begin{tikzpicture}
  \draw[shift={(1,1)}]
    plot [smooth,variable=\t,domain=0:31.42,samples=300]
         ({0.02*\t*sin(\t r)},{0.02*\t*cos(\t r)});
\end{tikzpicture}
```



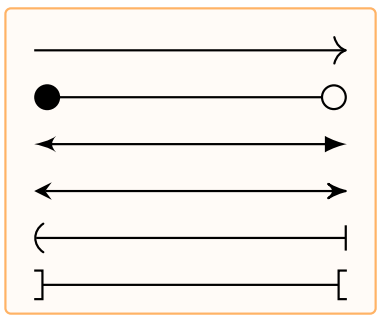


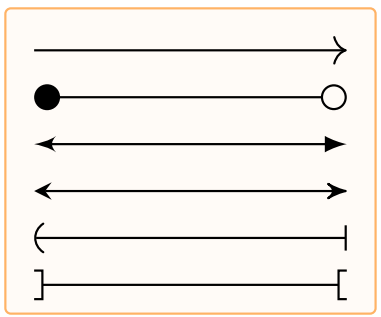
Überblick

Pfeile — alte Bibliothek

\usetikzlibrary{arrows}

```
\begin{tikzpicture}[pure,y={(0cm,-3mm)}]
  \draw[->]                    (0,0) -- ++(2,0);
  \draw[*-o]                   (0,1) -- ++(2,0);
  \draw[latex'-latex]          (0,2) -- ++(2,0);
  \draw[stealth-stealth']      (0,3) -- ++(2,0);
  \draw[(-|]                   (0,4) -- ++(2,0);
  \draw[arrows={]-[}]          (0,5) -- ++(2,0);
\end{tikzpicture}
```

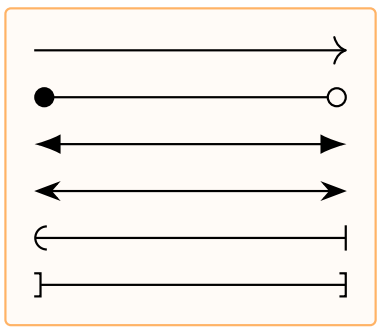


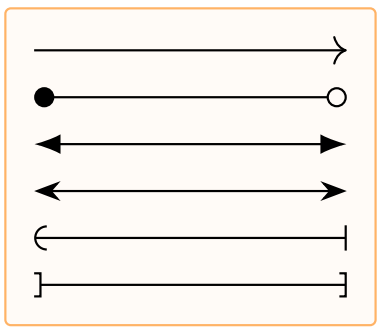


Pfeile — neue Bibliothek

\usetikzlibrary{arrows.meta}

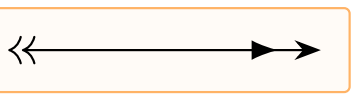
```
\begin{tikzpicture}[pure,y={(0cm,-3mm)}]
  \draw[->]                                (0,0) -- ++(2,0);
  \draw[{Circle}-{Circle[open]}]           (0,1) -- ++(2,0);
  \draw[{Latex[]}-{Latex[]}]               (0,2) -- ++(2,0);
  \draw[{Stealth[]}-{Stealth[]}]           (0,3) -- ++(2,0);
  \draw[{Arc Barb[]}-{Bar[]}]              (0,4) -- ++(2,0);
  \draw[{Bracket[reversed]}-{Bracket[]}] (0,5) -- ++(2,0);
\end{tikzpicture}
```

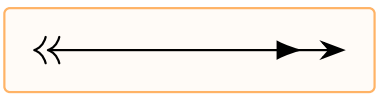




Mehrere Pfeilspitzen hintereinander

```
\begin{tikzpicture}[pure,y={(0cm,-3mm)}]
  \draw[arrows={<.<-{Latex[sep=3pt]Stealth[]}}]
      (0,0) -- ++(2,0);
\end{tikzpicture}
```





Überblick

tikzlibrary background

- `\usetikzlibrary{backgrounds}`
- Entkopplung von
  - Reihenfolge der Notation und
  - Reihenfolge der Zeichnung
- Benutzungsmöglichkeiten:
  - `\begin{tikzpicture}[show background rectangle]`
    …
  - `\begin{tikzpicture}[show background grid]`
    …
  - innerhalb eines `tikzpicture`:
    `\begin{scope}[on background layer]`
    …
    `\end{scope}`

tikzlibrary fit

- `\usetikzlibrary{fit}`
- Benutzung als Option einer `node` Pfadoperation:
  … `node [fit=`⟨*Liste*⟩`]` …wobei
  - ⟨*Liste*⟩ eine Liste von Knoten ist,
  - die durch Leerzeichen getrennt sind
  - laut Doku sind auch Punkte erlaubt; in der Realität
    - *muss dann die gesamte Liste in geschweifte Klammern gesetzt werden*
      (siehe Beispiel in `fit.tex`)
- erzeugt wird ein Knoten, der gerade groß genug ist, um
  - die `north`, `east`, `south` und `west` Anker
    aller aufgeführten Knoten
  - und alle aufgeführten Punkte

  zu umschließen.

fit: Beispiel (aus der tikz Doku)

```
\begin{tikzpicture}[inner sep=0pt,thick,
  dot/.style={fill=blue,circle,minimum size=3pt}]
  \node[dot] (a) at (1,1) {};
  \node[dot] (b) at (2,2) {};
  \node[dot] (c) at (1,2) {};
  \node[dot] (d) at (1.25,0.25) {};
  \node[dot] (e) at (1.75,1.5) {};
  \node[draw=red,   fit=(a) (b) (c) (d) (e)] {box};
  \node[draw,circle,fit=(a) (b) (c) (d) (e),fill=blue!9,opacity=0.8] {};
\end{tikzpicture}
```