Tegnekurs i Ti*k*Z

Veronika Heimsbakk
veronahe@ulrik.uio.no

6. oktober 2014

Innhold

Figurer

1 The Basics

For å kunne bruke pakken Ti*k*Z må man først inkludere pakken i dokumentet.

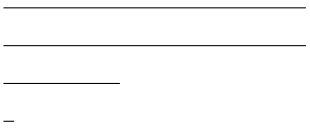
```
\usepackage{tikz}
```

1.1 `tikzpicture`

Alle illustrasjoner som skal tegnes ved hjelp av pakken Ti*k*Z krever et miljø som heter `tikzpicture`.

```
\begin{tikzpicture}
    <kode her>
\end{tikzpicture}
```

1.2 Linjer

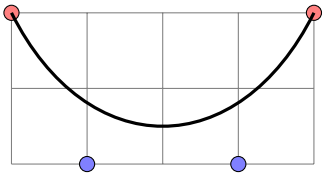


En av de mest brukte Ti*k*Z kommandoene er `\draw`. For å tegne ei rett linje sier man hvor man vil tegne *fra* og *til*:

```
\draw (0,2) -- (4,2);
\draw (0cm,1.5cm) -- (4cm,1.5cm);
\draw (0em, 1cm) -- (4em, 1cm);
\draw (0pt, 0.5cm) -- (4pt, 0.5cm);
```

1.3 Kurver

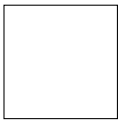
Vi bruker kontrollpunkter for å lage en kurvet linje. I eksempelet her, så starter vi i koordinatene `(-2,2)` og så tegner vi en kurve til første kontrollpunkt som er `(-1,0)`, så videre til `(1,0)`, og til slutt ender kurven opp i slutt-punktet som er `(2,2)`.



```
\draw (-2,2) .. controls (-1,0) and (1,0) .. (2,2);
```

1.4 Kvadrat

Vi kan bygge på linjen vår og lage et kvadrat:

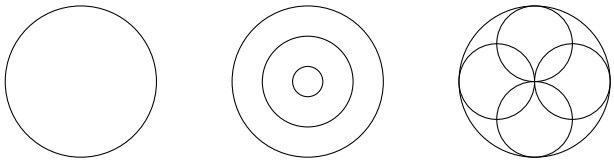


```
\draw (0,0) -- (1.5,0) -- (1.5,1,5) -- (0,1.5) -- (0,0);
```

Vi kan også bruke nøkkelordet `rectangle`, og lage en kortversjon som gjør akkurat det samme:

```
\draw (0,0) rectangle (1.5,1.5);
```

1.5 Sirkel



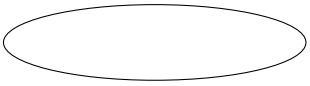
Den første koordinaten er sirkelens sentrum, og lengden vi oppgir til slutt er sirkelens radius.

```
\draw (0,0) circle (1cm);
```

Hvordan tegnes figurene ◎ og ⊗ ?

1.6 Ellipse

Ellipser tegnes ved at vi oppgir radiusen i x- og y-retningene:



```
\draw (0,0) ellipse (2cm and 0.5cm);
```

1.7 Buer

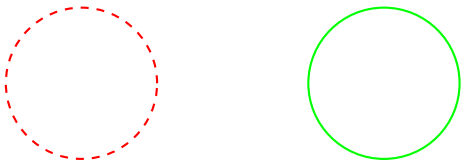
Buer (`arc`) skriver man på formen

```
\draw  (0,0) arc (0:180:1);
```

Hvor `(0,0)` er posisjonen. Og `(0:180:1)` betyr at vi skal tegne en bue fra 0 til 180 grader på en sirkel med radius 1.

1.8 Pynte litt

For å pynte litt på sirkelen vår, kan vi legge til noen ekstra argumenter til `\draw`-kommandoen. For eksempel slik:



```
\draw[red, thick, dashed] (2,2) circle (1cm);
\draw[green, thick] (6,2) circle (1cm);
```

1.9 Tykkelser

ultra thick
very thick
thick
thin
very thin
ultra thick

Man kan også definere egne tykkelser ved å bruke `line width`.

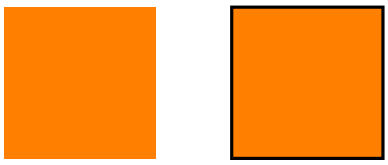
```
\draw[line width=10pt] (0,0) -- (2,0);
```

1.10 Farger

Figur 1: Farger i TikZ.

1.11 Fylle med farge

Vi kan også fylle formene våre ved å bruke kommandoen `\fill`. Ønsker vi å legge til en kant rundt kvadratet, kan vi bruke kommandoen `\filldraw`.



```
\fill[orange] (0,0) rectangle (2,2);
\filldraw[orange, draw=black, very thick] (3,0) rectangle (5,2);
```

1.11.1 Gradient

Vi har også gradient i TikZ, og det kan se slik ut:

```
\shade[left color=orange, right color=yellow]  (0,0) rectangle (2,2);
\shade[top color=orange, bottom color=yellow]  (3,0) rectangle (5,2);
\shade[inner color=orange, outer color=yellow] (6,0) rectangle (8,2);
```

Hvordan tegner vi dette ?

1.11.2 Blande farger

Vi kan også blande farger i Ti*k*Z. Her blander vi 50% blå og 50% orange med hverandre.

```
\fill[blue!50!orange] (6,0) rectangle (8,2);
```

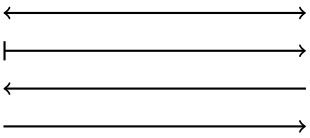
Når vi skriver

```
\fill[blue!50] (0,0) rectangle (2,2);
```

så blander vi 50% blå med 50% hvit.

1.12 Plotte funksjoner

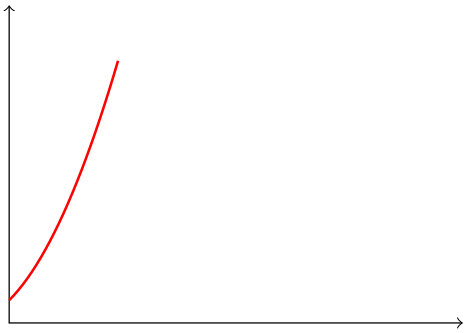
Man kan også plotte funksjoner i Ti*k*Z. Da er det kjekt å kjenne til de forskjellige typer piler.



Figur 2: Piler i TikZ.

```
\draw[<->] (0,1.5) -- (4,1.5);
\draw[|->] (0,1) -- (4,1);
\draw[<-]  (0,0.5) -- (4,0.5);
\draw[->]  (0,0) -- (4,0);
```

1.12.1 `plot`



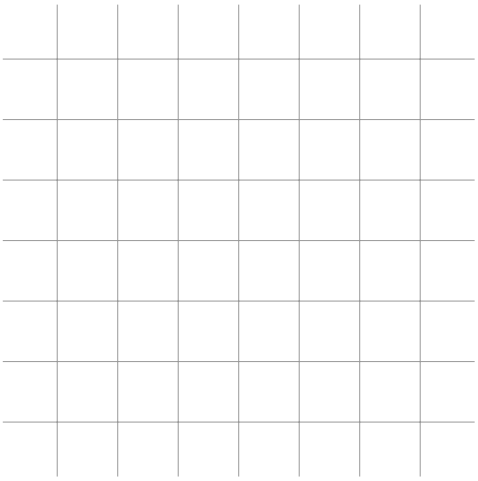
```
\begin{tikzpicture}
    \draw[<->] (0,3.5) -- (0,0) -- (5,0);
    \draw[red, thick, domain=0:1.2] plot (\x, {0.25+\x+\x*\x});
\end{tikzpicture}
```

`domain` er rekkevidden av x som blir plottet. I dette tilfellet plotter vi funksjonen $0.25 + x + x^2$. Legg merke til at det er parenteser rundt funksjonen som vi skal plotte `plot (\x, {function})`.

Hvordan kan vi plotte dette  ?

2 Koordinatsystem

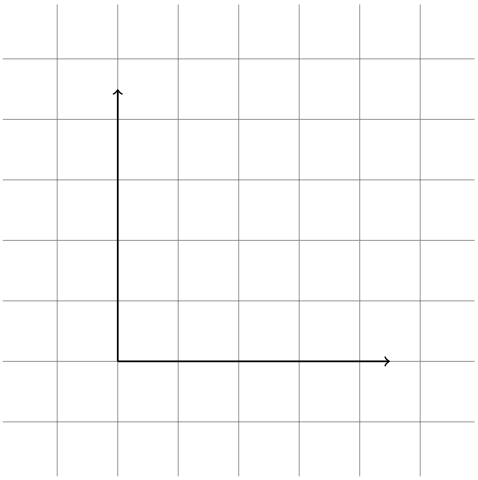
Dette eksempelet krever et rutenett, piler, noder og plassering av tall og bokstaver. Vi starter med et rutenett:



```
\draw[step=1cm,gray!80,very thin] (-1.9,-1.9) grid (5.9,5.9);
```

2.1 Akser

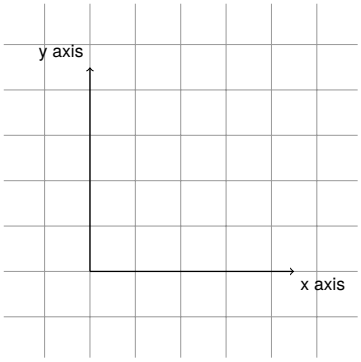
Videre trenger vi x-aksen og y-aksen. Dette er to linjer med piler i enden.



```
\draw[thick, ->] (0,0) -- (4.5,0);
\draw[thick, ->] (0,0) -- (0,4.5);
```

2.2 Noder

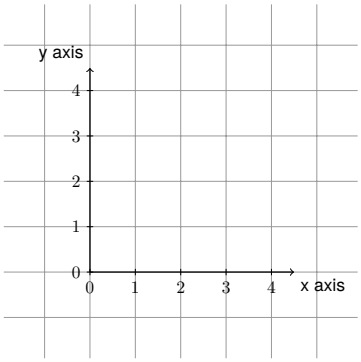
Vi kan legge på tekst (*label*) ved å bruke nøkkelordet `node`. Vi plasserer teksten ved linjene vi har tegnet ved å fortelle noden hvor vi vil ha den.




```
\draw[thick, ->] (0,0) -- (4.5,0) node[below right] {x axis};
\draw[thick, ->] (0,0) -- (0,4.5) node[above left] {y axis};
```

2.3 Løkker

Vi kan fortsette med tallene som skal gå langs aksene ved å bruke løkker:

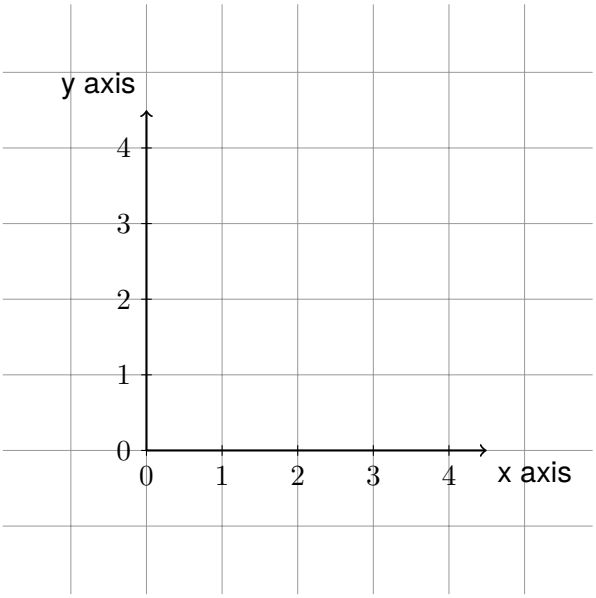


Denne løkken går over linjene vi allerede har tegnet, og setter en liten strek for hver centimeter. Og ved siden av linjen skriver vi et tall.

```
\foreach \x in {0,1,2,3,4}
  \draw (\x cm, 2pt) -- (\x cm, -2pt) node[below] {$\x$};
\foreach \y in {0,1,2,3,4}
  \draw (2pt, \y cm) -- (-2pt, \y cm) node[left]  {$\y$};
```

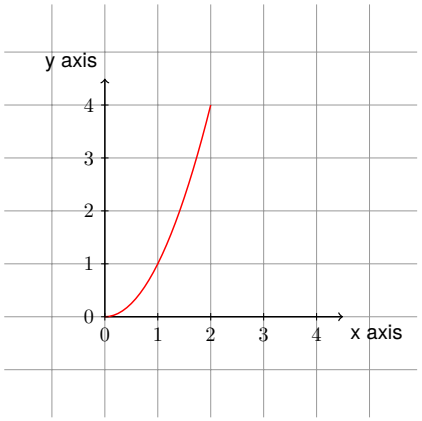
Hvordan kan vi bruke `foreach` til å tegne dette ●●●●●●●●●● ?

2.4 Hele koden for koordinatsystemet




```
\begin{tikzpicture}
  \draw[step=1cm,gray!80,very thin] (-1.9,-1.9) grid (5.9,5.9);
  \draw[thick, ->] (0,0) -- (4.5,0) node[below right] {x axis};
  \draw[thick, ->] (0,0) -- (0,4.5) node[above left] {y axis};

  \foreach \x in {0,1,2,3,4}
    \draw (\x cm, 2pt) -- (\x cm, -2pt) node[below] {$\x$};
  \foreach \y in {0,1,2,3,4}
    \draw (2pt, \y cm) -- (-2pt, \y cm) node[left] {$\y$};
\end{tikzpicture}
```

Hvordan kan vi tegne dette?



3 Trær

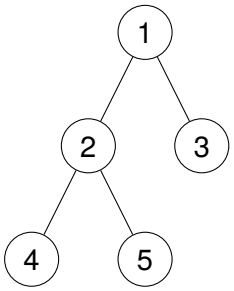
Et tre består av en rekke noder. Når vi tegner trær i Ti*k*Z starter vi med å definere rot-noden. Legg merke til attributtene vi git `tikzpicture`. Her sier vi at `every node` skal ha *stilen* (`.style`) sirkel med sort strek.


```
\begin{tikzpicture}[every node/.style={circle, draw=black}]
    \node {1};
\end{tikzpicture}
```

3.1 Bygge treet

Treet bygger vi ved å legge til barna. Barna skrives på formen:

`child { node[opt.] {value} }`

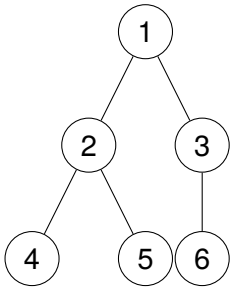



```
\node {1}
    child { node {2}
        child { node {4} }
        child { node {5} }
    }
    child { node {3} }
;
```

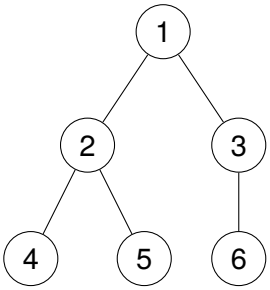
3.2 Justere avstand mellom noder

Når vi nå vil bygge videre og legge til tallet 6 under `child {node {3}}` vil vi overlappe 5. Da trenger vi å justere avstanden mellom søsken-noder.

Vi har:



Vi vil ha:



Da legger vi på et attributt til i listen til `tikzpicture` som forteller noe om avstanden mellom nodene.

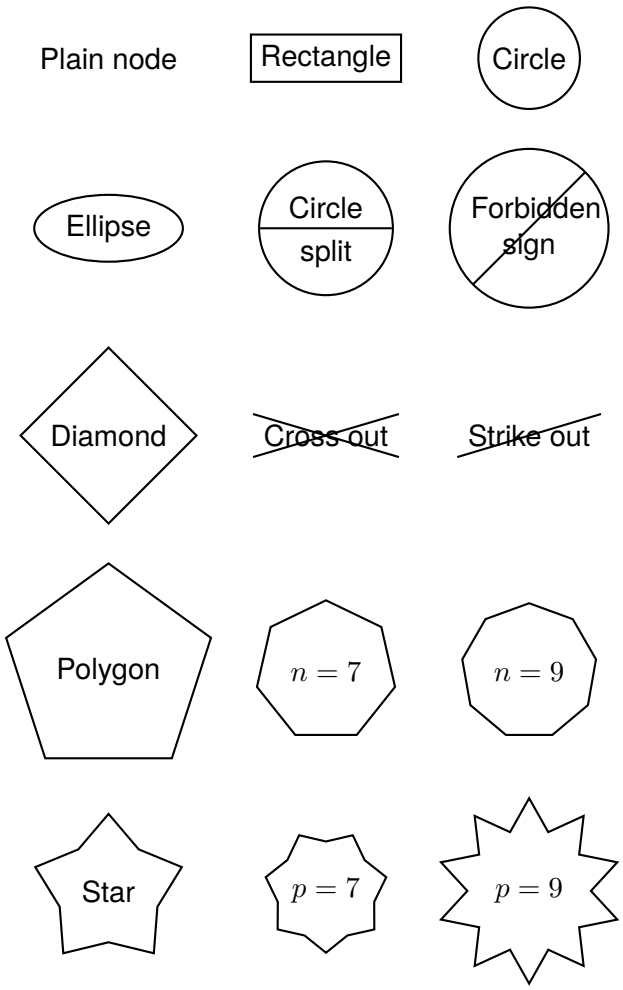
```
\begin{tikzpicture}[every node/.style={circle, draw=black},
                    level 1/.style={sibling distance=20mm},
                    level 2/.style={sibling distance=15mm}]

\node {1}
    child { node {2}
        child { node {4} }
        child { node {5} }
    }
    child { node {3}
        child {node {6} }
    }
;
\end{tikzpicture}
```

Her forteller vi at stilen til nodene på `level 1` skal være at de har avstand til sine søsken med 20 mm, og 15 mm for `level 2`. Vi kunne også lagt til attributtet `level distance` for å få større eller mindre avstand mellom lagene.

3.3 Former som noder kan ha

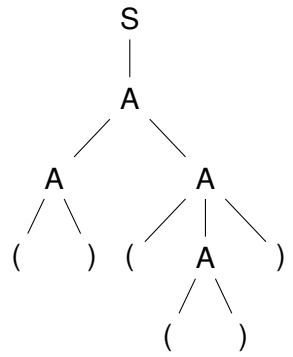
Man kan få forskjellige fasonger på noder ved å inkludere `\usetikzlibrary{shapes}`. Her er en oversikt over forskjellige fasonger en node kan ha. For å få ønsket fasong skriver man noden på denne formen:

```
\node[rectangle] {Rectangle};
\node[regular polygon, regular polygon sides=5] {n=5};
\node[circle split] {Circle \nodepart{lower} split};
```



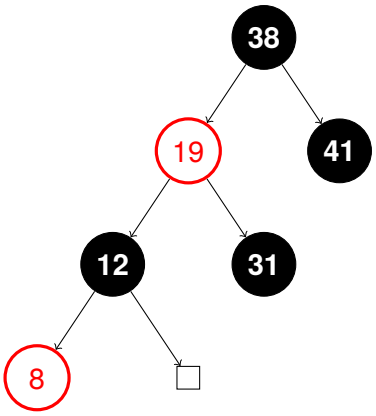
Figur 3: Fasonger på noder.

3.4 Eksempel på et tre med avstander




```
\begin{tikzpicture}[every node/.style={},
                    level 2/.style={sibling distance=20mm},
                    level 3/.style={sibling distance=10mm},
                    level distance=30pt]
\node {S}
    child { node{A}
        child { node {A}
            child { node {(} }
            child { node {)} }
        }
        child { node {A}
            child { node {(} }
            child { node {A}
                child { node {(} }
                child { node {)} }
            }
            child { node {)} }
        }
    }
;
\end{tikzpicture}
```

3.5 Rød-svarte trær



Å tegne trær på denne måten krever ingen tilleggsbiblioteker fra Ti*k*Z. Dette er et eksempel på tegning med egendefinerte noder. Dette gjør vi via `tikzset`, her kan vi gi stilen de forskjellige typer noder.

```
\tikzset{
   treenode/.style = {align=center},

   % Sorte noder
   node_black/.style = {treenode, circle, white,
                        font=\bfseries, draw=black,
                        fill=black, text width=0.8cm},
   % Røde noder
   node_red/.style = {treenode, circle, red, draw=red,
                      text width=0.8cm, very thick},
   % Null-pekere
   node_null/.style = {treenode, rectangle, draw=black,
                       minimum width=0.3cm, minimum height=0.3cm}
}
```

Starter med å definere `treenode`, som er felles for alle nodene. Røde og sorte noder tegnes som `circle`, hvor sorte noder har `fill=black` og tekstfarge `white`, mens røde noder har rødt omriss med `draw=red`, og tekstfarge `red`. Null-nodene sier vi skal være sorte `rectangle`. Tegnes som små kvadrater på 0.3 cm $\times$ 0.3 cm.

3.6 Bygge det rød-svarte treet

```
\begin{tikzpicture}[->,level/.style={ sibling distance = 2cm,
                    level distance = 1.5cm }]
\node [node_black] {38}
    child { node [node_red] {19}
        child { node [node_black] {12}
             child { node [node_red] {8} }
             child { node [node_null] {} }
        }
        child { node [node_black] {31} }
    }
    child { node [node_black] {41} }
;
\end{tikzpicture}
```

Setter forskjellige opsjoner med:

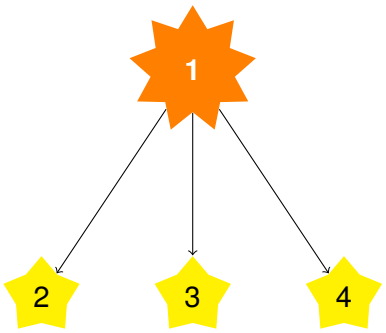
```
\begin{tikzpicture}[->,level/.style={ sibling distance = 2cm,
                    level distance = 1.5cm }]
```

Her sier vi at treet skal tegnes med piler (`->`), og at stilen (`.style`) for distansen mellom søskennoder skal være 2 cm, og distansen mellom barn og foreldre skal være 1.5 cm.

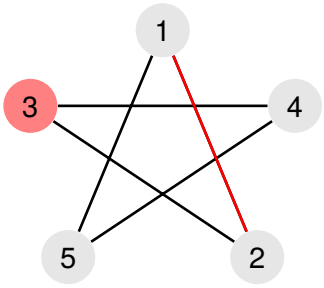
Videre så forteller vi barna i treet hva slags node de skal være.

```
child { node [node_red]   {x} }
child { node [node_black] {y} }
child { node [node_null]  {z} }
```

Hvordan kan vi tegne dette treet?



4 Grafer



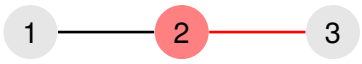
Det fins enklere måter å tegne grafer på enn dette, men jeg syns denne måten er fin. Den krever heller ingen andre biblioteker eller pakker enn Ti*k*Z selv.

Vi starter med å definere de forskjellige elementene til en graf.

```
\begin{tikzpicture}
    \tikzstyle{vertex} = [circle,fill=black!10]
    \tikzstyle{selected vertex} = [vertex, fill=red!50]

    \tikzstyle{selected edge} = [draw,line width=1pt,-,red!100]
    \tikzstyle{edge} = [-,black,line width=1pt]
\end{tikzpicture}
```

Her forteller vi at `vertex`er (eller noder), skal være sirkler. Markerte noder skal være fylt med rød farge.

Kanter skal tegnes som sorte linjer (`[-, black ...]`). Og markerte kanter skal være røde.



4.1 Tegne grafen

For å plassere nodene rundt om på arket sier man hvor man vil de skal være ved hjelp at koordinater.

```
\begin{tikzpicture}
    \tikzstyle{vertex}          = [circle,fill=black!10]
    \tikzstyle{selected vertex} = [vertex, fill=red!50]

    \tikzstyle{selected edge}   = [draw,line width=1pt,-,red!100]
    \tikzstyle{edge}            = [-,black,line width=1pt]

    \node[vertex]          (v1) at (1.25,1.7) {1};
    \node[vertex]          (v2) at (1.5,1.1)  {2};
    \node[selected vertex] (v3) at (0.9,1.5)  {3};
    \node[vertex]          (v4) at (1.6,1.5)  {4};
    \node[vertex]          (v5) at (1,1.1)    {5};

    \draw[edge]          (v1)--(v2)--(v3)--(v4)--(v5)--(v1);
    \draw[selected edge] (v1)--(v2);
\end{tikzpicture}
```

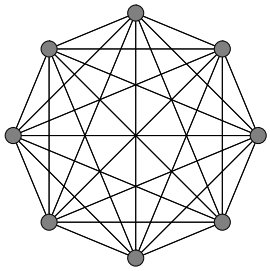
Nodene defineres ved å først bruke nøkkelordet `node`, så fortelle hvilken type node dette er. I dette tilfellet, så er det enten `vertex` eller `selected vertex` som vi har definert med `tikzstyle`. Nodens navn bruker man kun i egen kode, når vi skal tegne opp kantene trenger vi disse navnene. Koordinatene `(x,y)` forteller hvor vi vil plassere noden, og verdien er innholdet i noden.

```
\node[type of node] (node name) at (x,y) {value};
```

Kantene tegnes likt som linjer fra seksjon 1. Men her gir vi nøkkelordet `draw` en av to stiler, som vi definerte med `tikzstyle`. Enten `edge` eller `selected egde`.

```
\draw[type of edge] (from node) -- (to node);
```

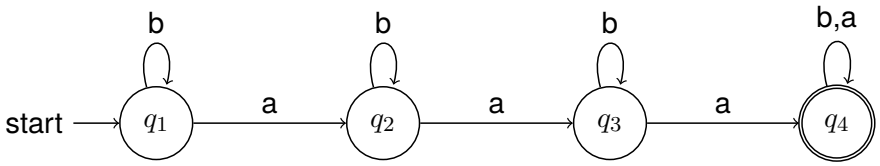
Hvordan kan vi tegne denne?



5 Automater

Denne måten å tegne automater på krever at man inkluderer

```
\usetikzlibrary{automata}
```




```
\begin{tikzpicture}[->,auto,node distance=3cm,line width=0.2mm]
  \node[initial,state   (A)                  {$q_1$};
  \node[state]          (B) [right of=A]     {$q_2$};
  \node[state]          (C) [right of=B]     {$q_3$};
  \node[state,accepting](D) [right of=C]     {$q_4$};

  \path (A) edge [loop above] node           {b}   (A)
            edge node                        {a}   (B)
        (B) edge [loop above] node           {b}   (B)
            edge node                        {a}   (C)
        (C) edge [loop above] node           {b}   (C)
            edge node                        {a}   (D)
        (D) edge [loop above] node           {b,a} (D);
\end{tikzpicture}
```

For denne måten å tegne automater på, så settes alle attributter som beskriver automaten i definisjonen til `tikzpicture`. Her har automaten følgende egenskaper:

```
{tikzpicture}[->, auto, node distance=3cm, line width=0.2mm]
```

Dette forteller oss at automaten skal tegnes med piler (`->`), nodene skal ha avstand på 3 cm, og linjene en tykkelse på 0,2 mm. Auto stiller teksten *over* linjene, i stedet for *på* linjene.

5.1 Automatens tilstander

En automat har tre typer tilstander: starttilstanden, vanlig tilstand(er), og akepterende tilstand(er).

```
\node[state] (node name) {state name};
```

I tillegg til `[state]`, så kan man ha med opsjonen `[initial, state]` for starttilstanden, eller `[state, accepting]` for aksepterende tilstand.

5.2 Stien gjennom automaten

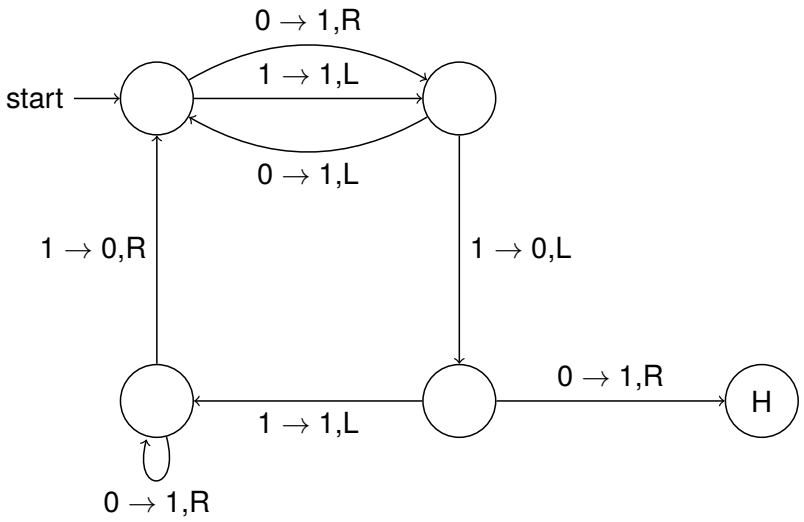
Stien tegnes gjennom en `path`. Denne konstrueres på følgende vis:

```
\path (from state) edge [opt.] node {weight} (to state)
```

Her kan `[opt]` være `loop above/below`, `bend left/right`.

Flittig bever

Her er en flittig 4-bever. Denne automaten dekker de fleste opsjoner.

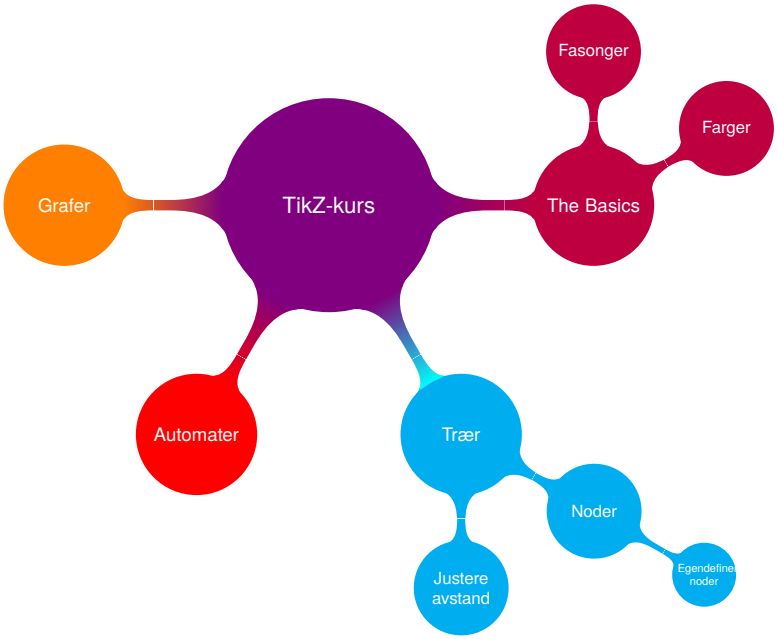



```
\begin{tikzpicture}[->,auto,node distance=4cm,line width=0.2mm]
  \node[initial,state] (A)              {};
  \node[state]         (B) [below of=A] {};
  \node[state]         (C) [right of=A] {};
  \node[state]         (D) [below of=C] {};
  \node[state]         (E) [right of=D] {H};

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

6 Forskjellige Ti*k*Z biblioteker

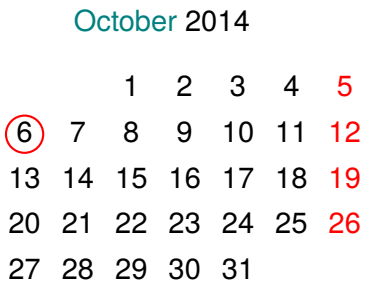
Som med automatene, er det flere andre Ti*k*Z-biblioteker som kan inkluderes. Her kommer noen eksempler.

6.1 `mindmap`




```
\path[mindmap,concept color=violet,text=white]
    node[concept] {TikZ-kurs}
    [clockwise from=0]
    child[concept color=purple] {
    node[concept] {The Basics} [clockwise from=90]
        child { node[concept] {Fasonger} }
        child { node[concept] {Farger} }
    }
    child[concept color=cyan] {
    node[concept] {Trær} [clockwise from=-20]
        child { node[concept] {Noder}
            child { node[concept] {Egendefinerte noder}}
        }
        child { node[concept] {Justere avstand} }
     }
    child[concept color=red] { node[concept] {Automater} }
    child[concept color=orange] { node[concept] {Grafer} };
```

6.2 `calendar`




```
\begin{tikzpicture}
    \calendar (mycalendar) [dates=2014-10-01 to 2014-10-31,week list,
                            month label above centered]
                            month text=\textcolor{teal}{\%mt} \%y-]
                            if (Sunday) [red]
                            if (equals=2014-10-06)
                               {\draw[red,thick] (0,0) circle (7pt);};
\end{tikzpicture}
```

Attributter Attributter som vi gir kalenderen `mycalendar` er at den skal strekke seg fra 1. oktober 2014 til 31. oktober 2014. Den skal tegnes som lister av uker, og månedens navn skal skrives på toppen, sentrert. Vi sier også at tekstfargen til måneden skal være *teal*, og at vi skal legge på året.

If-setninger Her har vi også et eksempel på `if`-setninger i Ti*k*Z. Disse er på formen

```
if=(<condition>)<code or options> else<else code or options>
```

I dette eksempelet sier vi at *hvis* dagen er en søndag, så skal teksten være rød. Og hvis datoen er 6. oktober 2014, så skal vi tegne en rød ring rundt denne.

7 Circuitikz

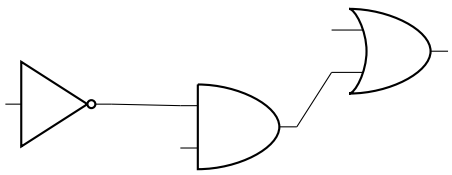
Noe som er kjekt å vite om er også logiske porter i Circuitikz. Dette får du ved å inkludere pakken:

```
\usepackage{circuitikz}
```

Siden dette *ikke* er Ti*k*Z jobber vi ikke i miljøet `tikzpicture`, men i miljøet `circuitikz`.

```
\begin{circuitikz} \draw
    <kode her>
\end{circuitikz}
```

7.1 Eksempel på en liten krets



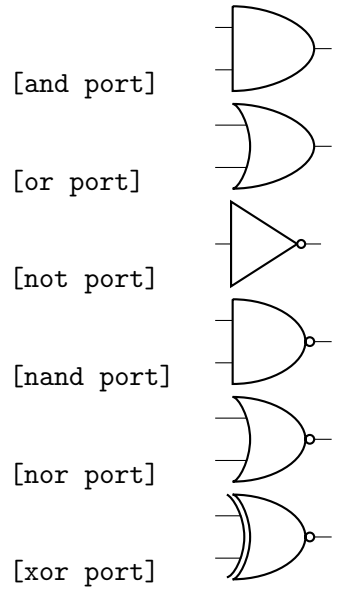
```
\begin{circuitikz} \draw
    (-3,0.3) node[not port] (not) {}
    (0,0)    node[and port] (and) {}
    (2,1)    node[or port]  (or)  {}

    (not.out) -- (and.in 1)
    (and.out) -- (or.in 2);
\end{circuitikz}
```

Det fungerer på samme måte som når vi tegner noder i TikZ. Vi starter med koordinatene, så definerer vi hva slags node (port) vi vil ha, og til slutt en evt. merkelapp.

```
(x,y) node [what kind of port] (name of port) {label}
```

Portens navn er valgfritt, og brukes kun i din egen kode.

7.2 Oversikt over forskjellige porter i Circuitikz



Figur 4: Forskjellige porter i Circuitikz

8 Ressurser

Gøyale eksempler

- Enderman
- Dartboard
- India map

8.1 Nyttige lenker

- A Ti*k*Z tutorial: Generating graphics in the spirit of TEX
- Ti*k*Z & PGF Manual
- Graphics with Ti*k*Z
- TeXample.net
- TEXUsers Group (tug.org)

Visste du at..

Roger Antonsens bok «Logiske Metoder» er full av Ti*k*Z/PGF?