UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR
THEORETISCHE INFORMATIK

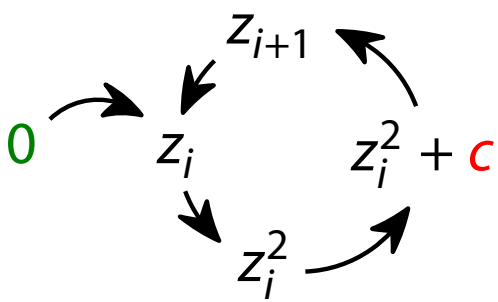
Kleine Ti*k*Z-Wunder

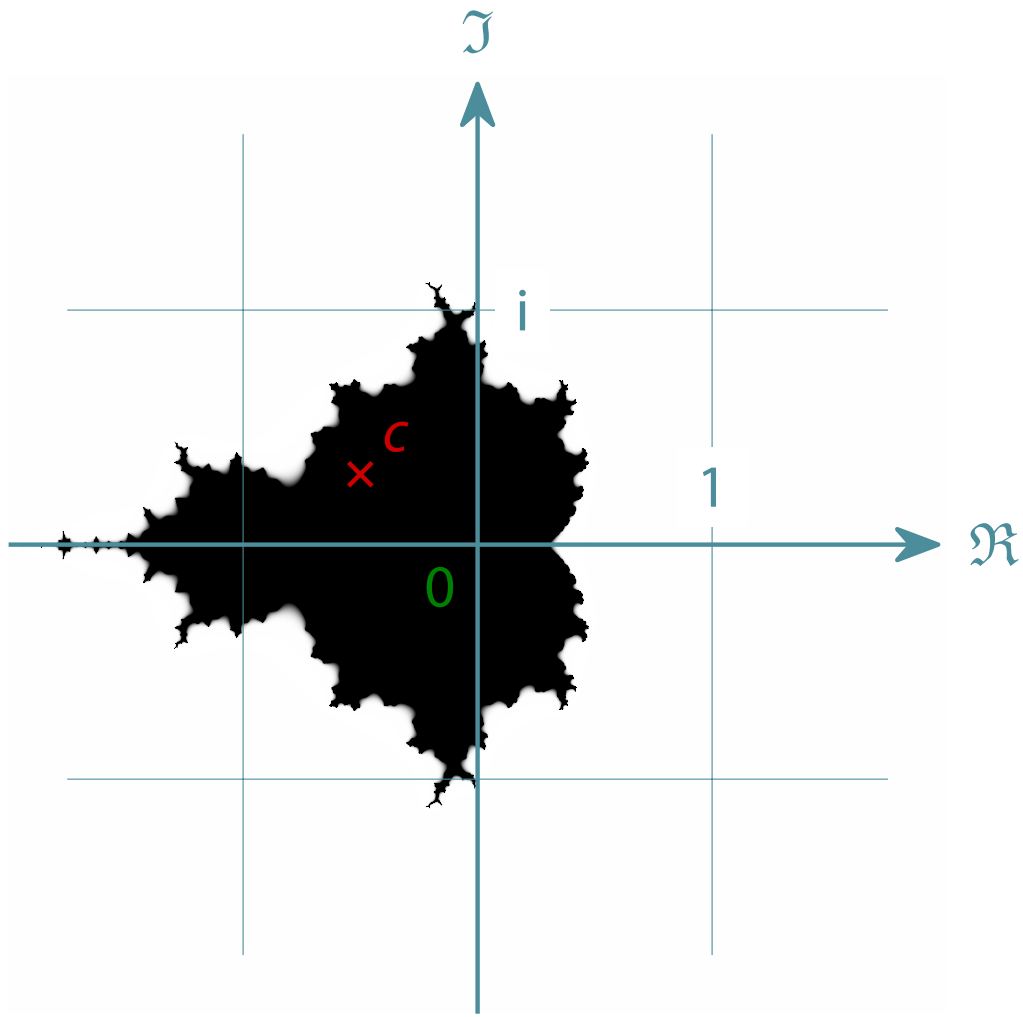
Till Tantau

DANTE-Frühjahrstagung 2015

IM FOCUS DAS LEBEN

Ein Bild voller kleiner Wunder.





Gliederung

Das Wunder der Geburt

Das Wunder der Graphen

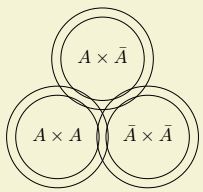
Das Wunder der Pfeile

Das Wunder der Schatten

2003: Wie alles begann.

On Structural Similarities of
Finite Automata and Turing Machine
Enumerability Classes

Till Tantau



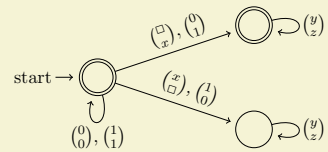
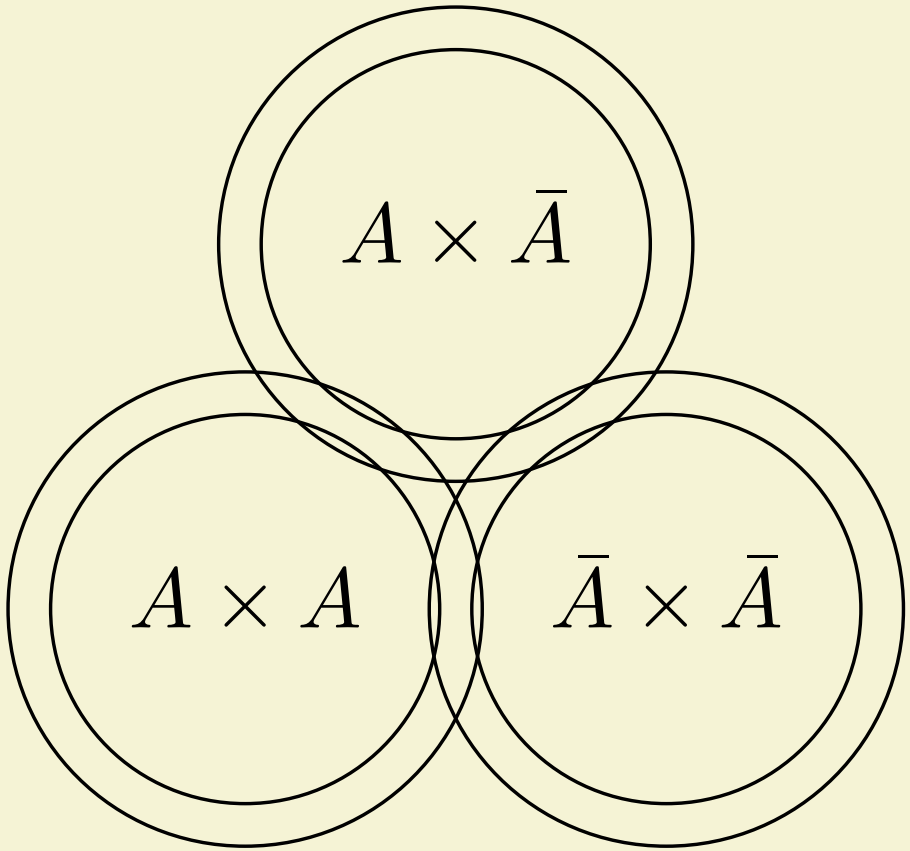


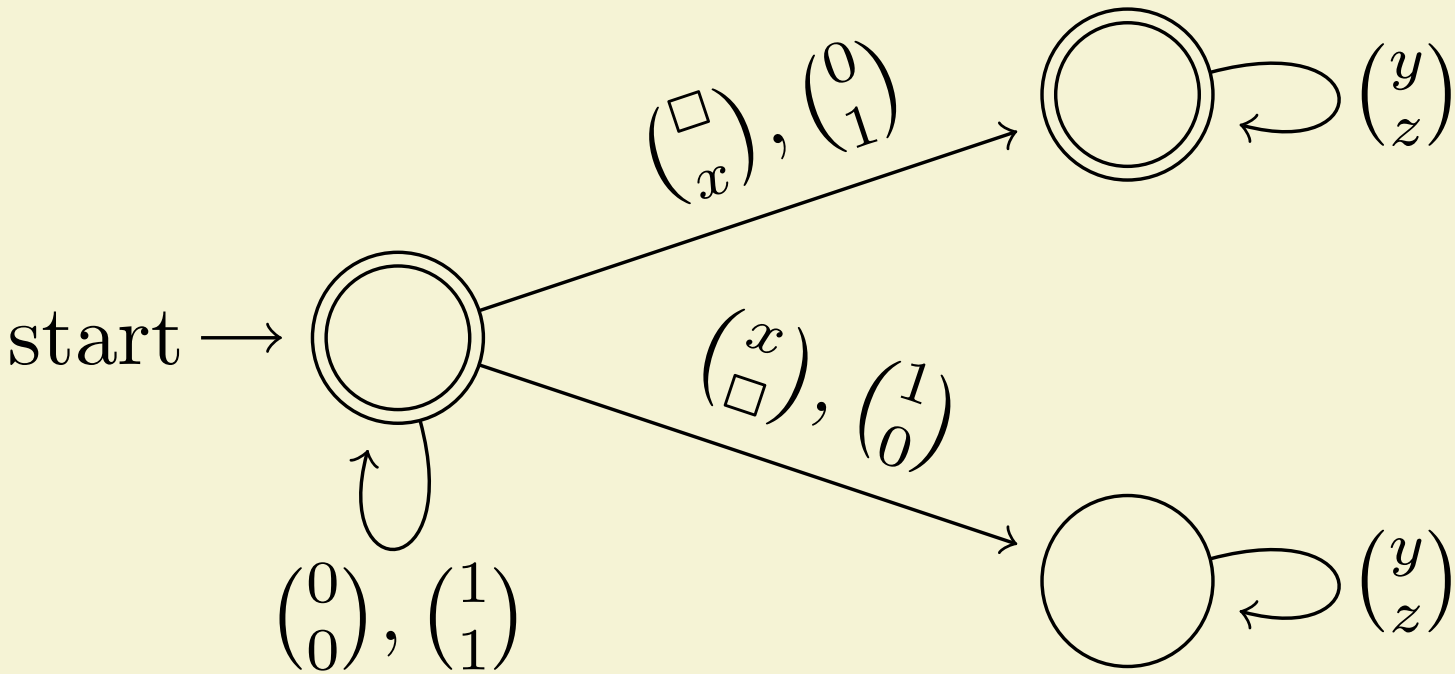
Figure 2-1
Given two words $u, v \in \{0,1\}^*$, the DFA accepts the coded word $\langle u, v \rangle$ iff $u \leq_{\text{lex}} v$. The variables $x \in \{0,1\}$ and $\binom{y}{z} \in \{0,1\}^{\langle 2 \rangle}$ represent arbitrary values. Double circles around states indicate accepting states.

36

2003: Wie alles begann.



2003: Wie alles begann.



2003: Das allererste »Ti*k*Z«-Bild.

```
\begin{pgfpicture}{-1.9cm}{-1.5cm}{1.9cm}{2cm}
  \pgfcircle[stroke]{\pgfpolar{90}{1cm}}{0.975cm}
  \pgfcircle[stroke]{\pgfpolar{210}{1cm}}{0.975cm}
  \pgfcircle[stroke]{\pgfpolar{330}{1cm}}{0.975cm}
  \pgfcircle[stroke]{\pgfpolar{90}{1cm}}{0.8cm}
  \pgfcircle[stroke]{\pgfpolar{210}{1cm}}{0.8cm}
  \pgfcircle[stroke]{\pgfpolar{330}{1cm}}{0.8cm}

  \pgfputat{\pgfrelative{\pgfpolar{90}{1cm}}%
    {\pgfpoint{0pt}{-.5ex}}}%
    {\pgfbox[center,base]{$A\times \bar A$}}
  \pgfputat{\pgfrelative{\pgfpolar{210}{1cm}}%
    {\pgfpoint{0pt}{-.5ex}}}%
    {\pgfbox[center,base]{$A\times A$}}
  \pgfputat{\pgfrelative{\pgfpolar{330}{1cm}}%
    {\pgfpoint{0pt}{-.5ex}}}%
    {\pgfbox[center,base]{$\bar A\times \bar A$}}
\end{pgfpicture}
```

2004: Die Version 0.62 von Ti*k*Z.
Die Liste der Dateien

```
AUTHORS                  pgf-tu-logo.jpg
ChangeLog                pgf-tu-logo.mask.jpg
FILES                    pgf.sty
INSTALL                  pgfarrows.sty
README                   pgfautomata.sty
TODO                     pgfheaps.sty
pgf-apple.jpg            pgfnodes.sty
pgf-apple.mask.jpg       pgfshade.sty
pgf-tu-logo.25.eps       pgfuserguide.pdf
pgf-tu-logo.25.jpg       pgfuserguide.tex
pgf-tu-logo.eps          xxcolor.sty
```

(Die aktuelle Ti*k*Z-Version auf diesem Rechner hat 4080 Dateien…)

2004: Die Version 0.62 von TikZ.
Das Handbuch

User's Guide to the PGF Package, Version 0.62
http://www.ctan.org/tex-archive/graphics/pgf/

Till Tantau
tantau@cs.tu-berlin.de

July 6, 2004

Contents

1 Introduction **1**
1.1 Overview . . . 1
1.2 Installation . . . 2
1.3 Installing Prebundled Packages . . . 2
1.3.1 Temporary Installation . . . 2
1.3.2 Installation in a texmf Tree . . . 3
1.4 Quick Start . . . 3
1.5 Gallery . . . 5

2 Basic Graphic Drawing **7**
2.1 Main Environments . . . 7
2.2 How to Specify a Point . . . 7
2.3 Coordinate Systems . . . 9
2.4 Path Construction . . . 10
2.5 Stroking and Filling . . . 11
2.6 Clipping . . . 13
2.7 Shape and Line Drawing . . . 14
2.8 Image Inclusion . . . 15
2.9 Text Drawing . . . 18
2.10 Drawing Arrows at Line Ends . . . 18
2.11 Placing Labels on Lines . . . 20
2.12 Shadings . . . 21

3 Using Nodes **22**
3.1 Node Creation . . . 23
3.2 Coordinates Relative to Nodes . . . 24
3.3 Connecting Nodes . . . 24
3.4 Placing Labels on Node Connections . . . 25

4 Extended Color Support **26**

1 Introduction

1.1 Overview

This user's guide explains the functionality of the PGF package. PGF stands for 'portable graphics format'. It is a TEX macro package that allows you to create graphics in your TEX documents using a special `pgfpicture` environment and special macros for drawing lines, curves, rectangles, and many other kind of graphic objects. Its usage closely resembles the PSTRICKS package or the normal picture environment of LATEX.

- Das Handbuch hatte 27 Seiten.
- Heute sind es 1165 Seiten.

2004: Die Version 0.62 von Ti*k*Z.
Die TODO-Datei

```
1) more nodes stuff
2) new "simple input mode" (more pstricks like input)
3) plotting interface to gnuplot
4) xfig interface
5) Syntax extensions, such as easy color specifications.
6) rotated shadings
7) option to reference nodes defined in other pictures
8) shading as a primitive like fill or stroke
9) \graphicspath command.
```

Gliederung

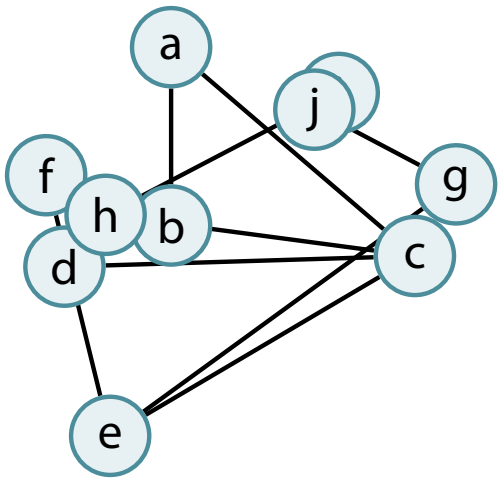
Das Wunder der Geburt

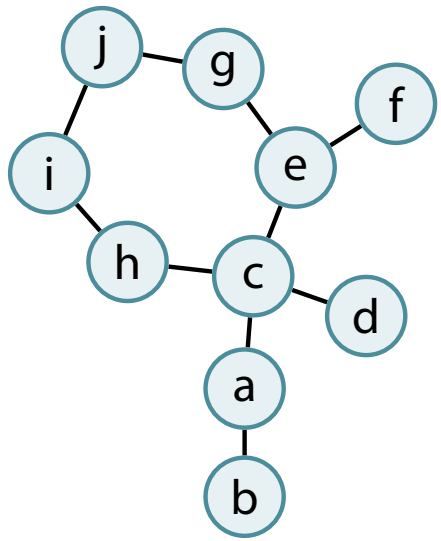
Das Wunder der Graphen

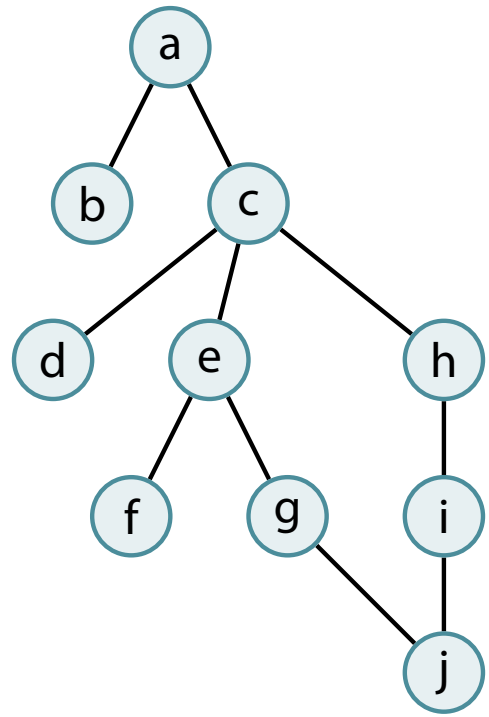
Das Wunder der Pfeile

Das Wunder der Schatten

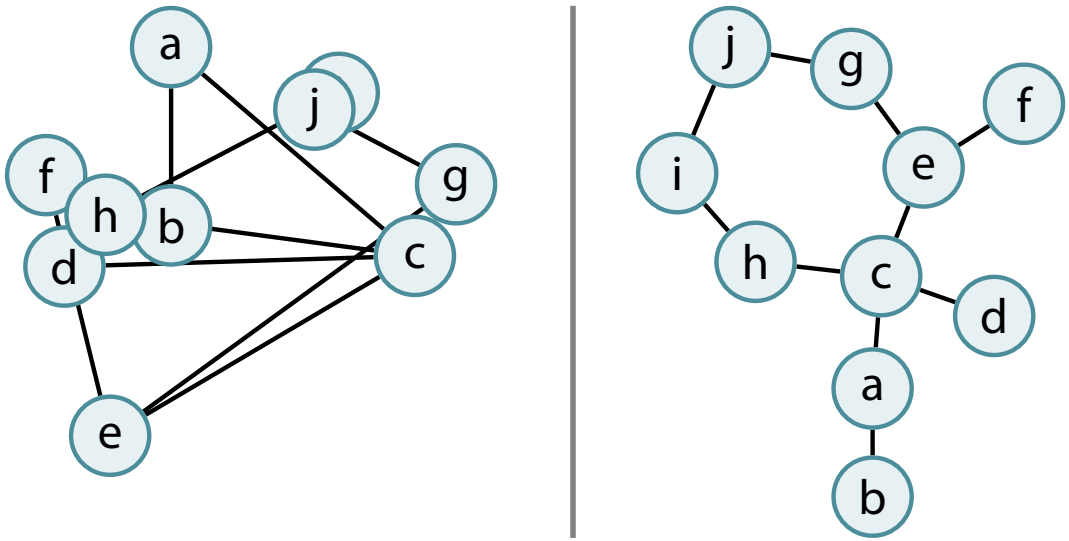
Welche Zeichnung des Graphen ist die schönste?







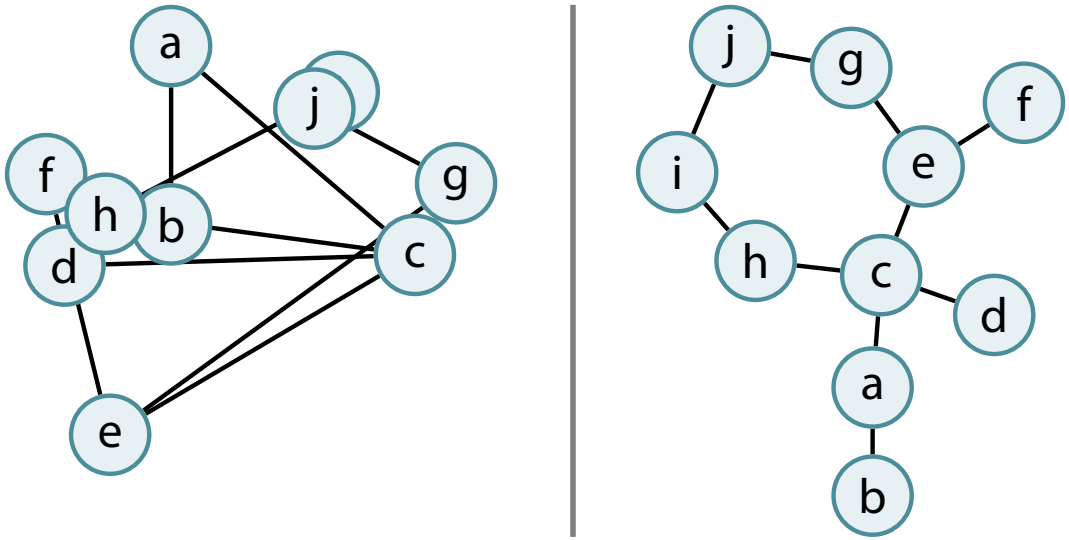
Warum ist die rechte Zeichnung besser als die linke?



Beobachtungen:

1. Rechts gibt es weniger *Überschneidungen*.
2. Rechts gibt es weniger *Überlappungen*.
3. Rechts gibt es mehr *Symmetrien*.
4. Die *Kantenlängen* sind rechts *gleichmäßiger*.
5. Die *Winkel den Knoten* sind rechts *gleichmäßiger*.

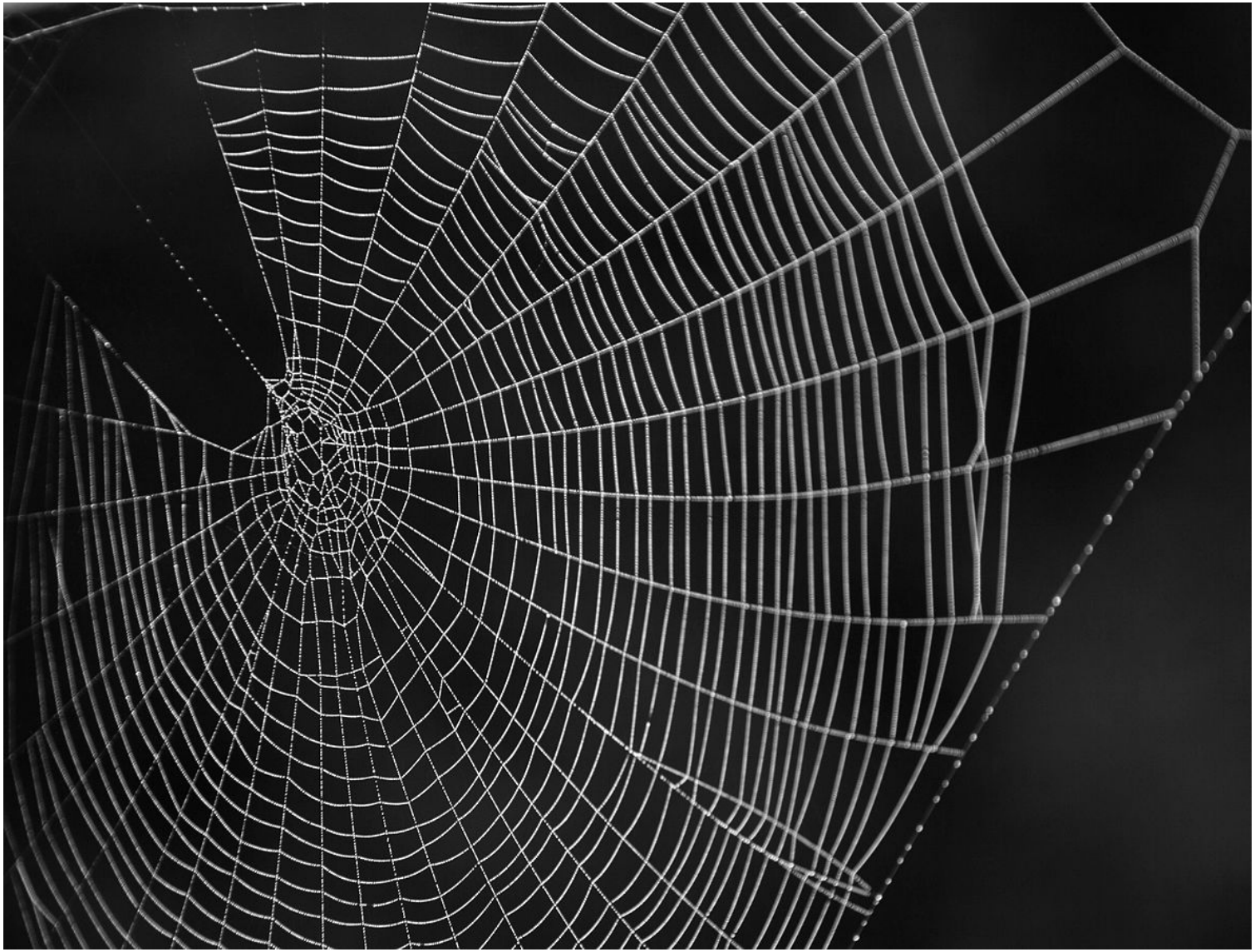
Warum ist die rechte Zeichnung besser als die linke?



Abgeleitetes Optimierungsproblem: »Zeichne Graphen so, dass

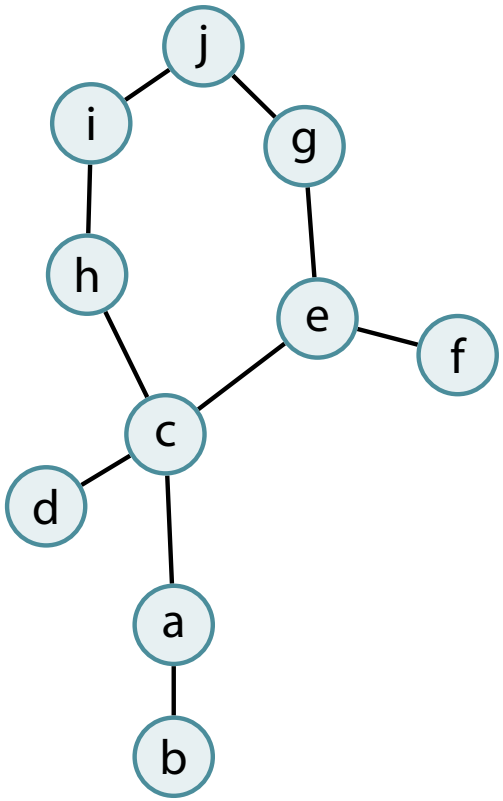
1. *Kantenüberschneidungen* minimiert werden,
2. *Knotenüberlappungen* ausgeschlossen sind,
3. *Symmetrien maximiert werden*,
4. *die Abweichung der Kantenlängen vom Idealwert* minimiert wird,
5. *die Varianz der Winkel an einem Knoten* minimiert wird.«

Die Natur erzeugt wunderschöne Zeichnungen von Graphen



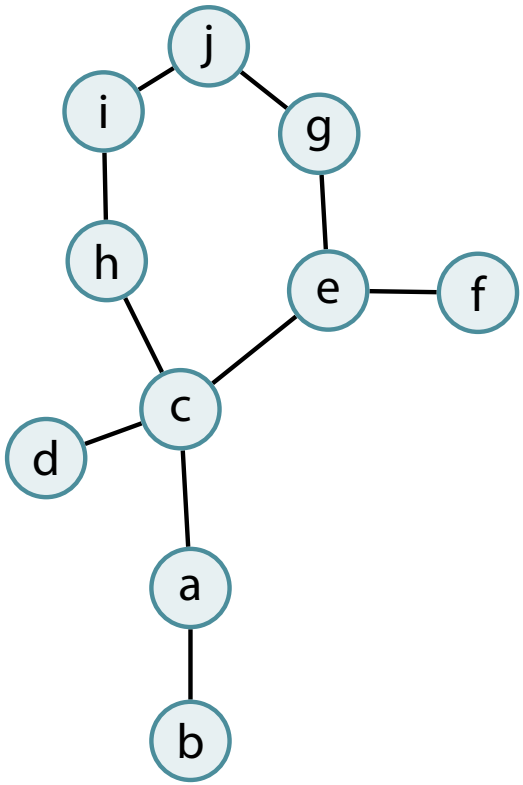
Die allgemeine Idee hinter kraftbasierten Algorithmen.

- Wir fassen die Knoten als *bewegliche Punkte* auf.
- Über die Kanten wirken dann *Kräfte* zwischen den Knoten.
- Wir *simulieren,* wie sich die Knoten bewegen,
  bis sie einen *Gleichgewichtszustand* erreichen.



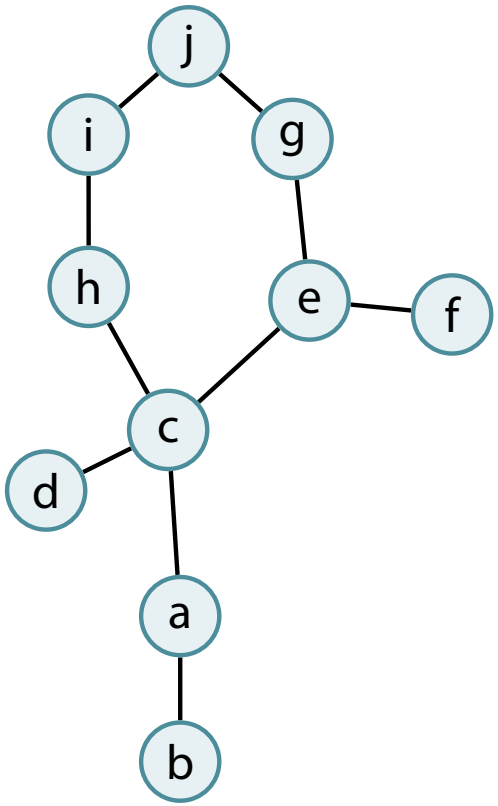
Die allgemeine Idee hinter kraftbasierten Algorithmen.

- Wir fassen die Knoten als *bewegliche Punkte* auf.
- Über die Kanten wirken dann *Kräfte* zwischen den Knoten.
- Wir *simulieren,* wie sich die Knoten bewegen,
  bis sie einen *Gleichgewichtszustand* erreichen.



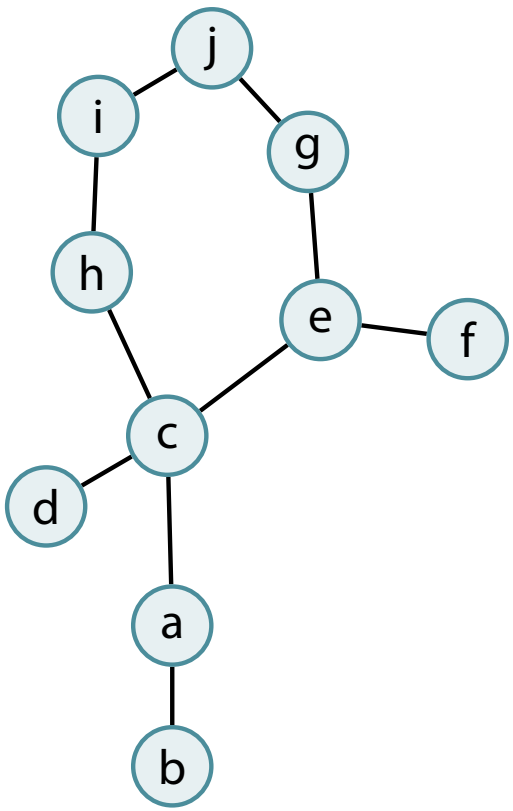
Die allgemeine Idee hinter kraftbasierten Algorithmen.

- Wir fassen die Knoten als *bewegliche Punkte* auf.
- Über die Kanten wirken dann *Kräfte* zwischen den Knoten.
- Wir *simulieren,* wie sich die Knoten bewegen,
  bis sie einen *Gleichgewichtszustand* erreichen.



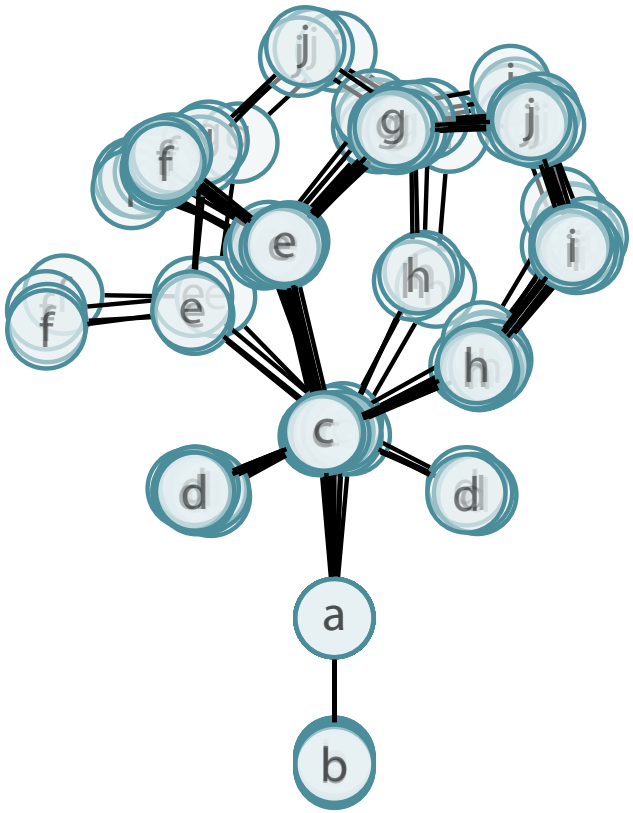
Die allgemeine Idee hinter kraftbasierten Algorithmen.

- Wir fassen die Knoten als *bewegliche Punkte* auf.
- Über die Kanten wirken dann *Kräfte* zwischen den Knoten.
- Wir *simulieren,* wie sich die Knoten bewegen,
  bis sie einen *Gleichgewichtszustand* erreichen.



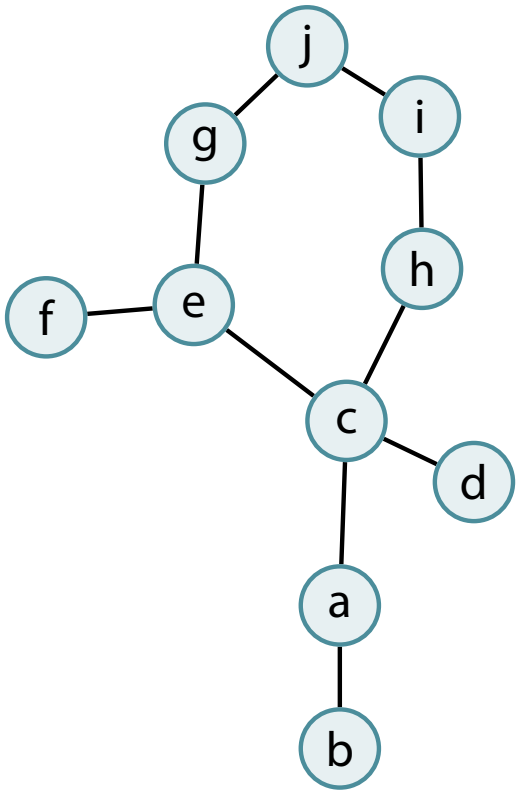
Die allgemeine Idee hinter kraftbasierten Algorithmen.

- Wir fassen die Knoten als *bewegliche Punkte* auf.
- Über die Kanten wirken dann *Kräfte* zwischen den Knoten.
- Wir *simulieren,* wie sich die Knoten bewegen,
  bis sie einen *Gleichgewichtszustand* erreichen.



Die allgemeine Idee hinter kraftbasierten Algorithmen.

- Wir fassen die Knoten als *bewegliche Punkte* auf.
- Über die Kanten wirken dann *Kräfte* zwischen den Knoten.
- Wir *simulieren,* wie sich die Knoten bewegen,
  bis sie einen *Gleichgewichtszustand* erreichen.

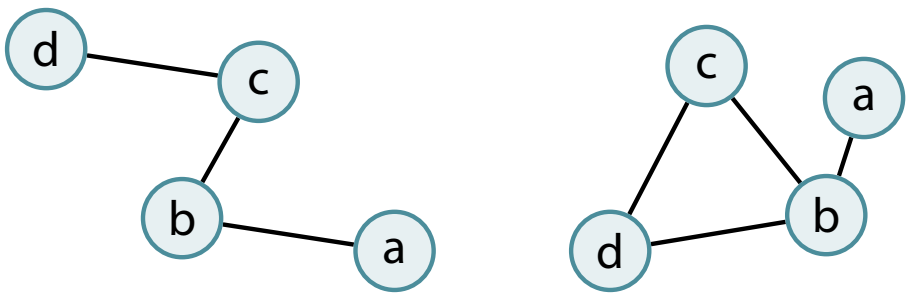


Kräfte 1 (Tutte): Federkräfte

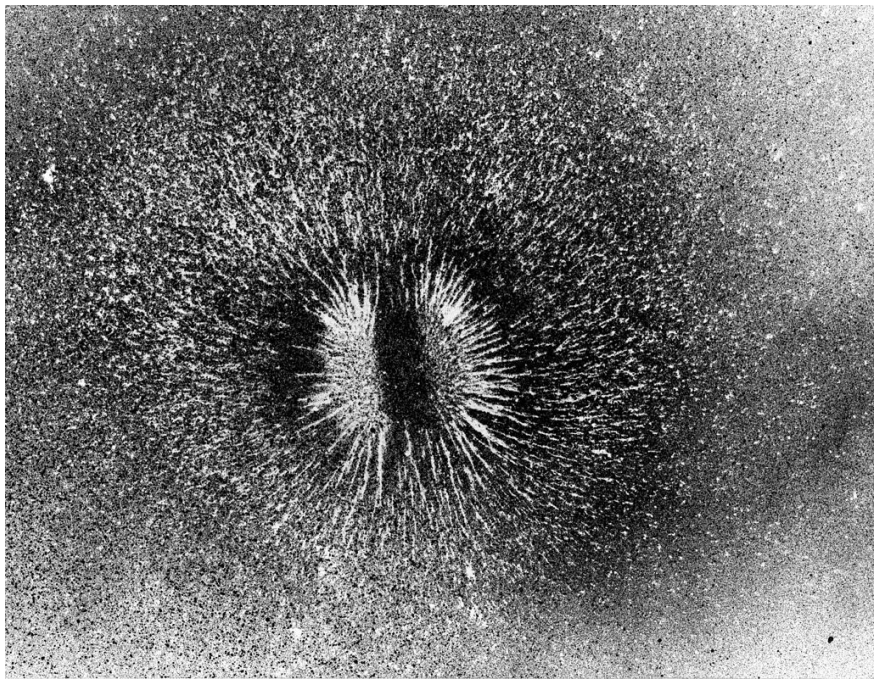
Public Domain

- Kanten sind Federn.
- Federn »möchten« eine bestimmte Länge haben.
- Sind Kanten zu kurz, »drücken sie Knoten auseinander«.
- Sind Kanten zu lang, »ziehen sie Knoten zusammen«.

Kräfte 1 (Tutte): Federkräfte

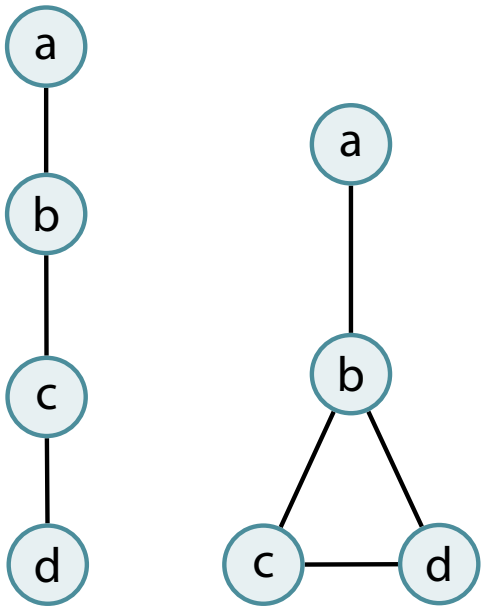


Kräfte 2 (Eades): Elektrische Kräfte

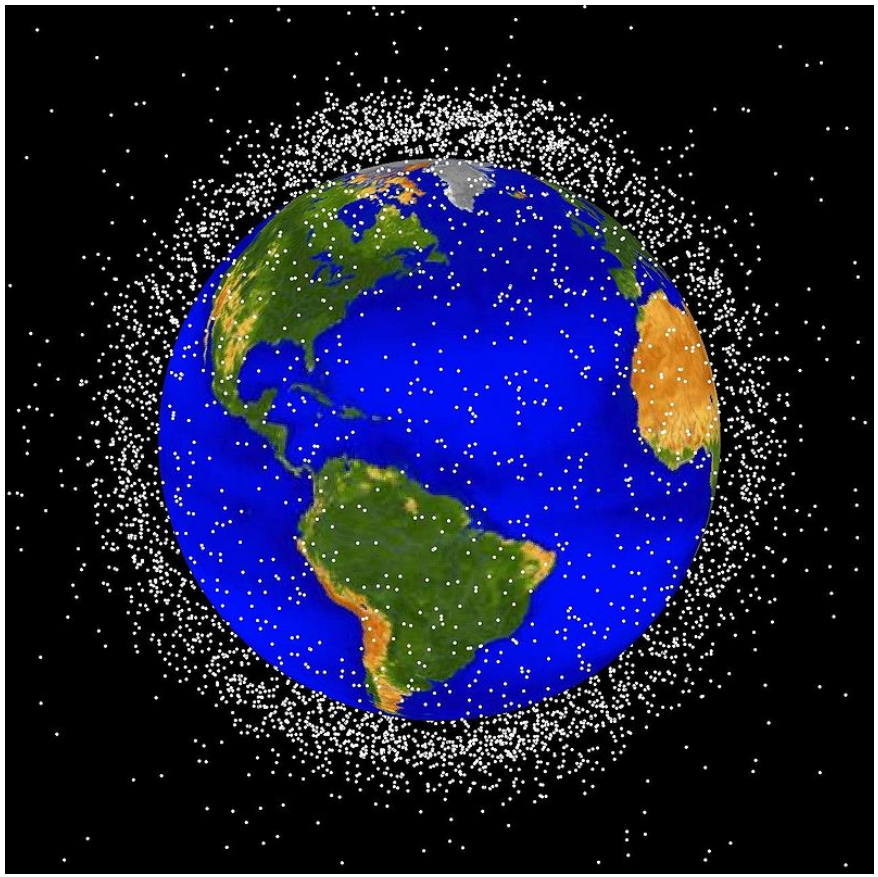


- Zwischen Knoten gibt es zusätzlich *abstoßende Kräfte*.
- Knoten richten sich daher »gerne« in »Linien« und »Kreisen« aus.
- Winkel gleichen sich aus.

Kräfte 2 (Eades): Elektrische Kräfte



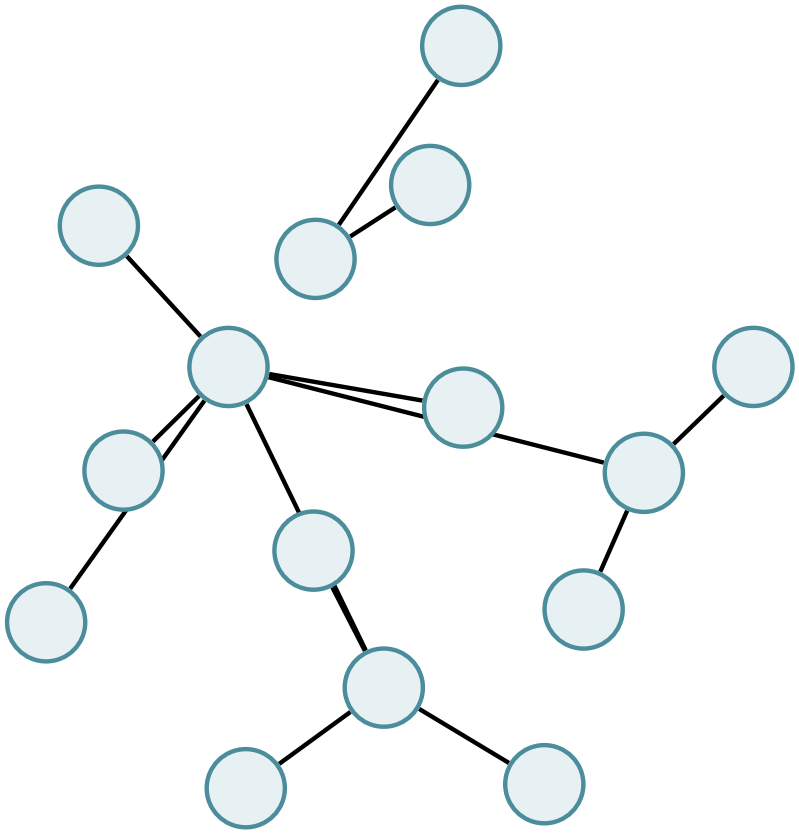
Kräfte 3: Graviationskräfte



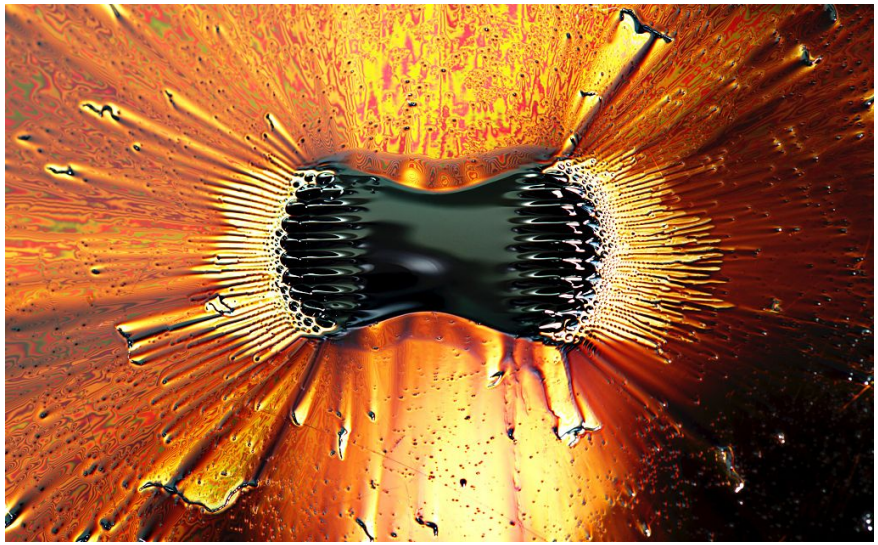
Public Domain

- Knoten ziehen sich zusätzlich an.
- Dies zieht »wichtige« Knoten in die Mitte.

Kräfte 3: Graviationskräfte



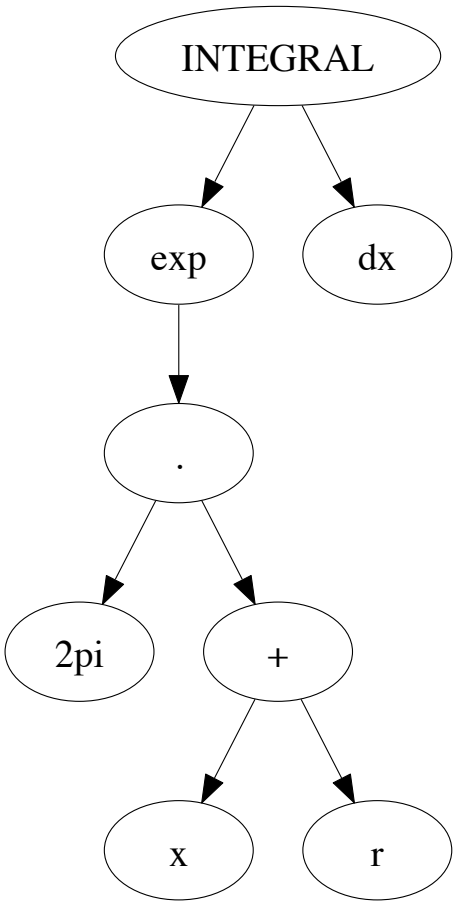
Kräfte 4: Magnetische Kräfte

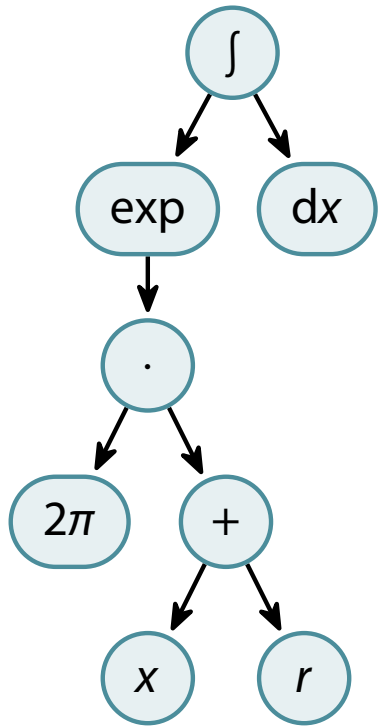


GNU Free Documentation License, Autor Gregory F. Maxwell

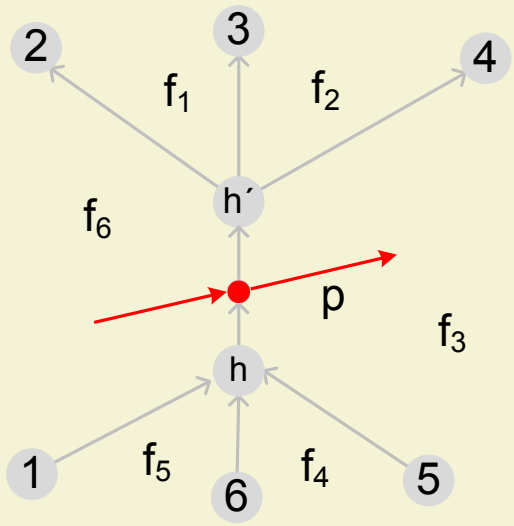
- Kanten »möchten sich entlang von Magnetfeldern« ausrichten.
- Hierdurch kann man *horizontale* und *vertikale* Kanten bevorzugen.

Integrieren von Graphzeichnungen in Dokumenten ist schwierig





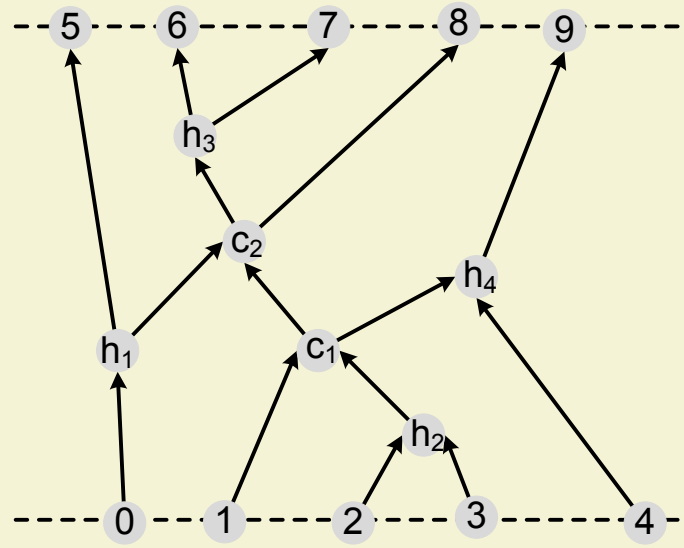
Integrieren von Graphzeichnungen in Dokumenten ist schwierig



(b) A realization of p.

al arc can reduce the crossings

aints, are restricted to the



(a)

3. Steps towards a final layout: (a) PR $\mathcal{R}$, (b) fine-layering of the subgra

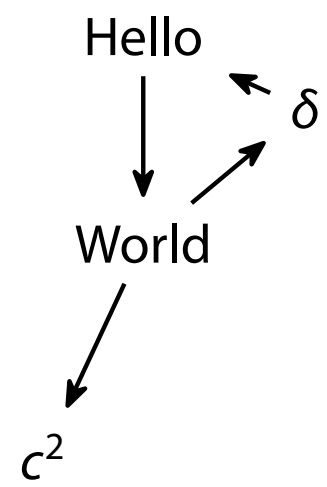
Die Lösung: Graphzeichnen in Ti*k*Z

- Man kombiniere eine *existierende Dokumentensprache* (TEX) mit einer *existierenden Graphiksprache* (Ti*k*Z).
- Füge Syntax hinzu, mit denen sich *Graphen leicht angeben lassen*.
- Führe die Graphzeichnen-Algorithmen als *Teil der Dokumentverarbeitung* aus.
- Das Aussehen von Knoten und Kanten passt nun zum Dokument.

Eine bessere Syntax für Graphen

- Eine knappe, gute Syntax für Graphen ist wichtig, wenn *Menschen Graphen »per Hand«* beschreiben wollen.
- Die neue Syntax *mischt die Philosophien* von DOT und Ti*k*Z.

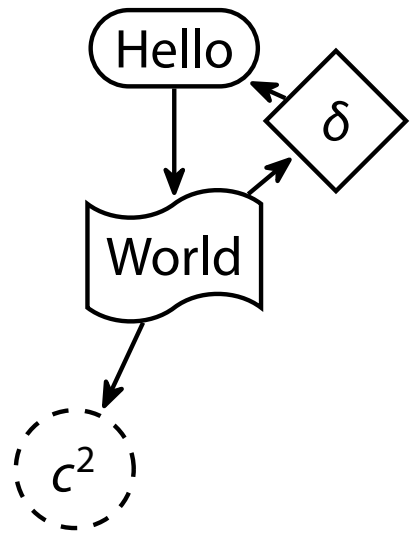
```
\tikz \graph[spring electrical layout] {
  Hello -> World -> "$c^2$";
  World -> "$\delta$" -> Hello;
};
```



Zentrale Eigenschaften der TikZ-Syntax für Graphen

- *Knotenoptionen folgen Knoten in eckigen Klammern.*
- Kantenoptionen folgen Kanten in eckigen Klammern.
- Spezielle Notation für Kanten.
- Bäume lassen sich sehr natürlich beschreiben.

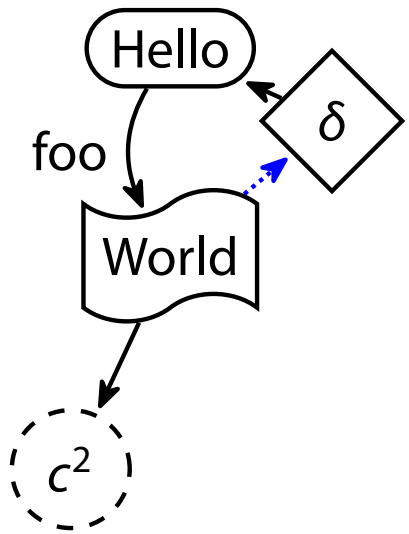
```
\tikz \graph[spring electrical layout] {
      Hello    [rounded rectangle]
   ->  World    [tape]
   ->  "$c^2$"  [circle, dashed];

   World
   ->  "$\delta$" [diamond]
   ->  Hello;
};
```



Zentrale Eigenschaften der Ti*k*Z-Syntax für Graphen

- Knotenoptionen folgen Knoten in eckigen Klammern.
- *Kantenoptionen folgen Kanten in eckigen Klammern.*
- Spezielle Notation für Kanten.
- Bäume lassen sich sehr natürlich beschreiben.

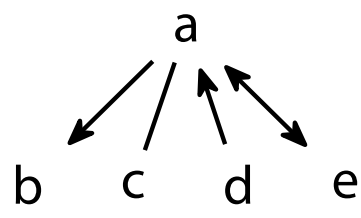
```
\tikz \graph[spring electrical layout] {
      Hello    [rounded rectangle]
  ->[bend right, "foo"']
      World    [tape]
  ->  "$c^2$" [circle, dashed];

  World
  ->[densely dotted, blue]
      "$\delta$"[diamond]
  -> Hello;
};
```



Zentrale Eigenschaften der Ti*k*Z-Syntax für Graphen

- Knotenoptionen folgen Knoten in eckigen Klammern.
- Kantenoptionen folgen Kanten in eckigen Klammern.
- *Spezielle Notation für Kanten.*
- Bäume lassen sich sehr natürlich beschreiben.

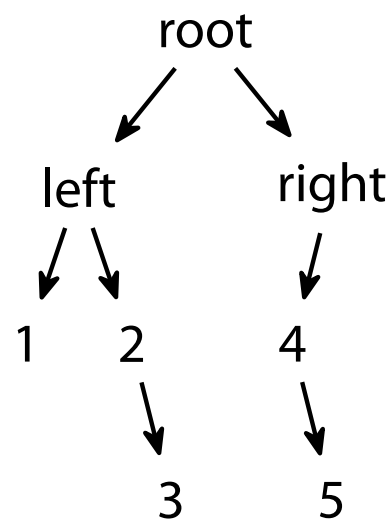
```
\tikz \graph [tree layout] {
  a -> b -- c <- d <-> e;
};
```



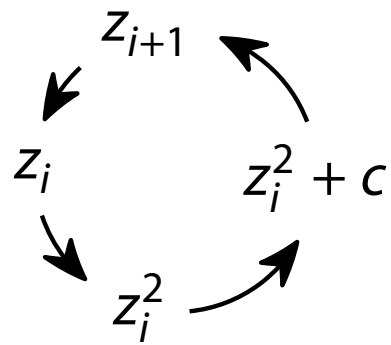
Zentrale Eigenschaften der Ti*k*Z-Syntax für Graphen

- Knotenoptionen folgen Knoten in eckigen Klammern.
- Kantenoptionen folgen Kanten in eckigen Klammern.
- Spezielle Notation für Kanten.
- *Bäume lassen sich sehr natürlich beschreiben.*

```
\tikz \graph [binary tree layout] {
  root -> {
    left -> {
      1,
      2 -> 3 [second]
    },
    right -> {
      4 -> { , 5 }
    }
  }
};
```

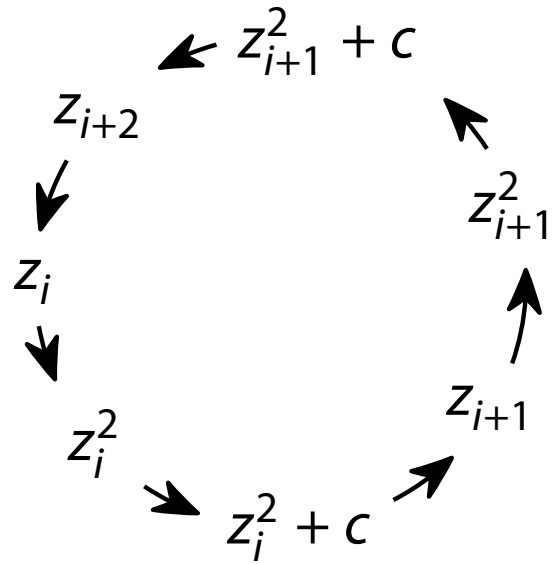


Das Wunder des Kreises.




```
\tikz \graph [simple necklace layout, necklace routing] {
  "$z_i$"  -> "$z_i^2$"
           -> "$z_i^2 + c$"
           -> "$z_{i+1}$"
           -> "$z_i$"
};
```

Das Wunder des Kreises.




```
\tikz \graph [simple necklace layout, necklace routing] {
  "$z_i$"  -> "$z_i^2$"
           -> "$z_i^2 + c$"
           -> "$z_{i+1}$"
           -> "$z_{i+1}^2$"
           -> "$z_{i+1}^2 + c$"
           -> "$z_{i+2}$"
           -> "$z_i$"
};
```

Gliederung

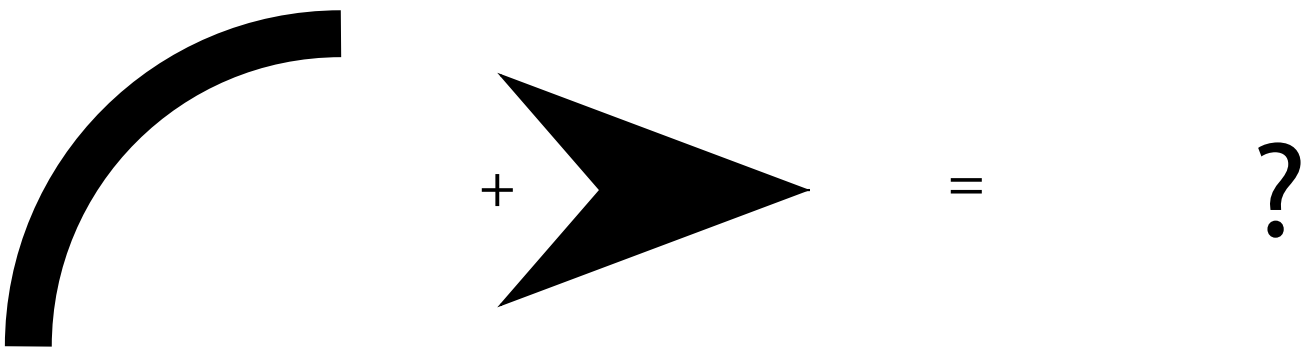
Das Wunder der Geburt

Das Wunder der Graphen

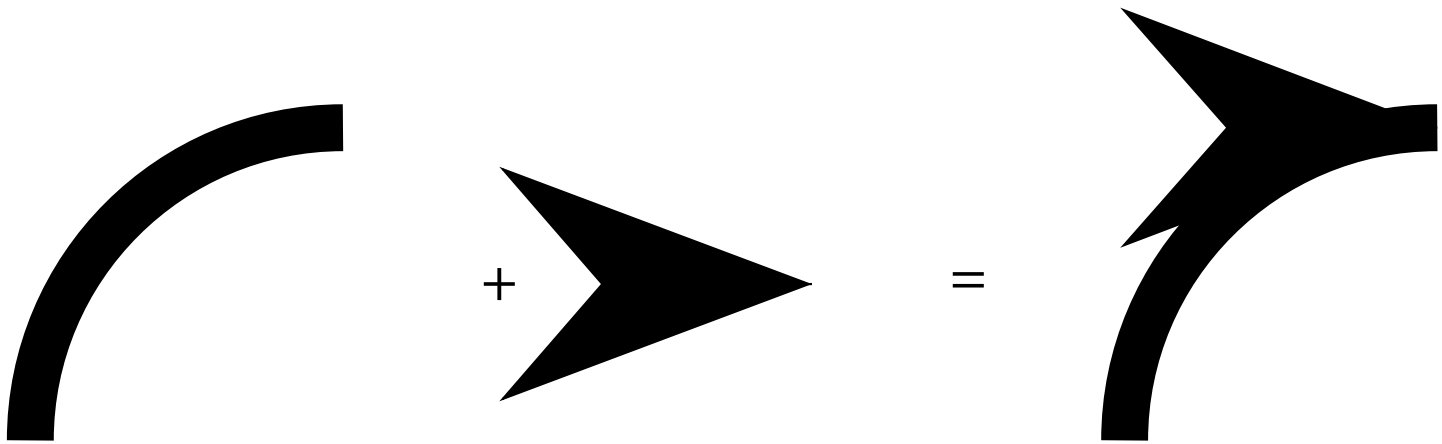
Das Wunder der Pfeile

Das Wunder der Schatten

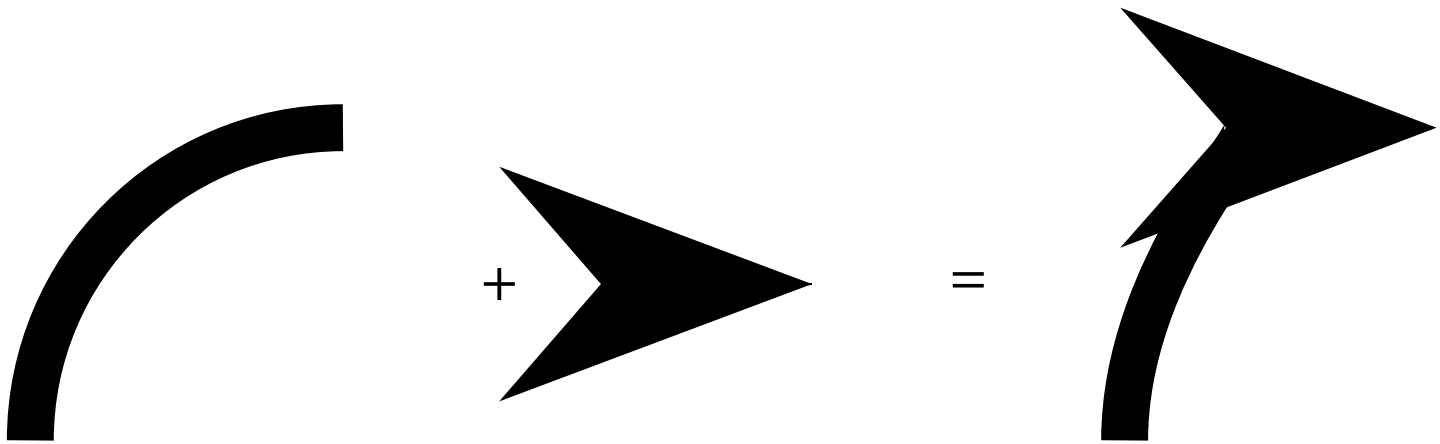
Warum es so schwierig ist, Pfeile richtig zu zeichnen.



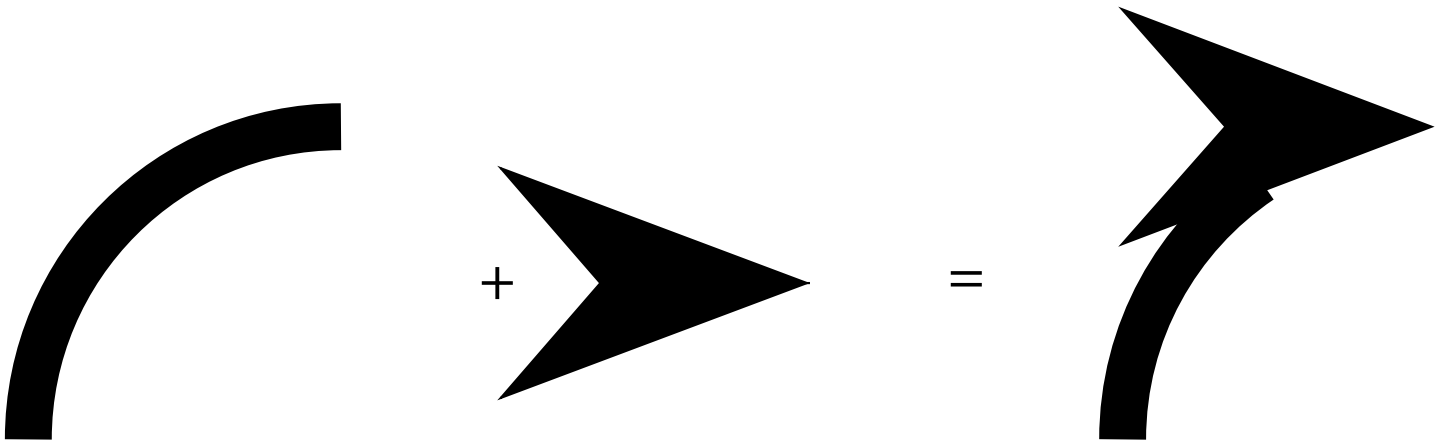
Warum es so schwierig ist, Pfeile richtig zu zeichnen.



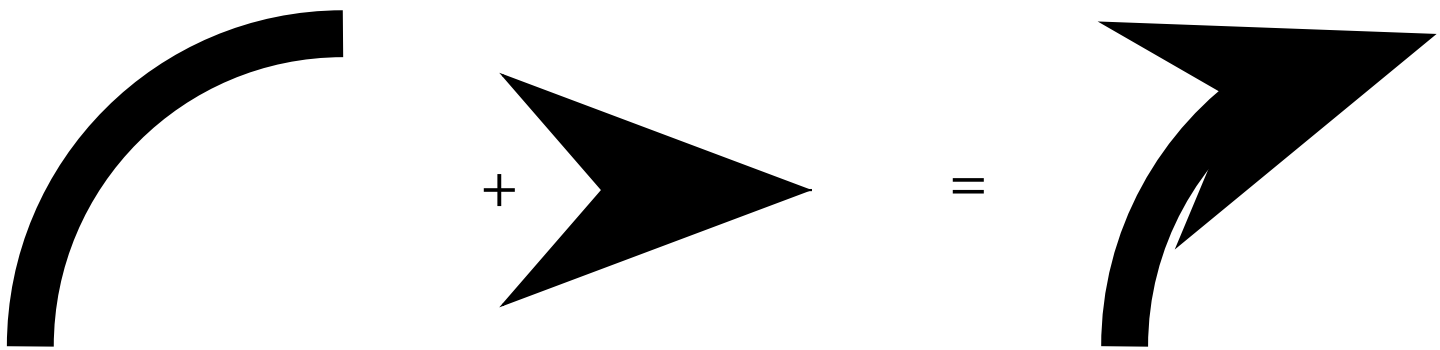
Warum es so schwierig ist, Pfeile richtig zu zeichnen.



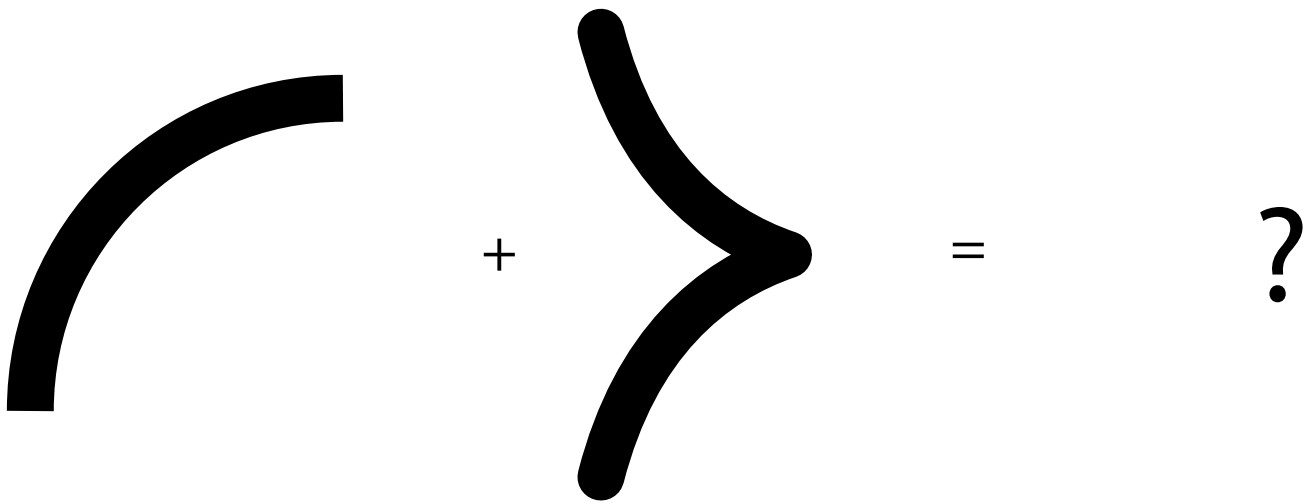
Warum es so schwierig ist, Pfeile richtig zu zeichnen.



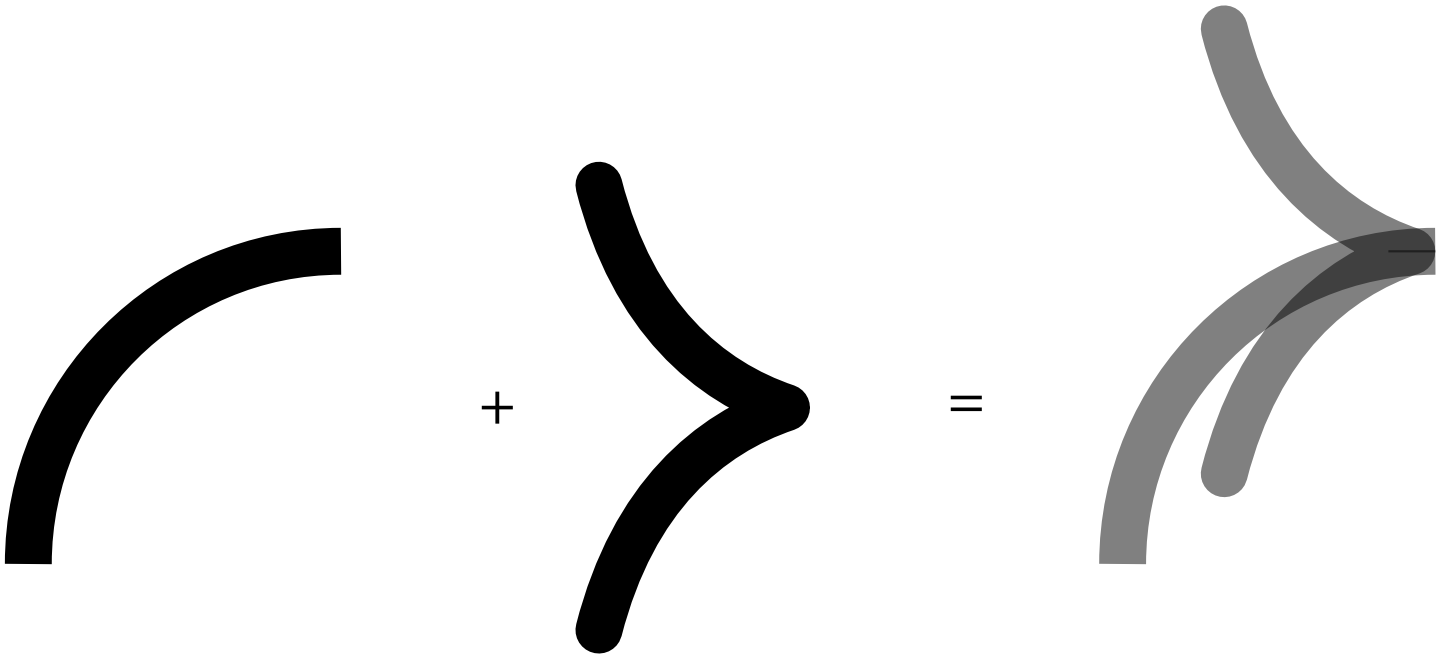
Warum es so schwierig ist, Pfeile richtig zu zeichnen.



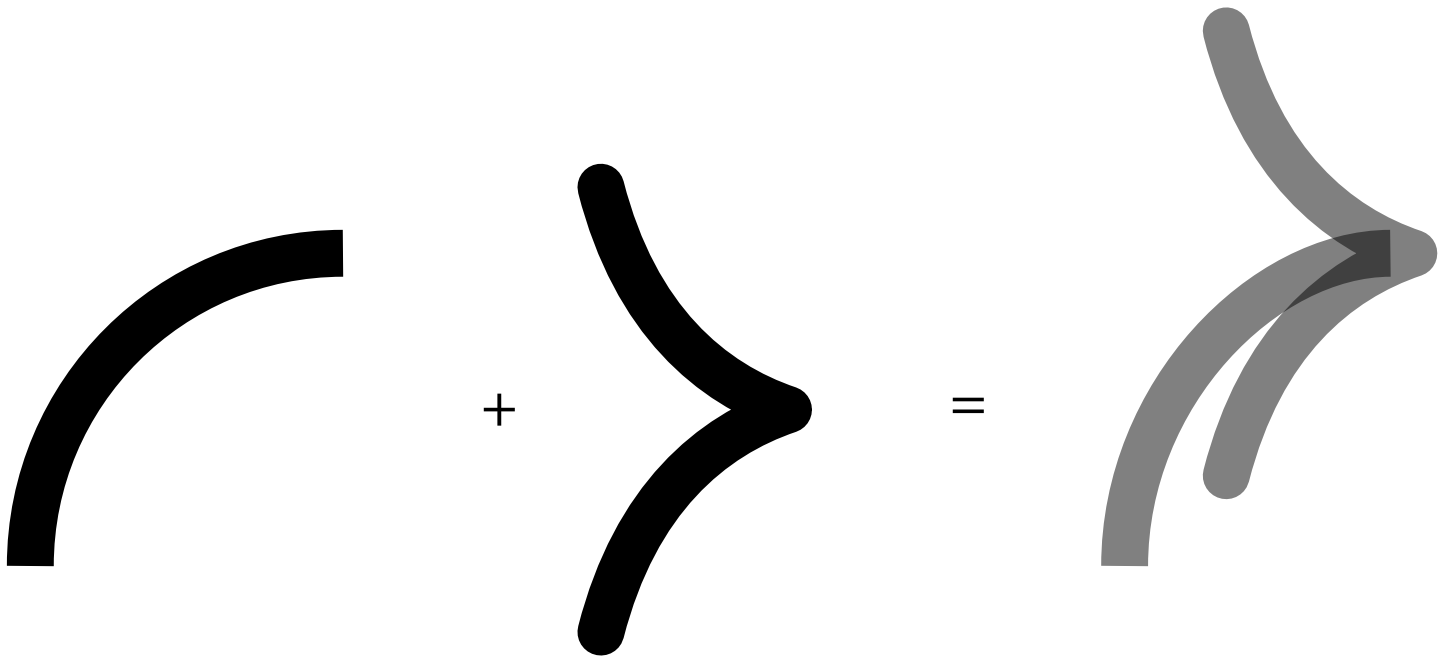
Warum es so schwierig ist, Pfeile richtig zu zeichnen.



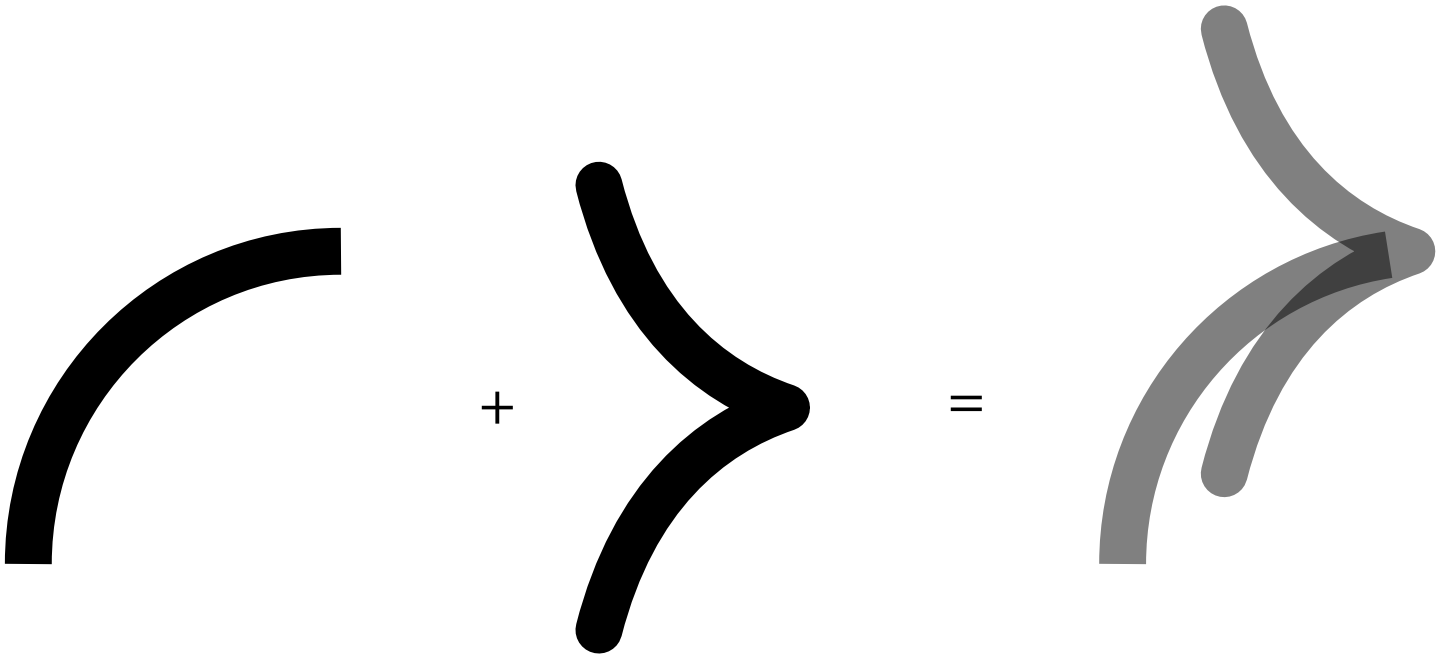
Warum es so schwierig ist, Pfeile richtig zu zeichnen.



Warum es so schwierig ist, Pfeile richtig zu zeichnen.



Warum es so schwierig ist, Pfeile richtig zu zeichnen.



Es ist schwierig, neue Arten Pfeilspitzen zu definieren…

- TikZ muss »sehr viel wissen« über die Pfeilspitzen (insbesondere sind die »Dreh- und Angelpunkte« wichtig sowie die »Länge«).
- TikZ muss »oft und schnell« Pfeilspitzen zeichnen.

```
\pgfdeclarearrow{
  name = Computer Modern Rightarrow,
  defaults = {
    length = +1.6pt 2.2, % 0pt 5.2
    width' = +0pt 2.096774,
    line width = 0pt 1 1,
    round
  },
  setup code =
  {
    % inner length:
    \pgfutil@tempdima\pgfarrowlength
    \advance\pgfutil@tempdima
      by-\pgfarrowlinewidth
    \pgfutil@tempdimb\pgfarrowwidth
    \advance\pgfutil@tempdimb
      by-\pgfarrowlinewidth
    % The following are needed in the code:
... 58 Zeilen ...
```

```
...
    \else
      \pgfpathcurveto
        {\pgfqpoint{-0.41019\pgfutil@tempdima}{-0.
        {\pgfqpoint{-0.81731\pgfutil@tempdima}{-.2
        {\pgfqpoint{-\pgfutil@tempdima}{-.5\pgfuti
    \fi
    \pgfusepathqstroke
  },
  parameters = {
    \the\pgfarrowlinewidth,%
    \the\pgfarrowlength,%
    \the\pgfarrowwidth,%
    \ifpgfarrowharpoon h\fi
    \ifpgfarrowharpoon r\fi
    \ifpgfarrowroundjoin j\fi%
    \ifpgfarrowroundcap c\fi%
  },
}
```

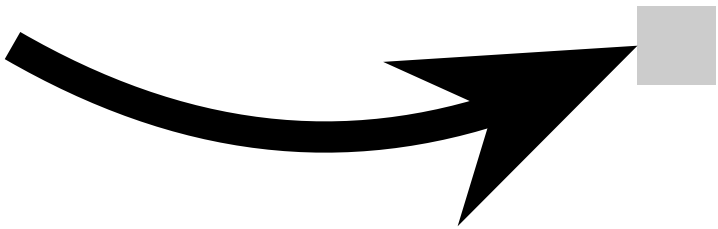
…aber es ist auch nicht mehr nötig.

```
\usetikzlibrary{arrows.meta}
...

\draw [ arrows = - ]
  (0,1) edge[bend right] (4,1);
```

…aber es ist auch nicht mehr nötig.

```
\usetikzlibrary{arrows.meta}
...

\draw [ arrows = -{Stealth} ]
  (0,1) edge[bend right] (4,1);
```

…aber es ist auch nicht mehr nötig.

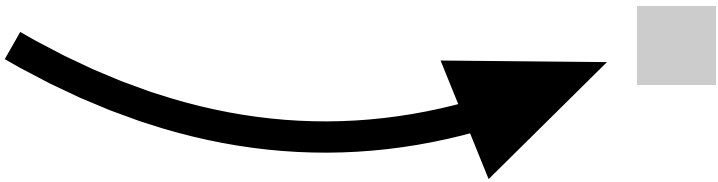


```
\usetikzlibrary{arrows.meta}
...


\draw [ arrows = -{Stealth[scale=1.5]} ]
  (0,1) edge[bend right] (4,1);
```

…aber es ist auch nicht mehr nötig.

```
\usetikzlibrary{arrows.meta}
...


\draw [ arrows = -{Stealth[inset=0pt]} ]
  (0,1) edge[bend right] (4,1);
```

…aber es ist auch nicht mehr nötig.

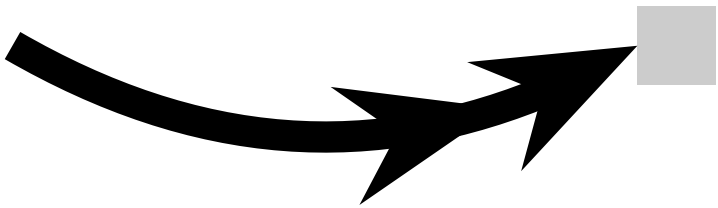


```
\usetikzlibrary{arrows.meta}
...


\draw [ arrows = -{Stealth[inset=0pt, sep=2mm]} ]
  (0,1) edge[bend right] (4,1);
```

…aber es ist auch nicht mehr nötig.

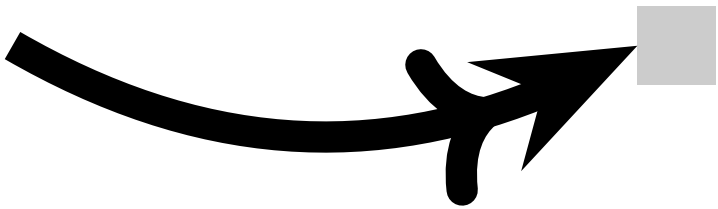
```
\usetikzlibrary{arrows.meta}
...


\draw [ arrows = -{Stealth[round, scale=2, open]} ]
  (0,1) edge[bend right] (4,1);
```

…aber es ist auch nicht mehr nötig.



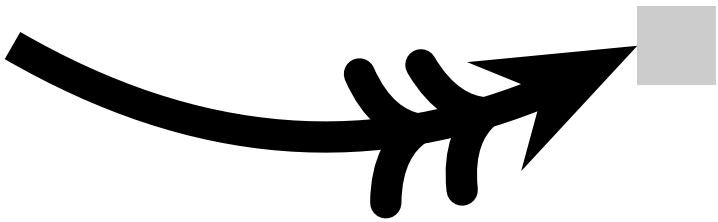
```
\usetikzlibrary{arrows.meta}
...


\draw [ arrows = -{Stealth[] Stealth[]} ]
  (0,1) edge[bend right] (4,1);
```

…aber es ist auch nicht mehr nötig.



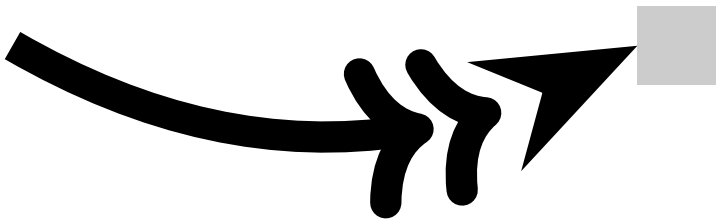
```
\usetikzlibrary{arrows.meta}
...


\draw [ arrows = -{To[] Stealth[]} ]
  (0,1) edge[bend right] (4,1);
```

…aber es ist auch nicht mehr nötig.



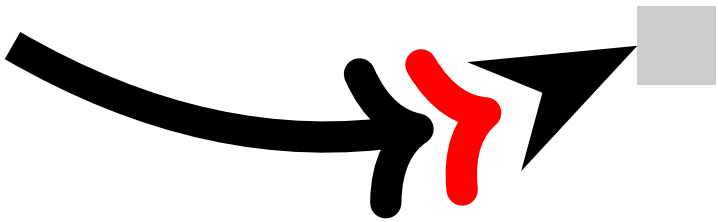
```
\usetikzlibrary{arrows.meta}
...

\draw [ arrows = -{To[] To[] Stealth[]} ]
  (0,1) edge[bend right] (4,1);
```

…aber es ist auch nicht mehr nötig.



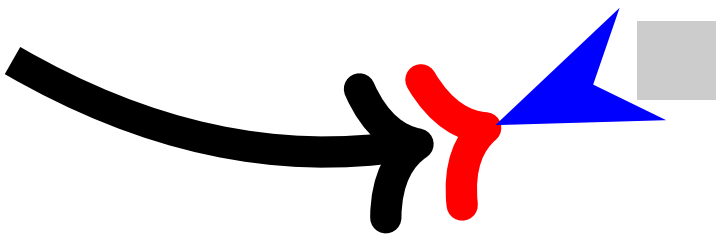
```
\usetikzlibrary{arrows.meta}
...


\draw [ arrows = -{To[]. To[] Stealth[]} ]
  (0,1) edge[bend right] (4,1);
```

…aber es ist auch nicht mehr nötig.



```
\usetikzlibrary{arrows.meta}
...


\draw [ arrows = -{To[]. To[red] Stealth[]} ]
  (0,1) edge[bend right] (4,1);
```

…aber es ist auch nicht mehr nötig.

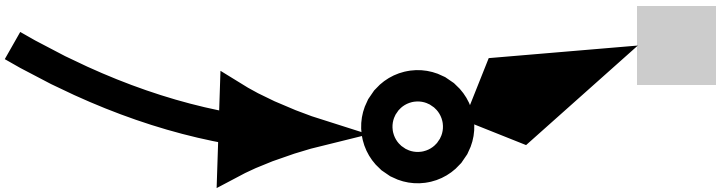


```
\usetikzlibrary{arrows.meta}
...

\draw [ arrows = -{To[]. To[red] Stealth[reversed,blue]} ]
  (0,1) edge[bend right] (4,1);
```

…aber es ist auch nicht mehr nötig.

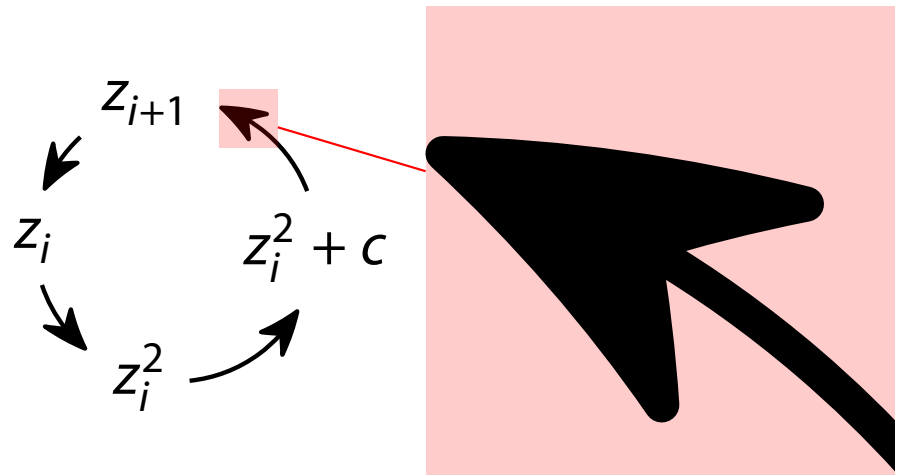
```
\usetikzlibrary{arrows.meta}
...
\draw [ arrows = -{Arc Barb[sep]. Bar[sep] Bracket[sep]
                   Hooks[sep] Parenthesis[sep]
                   Straight Barb[sep] Tee Barb[]} ]
  (0,1) edge[bend right] (4,1);
```

…aber es ist auch nicht mehr nötig.



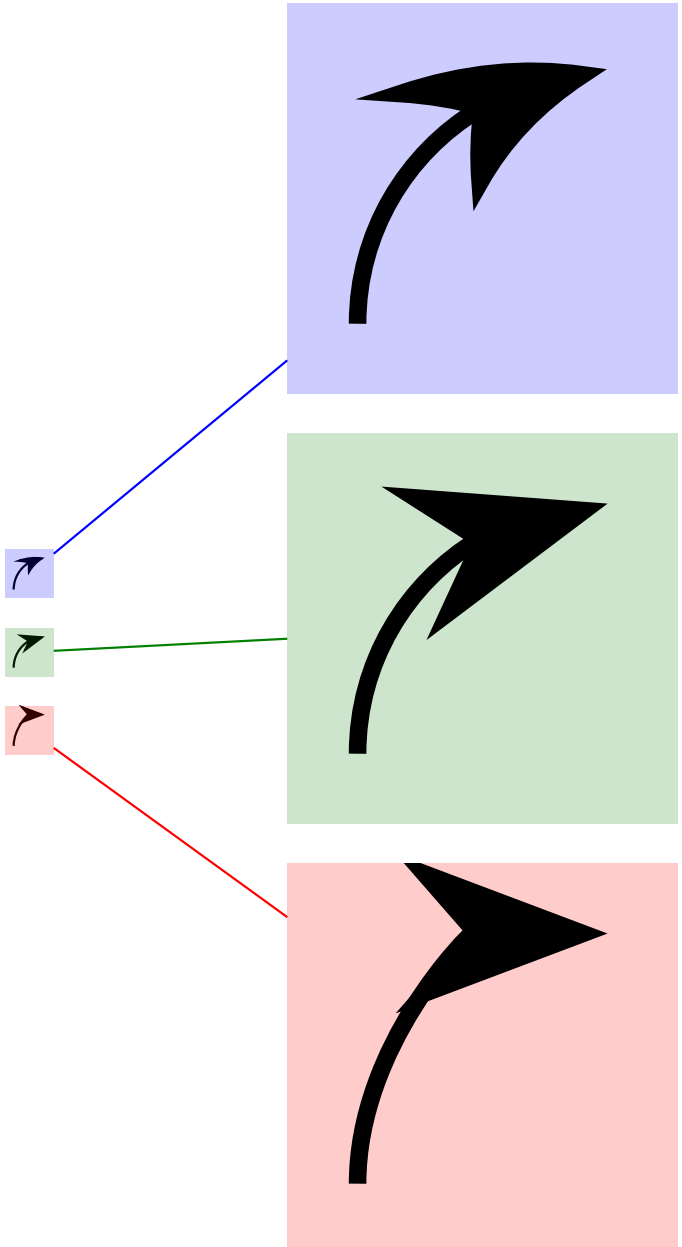
```
\usetikzlibrary{arrows.meta}
...


\draw [ arrows = -{Latex[]. Circle[open] Kite[]} ]
  (0,1) edge[bend right] (4,1);
```

Das Wunder des Kreises.




```
\tikz \graph [simple necklace layout, necklace routing] {
  "$z_i$"  -> "$z_i^2$"
           -> "$z_i^2 + c$"
           -> "$z_{i+1}$"
           -> "$z_i$"
};
```

TikZ kann Pfeile auf drei Arten »biegen«.

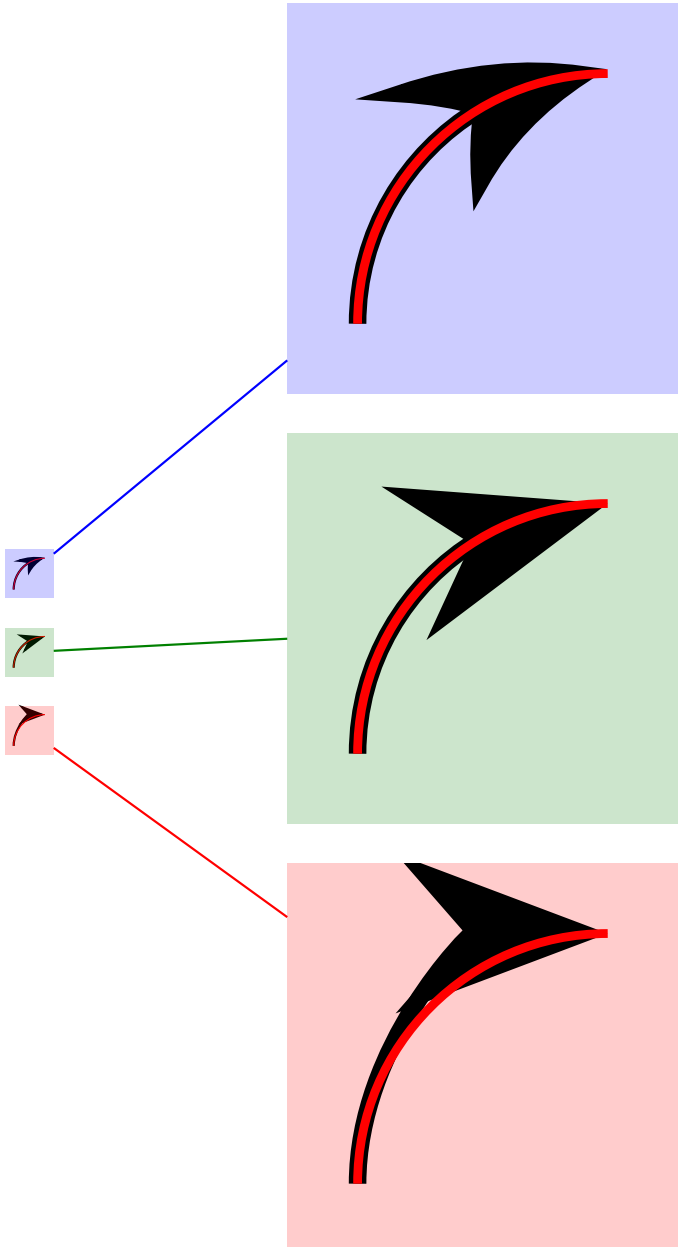


```
\tikzset{ > = Stealth[bend] }
\draw [->] (0,0)
      arc (180:90:2mm);
```

```
\tikzset{ > = Stealth[flex] }
\draw [->] (0,0)
      arc (180:90:2mm);
```

```
\tikzset{ > = Stealth[quick] }
\draw [->] (0,0)
      arc (180:90:2mm);
```

TikZ kann Pfeile auf drei Arten »biegen«.

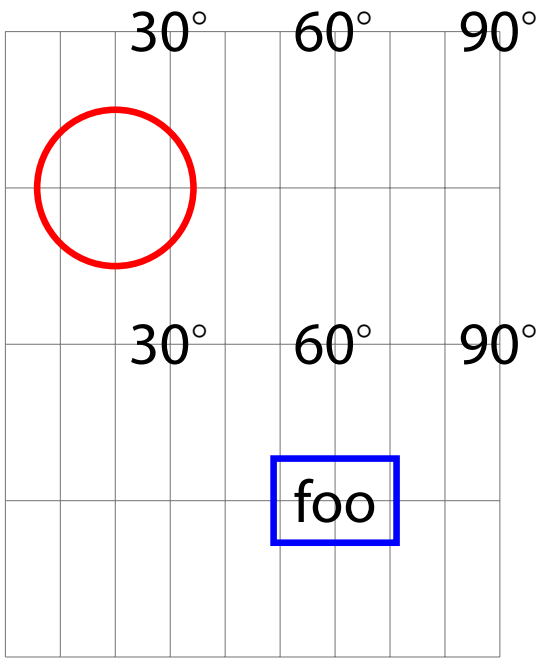


```
\tikzset{ > = Stealth[bend] }
\draw [->] (0,0)
      arc (180:90:2mm);
```

```
\tikzset{ > = Stealth[flex] }
\draw [->] (0,0)
      arc (180:90:2mm);
```

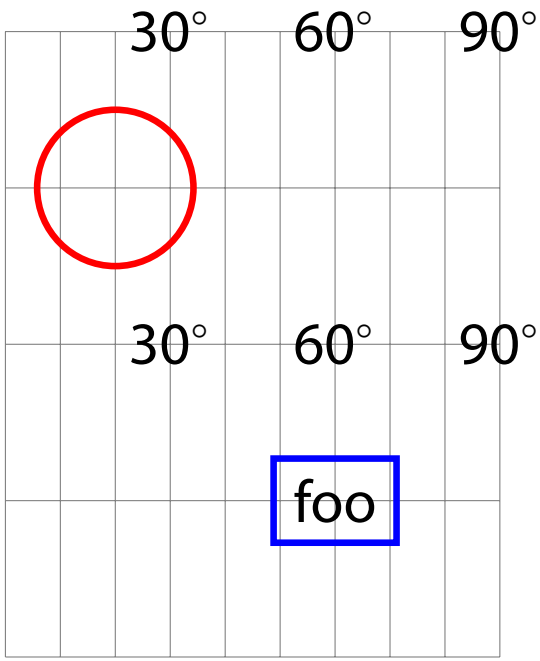
```
\tikzset{ > = Stealth[quick] }
\draw [->] (0,0)
      arc (180:90:2mm);
```

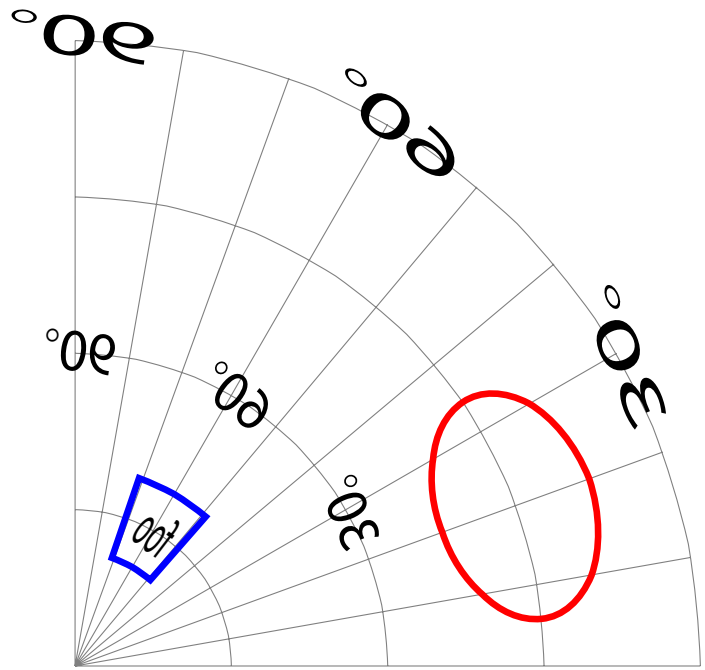
TikZ kann viele Dinge »biegen« – insbesondere Pfeilspitzen.




```
\draw [help lines] (0pt,0mm) grid[xstep=10pt] (90pt, 4cm);
\draw [very thick, red] (20pt,3cm) circle [radius=5mm];
\node [draw=blue, very thick] at (60pt,1cm) {foo};
\foreach \angle in {0,30,60,90}
  \foreach \dist in {2,4}
    \node at (\angle pt, \dist cm) {\angle$^\circ$};
```

TikZ kann viele Dinge »biegen« – insbesondere Pfeilspitzen.






```
\pgftransformnonlinear{\polartransformation}
\draw [help lines] (0pt,0mm) grid[xstep=10pt] (90pt, 4cm);
\draw [very thick, red] (20pt,3cm) circle [radius=5mm];
\node [draw=blue, very thick] at (60pt,1cm) {foo};
\foreach \angle in {0,30,60,90}
  \foreach \dist in {2,4}
    \node at (\angle pt, \dist cm) {\angle$^\circ$};
```

Gliederung

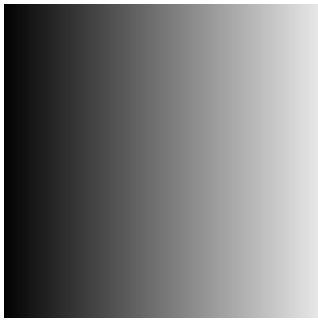
Das Wunder der Geburt

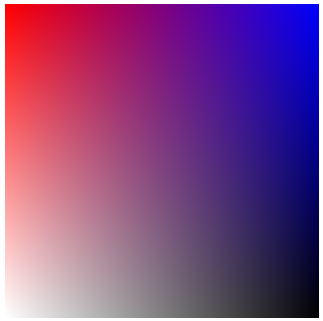
Das Wunder der Graphen

Das Wunder der Pfeile

Das Wunder der Schatten

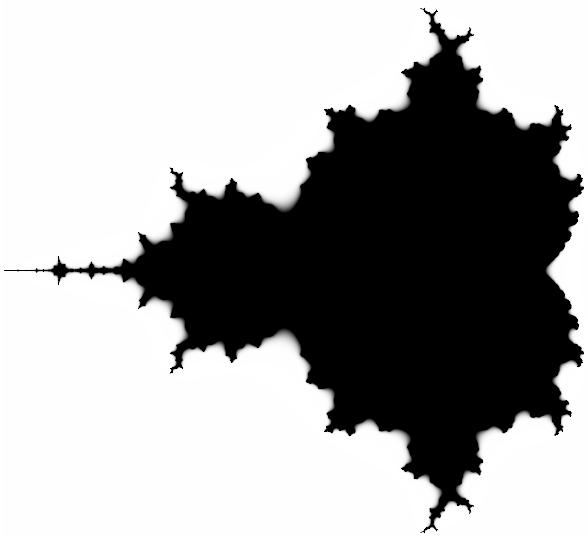
»Schatten« sind »Farbverläufe«.





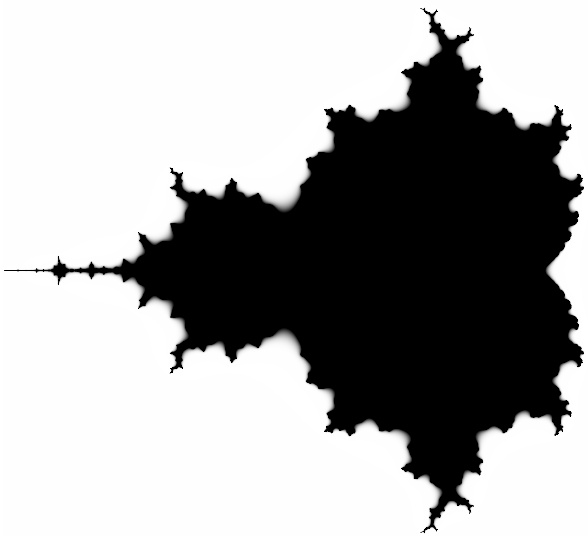
- Bei einem »Farbverlauf« hat jeder Punkt eine eigene Farbe.
- Man braucht eine Funktion, die Koordinaten (x, y) auf Farben (r, g, b) abbildet.
- Funktionen wie die für einen *linearen Verlauf* sind vordefiniert,…
- …man kann aber auch selber Funktionen definieren…
- …*sogar die für die Mandelbrotmenge!*

»Schatten« sind »Farbverläufe«.



```
\shade[shading=Mandelbrot set]
  (-3,-3) rectangle (3,3);
```

»Schatten« sind »Farbverläufe«.



```
\pgfdeclarefunctionalshading{Mandelbrot set}
{\pgfpoint{-50bp}{-50bp}}
{\pgfpoint{50bp}{50bp}}{}
{
  12.5 div exch 12.5 div exch
  1 index 1 index
  % Stack: c_r c_i z_r z_i
  % Formula: z' = z^2 + c = (z_r + i z_i)^2 + c_r + i c_i
  %          = (z_r^2 - z_i^2 + c_r) + i (2 z_r z_i + c_i)
  %
  % First iteration
  % 1. Compute z_r^2 -z_i^2 + c_r
  1 index dup mul % z_r^2
  1 index dup mul % z_i^2
  sub             % z_r^2 - z_i^2
  4 index add % z_r^2 -z_i^2 + c_r
  % 2. Compute 2 z_r z_i + c_i
  3 1 roll
  mul 2 mul    % 2 z_r z_i
  2 index add % 2 z_r z_i + c_i
  % Second iteration

... 96 Zeilen ...

  add sqrt
  dup 4 1 roll
  2 gt { pop pop 2.0 exch
    div 1.0 exch sub dup dup}
    {pop pop 0.0 0.0 0.0} ifelse
}
```

Zusammenfassung

Ti*k*Z ist voller kleiner Wunder.