Kapitel 14

Gestaltung von Diagrammen

14.1 Überblick, Arten von Diagrammen

In den vorangegangenen Kapiteln haben wir die Funktionalität von MP kennengelernt und an einfachen Beispielen gezeigt, wie man die einzelnen Befehle und Konstrukte anwendet. Wir wollen jetzt etwas innehalten und uns einen Überblick verschaffen, welche Arten von Diagrammen in der Praxis am häufigsten verwendet werden und welche Gesichtspunkte für deren Gestaltung beachtet werden sollten, um qualitativ hochwertige Diagramme zu gestalten.

Die Auswertung der Abbildungen in einer renommierten Fachzeitschrift im technisch-wissenschaftlichen Bereich [1] ergab beispielhaft folgendes Ergebnis.
Die Grafiken können im wesentlichen folgenden Gruppen zugeordnet werden:

1. Funktionsverläufe, Kurvendiagramme, Kurvenscharen. Darstellung von gemessenen und berechneten funktionellen Abhängigkeiten.
2. Schematische Skizzen von Versuchsanordnungen und untersuchten Objekten mit individueller Gestaltung.
3. Fotos, Oszillogramme und Graubilder, teilweise mit Beschriftung als Overlay.
4. Balkendiagramme, Histogramme.
5. Einfache perspektivische dreidimensionale Darstellungen.

6. Box- oder Blockdiagramme, Strukturdiagramme, Blockschaltbilder.
7. Statistische Daten: Box-Plot, Scatter-Plot, Regression, ROC-Kurve.
8. Elektrische Schaltbilder

Von 355 Abbildungen entfielen ca. 90% auf die vier Kategorien:
— Kurvendiagramme, 56%
— Versuchsanordnungen, 16%
— Fotos, 10%
— Balkendiagramme, 6%
Die restlichen ca. 10% verteilen sich auf:
— einfache perspektivische Darstellungen, 3.7%
— Statistische Box-Plot, 2.8%
— Box-Diagramme, 2.5%
— elektrische Schaltbilder, 1.7%
— Scatter-Plot, 0.5%.

Mit der Erstellung von Kurvendiagrammen, Balkendiagrammen, Box-Plots und Scatter-Plots mit Hilfe des Graph-Pakets haben wir uns bereits ausführlich beschäftigt. Die Grafiken für Versuchsanordnungen erfordern individuelle Lösungen, die nicht allgemeingültig behandelt werden können. Verbleiben Box-Diagramme, Fotos mit Overlay und perspektivische Darstellungen. Obwohl in dem betrachteten Fall, nur einfache dreidimensionale Darstellungen auftraten, spielt die perspektivische Darstellung von Körpern in anderen Anwendungsgebieten wie z. B. Architektur und Maschinenbau eine wichtige Rolle und wird daher eigens in Kapitel 16 behandelt. Box-Diagramme, Fotos mit Overlay und viele andere praktische Konstrukte findet man in Kapitel 15.

14.2 Gestaltung

Sehr lesenswert sind in diesem Zusammenhang die exzellenten Bücher von Edward R. Tufte [40] und William S. Cleveland [9], die zahlreiche Ratschläge für die Gestaltung grafischer Darstellungen enthalten. Insbesondere das erstgenannte diskutiert zahlreiche Beispiele, historische und heutige, für gut und schlecht gestaltete Diagramme, es analysiert die Merkmale, die eine gute Grafik ausmachen, und gibt Empfehlungen, die man bei der Gestaltung unbedingt beachten sollte.

Diagramme sind eine bildhafte Veranschaulichung von meist sehr umfangreichen Daten. Es kommt also darauf an, die Datenmenge so darzustellen, dass dem Betrachter in kürzester Zeit und auf engstem Raum eine Idee und Vorstellung des zugrundeliegenden Sachverhalts vermittelt wird. Klar, präzise und effizient. Man konzentriere sich auf die Daten und vermeide ablenkende und störende Elemente.

Die Darstellung muss den Daten entsprechen, d. h. eindimensionale Daten müssen durch eindimensionale grafische Elemente dargestellt werden. Ein reiner Zahlenwert wird also am besten durch eine gerade Linie entsprechender Länge repräsentiert. Bei einem

Tortendiagramm, Kreisdiagramm (*pie chart*) wird ein Zahlenwert durch ein Kreissegment, also ein zweidimensionales Flächenstück dargestellt. Deshalb ist der Betrachter verwirrt und außerstande der Grafik die dargestellten Prozentwerte korrekt zu entnehmen. Jacques Bertin [6] bringt es auf den Punkt

> "*pie charts are completely useless*".

Beispiel: Versuchen Sie, aus dem Tortendiagramm die prozentualen Anteile abzulesen. Zum Vergleich habe ich die Daten adäquat als Balkendiagramm dargestellt. Die Werte können mühelos auf 1 bis 3 gültige Ziffern genau entnommen werden.

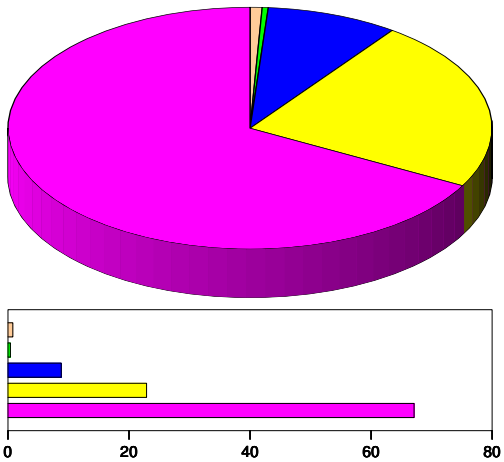


Bild 14.1: Tortendiagramm vs. Balkendiagramm.

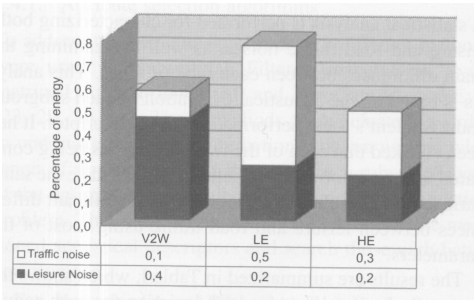
Die Zahl der informationstragenden Dimensionen in der Grafik und in den Daten sollte übereinstimmen, sonst kann der Betrachter der Grafik die ihn interessierenden Werte nicht korrekt entnehmen.

Trotz dieser Aussage sind Tortendiagramme in der Praxis weit verbreitet vor allem im medialen und Finanzbereich. Wenn man daher mit MP solche Diagramme mit hoher Qualität zeichnen will, sei auf das sehr leistungsfähige Paket piechartmp [31] von Jens-Uwe Morawski verwiesen, das auch in der TEX Live-Distribution enthalten ist.

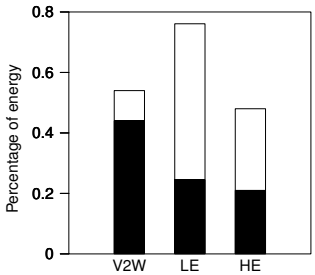
Auch perspektivische Darstellungen von eindimensionalen Daten sind problematisch, weil bei gleich großen Zahlen, die in der Ferne liegenden kleiner dargestellt sind als jene im Vordergrund, wodurch der falsche Eindruck eines Preisanstiegs entsteht.

Beispiel:
Die Grafik zeigt 6 Zahlenwerte als räumliches Säulendiagramm und zusätzlicher tabellarischer Angabe der Werte, die jedoch nicht ganz mit der Grafik übereinstimmen. Trotz vieler Gitterlinien ist es nicht möglich, die interessierende Information korrekt zu entnehmen.

Auch hier habe ich zum Vergleich ein entsprechendes Balkendiagramm gezeichnet, dem die Daten klar und deutlich ohne Probleme entnommen werden können.



(a)



(b)

Bild 14.2: Balkendiagramm: (a) schlechtes Beispiel, (b) gutes Design.

Die gezeigten Beispiele lassen bereits erahnen, dass es wohl darauf ankommt, die Daten mit möglichst wenig grafischem Aufwand darzustellen, was Tufte dazu veranlasste, ein Datentinteverhältnis (*data-ink ratio*) zu definieren, das es zu maximieren gilt

$$\text{data-ink ratio} = \frac{\text{data-ink}}{\text{total ink to print the data}}$$

Mit anderen Worten, der Tinteverbrauch zur Darstellung der Daten sollte möglichst groß sein im Vergleich zu jenem für nicht datenrelevante Elemente. Diesen Grundsatz haben Ludwig Mies van der Rohe und Robert Venturi in folgende Worte gefasst

> "*For non-data ink, less is more,*
> *for data-ink, less is bore.*"

oder auf deutsch: „weniger ist oft mehr".

Dies ist ein **minimalistischer** Ansatz, man halte die Grafik so einfach wie möglich, vermeide unnötigen Zierrat und nicht datenrelevante Linien.

Beispiel:
Box-Plots. Tufte stellt der klassischen Form nach Tukey [41] eine Variante gegenüber, die den Median als Punkt und die Whisker als Linien darstellt und verzichten auf die rechteckige Box, die keine Information trägt. Der Datentintefaktor erreicht dadurch seinen maximalen Wert Eins.

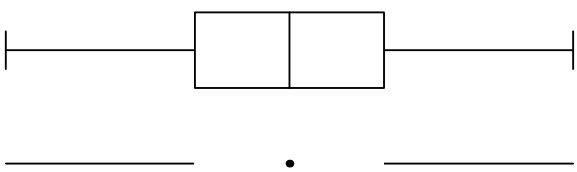


Bild 14.3: Minimalistischer Entwurf eines Boxplots.

Dies ist vielleicht ein extremes Beispiel, aber es soll den Grundgedanken der Maximierung des Datentintefaktors verdeutlichen. Man sollte alles überflüssige weglassen

und die Daten in der einfachsten Form präsentieren. Dies betrifft insbesondere Ausschmückungen und Dekorationen (*chartjunk*) wie Schraffuren, Schatten, Doppellinien, abgerundete Ecken, Farben und Gitterlinien. Schraffuren erzeugen im Auge des Betrachters Vibrationen durch Moiré-Effekte, die wehtun und die Wahrnehmung der Daten vernebeln. Schraffuren und Farben sollte man vermeiden und falls nötig durch abgestufte Grauwerte ersetzen. Beim Einsatz von Farben sollte man auch an die 5–10% Mitmenschen denken, die farbenblind sind und die am ehesten noch die Farbe blau erkennen können, rot und grün scheiden aus. Gitter enthalten keine Dateninformation und sollten nur in begründeten Fällen zur Erleichterung des Ablesens von Datenwerten eingesetzt werden, aber auch dann nur wenige dünne Linien in einem dezenten Grauton.

Die Beschriftung ist ein wichtiger Bestandteil einer Grafik. Wir haben dies bereits in Kapitel 4 diskutiert. Schriftfont und -größe sollten zum Text passen, in den die Grafik eingefügt wird. Bei umfangreicher Beschriftung sind Serifen-Schriften zu bevorzugen, da die Serifen das Auge führen und so das Lesen erleichtern. Der Schriftgrad im Diagramm kann gegenüber dem Text um ca. 2 pt kleiner gewählt werden. Die Beschriftung sollte vorzugsweise waagrecht angeordnet werden, Abkürzungen und reine Großschreibung sollte man vermeiden. Die Regeln der Typografie insbesondere für mathematische Größen und Einheiten müssen beachtet werden.

Ein mehr ästhetischer Gesichtspunkt betrifft die Wahl eines geeigneten Formats für die Grafik. Die Breite liegt, wie wir sahen, durch die Spaltenbreite des Seitenlayouts mit ca. 8 cm fest. Die Höhe einer Grafik wird natürlich in erster Linie durch die darzustellenden Daten bestimmt, wobei durch Skalierung eine gewisse Variationsmöglichkeit besteht. Ansonsten sollte die Grafik mehr breit als hoch gezeichnet werden, weil das Auge ein Bild im Querformat leichter überblicken kann. In der Praxis entsprechen über 90% der Diagramme dieser Vorstellung mit einem Seitenverhältnis von 1:1 (quadratisch), $\sqrt{2}$:1 (DIN-Format quer), $(\sqrt{5}+1)/2 = 1.62$:1 (Goldener Schnitt) bis 2:1 (zwei Quadrate nebeneinander). Eine gute Wahl ist beispielsweise bei einer Breite von 8 cm eine Höhe von 5 cm.

Bei Grafiken mit Beschriftung wird die Größe der grafischen Elemente teilweise oder überwiegend durch die Länge der Texte bestimmt, Grafiken ohne Beschriftung können dagegen extrem verkleinert werden, wenn man bedenkt, dass das menschliche Auge noch Linien von 0.1 mm Abstand unterscheiden kann. Ein Maß für die Dichte der Information pro Fläche ist

$$\text{Datendichte der Grafik} = \frac{\text{Zahl der Elemente der Datenmatrix}}{\text{Fläche der Grafik}}$$

Dieses Maß variiert in der Praxis in einem riesigen Bereich von 0.02 bis 20 Daten/cm^2 und mehr. Die Auflösung des Auges ermöglicht theoretisch eine Dichte von 10000 Daten pro cm^2 (ca. 250 dpi).

Tabelle 14.1 zeigt als Beispiel den Verlauf eines Börsenkurses [4] in einer von Tufte entwickelten miniaturisierten Darstellung als Zeitreihendiagramm oder Wortgrafik (*sparkline*) [40], die in eine Textzeile passt. Zur Orientierung werden Startwert, Endwert, Maximum und Minimum als Zahlenwerte hinzugefügt.

Tabelle 14.1: Beispiel für die Integration einer Sparkline in den Text.

	7. Februar 2006			Minimum	Maximum
Dow Jones	10765.45		10732.63	10723.67	10798.30

Zum Entwurf von qualitativ hochwertigen Grafiken gibt es viele Optionen und der Designer wird daraus die jeweils geeignetsten für den vorliegenden Fall auswählen. Die beschriebenen Regeln sind nicht als starr zu verstehen, oft ist es auch eine Frage des persönlichen Geschmacks und des ästhetischen Empfindens, wie man eine Grafik gestaltet. Man benötigt etwas Erfahrung und Übung, aber bei allem Tun sollte man sich stets von der Grundidee eines minimalistischen Entwurfstils leiten lassen, oder um es mit Goethe zu sagen

> „In der Beschränkung zeigt sich erst der Meister."

Kapitel 15

Anwendungen und Beispiele

15.1 Struktur-, Block- und Flussdiagramme

15.1.1 Strukturierter Entwurf

Blockdiagramme bestehen im wesentlichen aus Boxen (Rechtecken), die eine Beschriftung enthalten (Textboxen) und die durch Flusslinien miteinander verbunden sind. Aufgrund der hervorragenden Parametrisierung von Pfaden in MP ist es sehr einfach, die Mitte, die Ecken und jeden Punkt auf dem Rand von Boxen zu identifizieren und miteinander zu verbinden. Wir werden auch die Eigenschaft der Überdeckung nutzen und die Verbindungslinien zuerst zeichnen und dann die Boxen mit `unfill`, `draw` und `label` darüberlegen, sodass die Verbindungslinien die Boxen zunächst beliebig durchkreuzen dürfen. Wichtig ist aber ein systematisches Vorgehen und eine gewisse Planung und Strukturierung des Diagramms. Es empfiehlt sich stets vorab auf kariertem Papier die prinzipielle Anordnung der Boxen und der Verbindungslinien zu skizzieren, um unnötige Überkreuzungen zu vermeiden und eine übersichtliche und harmonische Struktur

zu erzielen.
Die Schritte im einzelnen:

1. Festlegung einer Zeicheneinheit `ut`, z. B. `ut:=1mm`. Alle Variablen für Abmessungen, außer Boxhöhen, gibt man nun als Vielfache von `ut` an, z. B. `a:=20ut`, sodass das gesamte Diagramm durch Änderung von `ut` skaliert werden kann.

2. Festlegung der Abmessungen der Textboxen. Diese richten sich im wesentlichen nach dem Text der vorgesehenen Beschriftung, einzeilig, zweizeilig, allgemein n-zeilig. Bei einer 8pt-Schrift sind geeignete Boxhöhen: $h_1 = 7\,\text{mm}$, $h_2 = 10.5\,\text{mm}$, $h_3 = 14\,\text{mm}$, allgemein $h_n = 3.5(n+1)\,\text{mm}$, die stets in **absoluten** Einheiten festgelegt werden, da sich die Schrift beim Skalieren nicht ändert. Mehrzeiliger Text wird in einer tabular-Umgebung gesetzt. Die Breiten w_i der Boxen richten sich nach der Breite des jeweiligen Textes, wobei die Breite eines Buchstabens im Mittel ca. 1.5 mm beträgt. Dabei sollte man sich aber auf wenige Standard-Boxbreiten beschränken, um später ein harmonisches Erscheinungsbild zu erreichen.

3. Im Prinzip ordnet man die Boxen in einer matrix- oder tabellenartigen Struktur von Zeilen und Spalten an. Dieser Grundgedanke sollte auch bei heterogenen Anordnungen stets als Leitlinie dienen. Zur Orientierung nummerieren wir die Boxen fortlaufend durch.

4. Alle Boxentexte formatieren wir als Picture, eventuell mit einem Switch in mehreren Sprachen.

```
picture lab[];
lab1:=btex AV-Dis etex;
...
```

5. Alle Boxen definiert wir als geschlossene Pfade

```
path p[];
p1:=unitsquare xscaled w1 yscaled h1 shifted (W,H);
...
```

   Die Anordnung der Rechtecke mit dem Referenzpunkt in der linken unteren Ecke erfolgt rekursiv mit den Variablen W und H, beginnend mit `H:=0; W:=0;` (linke, obere Ecke der Boundingbox des Diagramms). Als Abstandsmaß zwischen den Boxen legt man eine Variable d fest, z. B. `d:=2.5ut`.

```
...
H:=H-d-h[i];
W:=0; W:=W+w[i]+d;...
...
```

6. Die Flusslinien (Verbindungslinien) werden zuerst gezeichnet, auch quer durch Boxen, orientiert an `center p[i]` oder `point t of p[i]`.

```
...
draw center p2--center p4;
```

```
...
```

7. Dann werden die Boxen gezeichnet. Das Innere der Boxen wird mit `unfill` gelöscht, der Boxpfad p_i gezeichnet und die Beschriftung mit dem label-Befehl eingefügt.

```
for i=1 upto n:
  unfill p[i];
  draw p[i];
  label(lab[i],center p[i]);
endfor
```

Bei konsequenter Anwendung dieser Entwurfsschritte bleibt das Flussdiagramm stets konsistent und nachträgliche Änderungen sind einfach möglich.

Die folgenden Flussdiagramme hat in dankenswerter Weise Andreas Entenmann zur Verfügung gestellt.

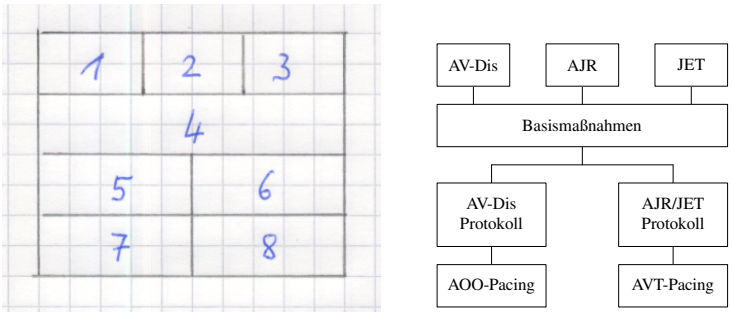
Wir wollen die Vorgehensweise an einem einfachen Beispiel gemäß Bild 15.1 demonstrieren. Zuerst skizzieren wir das Diagramm als tabellenartige Struktur und nummerieren die Boxen. Die weiteren Schritte zeigt das Listing. Wir wählen zunächst ut = 1 mm und die größte Breite gleich der Spaltenbreite $w_2 = 80$ut. Nach Fertigstellung des Diagramms, skaliert man mit `ut` das Diagramm soweit, dass die Texte gut in die Boxen passen, hier ut = 0.6 mm. Die Boxhöhen werden immer in absoluten Einheiten (mm) angegeben, da sich die Schrift beim Skalieren nicht verändert. Es folgen die Boxentexte. Soll das Diagramm wahlweise auf deutsch oder englisch beschriftet werden, kann man den Textblock wiederholen und die Texte übersetzen. Beispiel:

```
boolean deutsch;
deutsch:=true;
% deutsch:=false;
if deutsch:
  ...
  lab4:=btex Basisma\ss{}nahmen etex
  ...
else:
  ...
  lab4:=btex basic measures etex;
  ...
fi
```

Durch Auskommentieren von einer der beiden Anweisungen `deutsch:=...` wird das Diagramm in der einen oder anderen Sprache beschriftet.

Die Definition der Boxpfade erfolgt rekursiv beginnend in der linken oberen Ecke als Nullpunkt. Da der Abstand sowie die Breiten und Höhen der Boxen bereits festgelegt sind, ist die Dimensionierung und Platzierung sehr einfach. Bei `unitsquare` liegt der Referenzpunkt in der linken unteren Ecke. Beim Zeichnen der Verbindungslinien orientiert man sich an den Mittelpunkten und den Randpunkten der Boxen. Die Linien dürfen auch quer durch Boxen verlaufen. Um Tipparbeit zu sparen wird man bestimmte Punkte

auf z-Variable abspeichern, sodass man sehr einfach auf deren x- und y-Komponenten zugreifen kann. Schließlich werden alle Boxpfade überdeckend gezeichnet und die Beschriftungen eingefügt.



(a) (b)

Bild 15.1: Beispiel Boxdiagramm. (a) Skizze der tabellarischen Anordnung, (b) Ausführung.

```
beginfig(1)
  path p[]; picture lab[]; numeric w[], h[];       % Deklaration
  ut:=0.6mm;                                      % Zeicheneinheit
  d:=5ut;                                            % Abstand
  w2:=80ut;                                        % Boxbreiten
  3w1+4d=w2;
  w3:=30ut;
  h1:=7mm; h2:=10.5mm;                          % Boxhoehen, absolut
  lab1:=btex AV-Dis etex;                            % Boxentexte
  lab2:=btex AJR etex;
  lab3:=btex JET etex;
  lab4:=btex Basisma\ss{}nahmen etex;
  lab5:=btex \begin{tabular}{c}AV-Dis\\Protokoll\end{tabular} etex;
  lab6:=btex \begin{tabular}{c}AJR/JET\\Protokoll\end{tabular} etex;
  lab7:=btex AOO-Pacing etex;
  lab8:=btex AVT-Pacing etex;
  H:=0; W:=0;                                     % Boxen definieren
  H:=H-h1; p1:=unitsquare xscaled w1 yscaled h1 shifted(W,H);
  W:=W+w1+2d; p2:=unitsquare xscaled w1 yscaled h1 shifted(W,H);
  W:=W+w1+2d; p3:=unitsquare xscaled w1 yscaled h1 shifted(W,H);
  H:=H-d-h1;W:=0; p4:=unitsquare xscaled w2 yscaled h1 shifted(W,H);
  H:=H-2d-h2; p5:=unitsquare xscaled w3 yscaled h2 shifted(W,H);
  W:=w2-w3; p6:=unitsquare xscaled w3 yscaled h2 shifted(W,H);
  H:=H-d-h1;W:=0; p7:=unitsquare xscaled w3 yscaled h1 shifted(W,H);
  W:=w2-w3; p8:=unitsquare xscaled w3 yscaled h1 shifted(W,H);
  z4=point 0.5 of p4;                                % Verbindungen
  z5=point 2.5 of p5;
  z6=point 2.5 of p6;
  draw center p1--(xpart center p1,ypart center p4)--center p4;
  draw center p2--center p4;
```

```
  draw center p3--(xpart center p3,ypart center p4)--center p4;
  draw z4--(x4,y4-d);
  draw (x5,y5+d)--(x6,y5+d);
  draw (x5,y5+d)--center p7;
  draw (x6,y6+d)--center p8;
  for i=1 upto 8:                                    % Boxen zeichnen
    unfill p[i];
    draw p[i];
    label(lab[i],center p[i]);
  endfor
endfig;
```

Flexible Boxverbindung Die Verbindungen von Box 1 und Box 3 mit Box 4 in Bild 15.1 sind so gewählt, dass auch bei einer relativen Verschiebung von Box 1 bzw. Box 3 gegenüber Box 4 die Flusslinien nicht im freien Raum enden, sondern die Boxen weiterhin sinnvoll verbunden bleiben, wie Bild 15.2 zeigt. Also nicht, wie vielleicht naheliegend,

```
draw center p1--(-(0,h1+d)+center p1);
```

sondern vom Mittelpunkt der Box 1 und 3 jeweils senkrecht nach unten und im rechten Winkel waagrecht bis zum Mittelpunkt der Box 4.

Bild 15.2: Flexible Boxverbindung.

Horizontales Boxdiagramm Bild 15.3. Bei einem horizontal ausgerichteten Diagramm orientiert man sich an den Spalten und nummeriert die Boxen pro Spalte von oben nach unten. Die Platzierung der Boxen erfolgt dann spaltenweise.

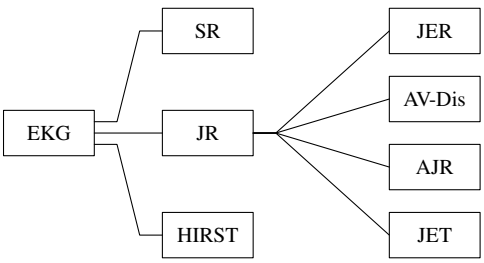


Bild 15.3: Horizontal orientiertes Boxdiagramm.

Vertikal symmetrisches Boxdiagramm Bild 15.4. Das Diagramm ist im wesentlichen spiegelbildlich zur vertikalen Mittellinie strukturiert, sodass nur eine Flusslinie gezeichnet werden muss, die andere ergibt sich dann durch Spiegelung mit dem Befehl `reflectedabout`.

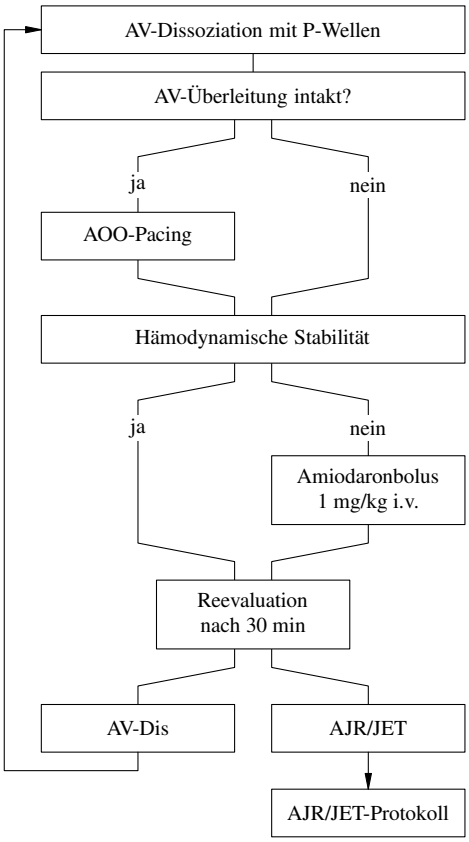


Bild 15.4: Vertikal symmetrisches Flussdiagramm.

Boxdiagramm mit Baumstruktur Bild 15.6. Das Diagramm besitzt eine Baumstruktur. Wenn man die Boxen wie empfohlen rekursiv in der Reihenfolge der in Bild 15.5 angegebenen Nummerierung platziert, empfiehlt es sich, die Positionen der Boxen 3, 8, 11 und 12 auf Variable abzuspeichern als Referenz für die restlichen Boxen auf der jeweils gleichen Höhe. Um die wesentlichen x-Positionen des Diagramms zu ermitteln, beginnt man unten in der letzten Zeile, wo 6 Boxbreiten w_1 jeweils getrennt durch den Abstand d und einmal D zu liegen kommen. Nach oben fortschreitend ergeben sich dann die anderen x-Positionen nacheinander durch symmetrisches Zusammenfassen von je zwei Strängen. Die so ermittelten Abszissenwerte wird man ebenfalls abspeichern, da sie häufig benötigt werden. Dabei ist noch eine kleine horizontale Versetzung zwischen Box 9 und 10 um d zu beachten. Die Skizzen in Bild 15.5 zeigen schematisch die Anordnung der Baumstruktur mit den Boxbreiten und die Nummerierung der Boxen als Tabelle.

Boxdiagramm mit heterogener Baumstruktur Bild 15.8. Auch dieses Diagramm hat eine Baumstruktur, wenn auch etwas unregelmäßig wie in Bild 15.7 zu sehen. Die

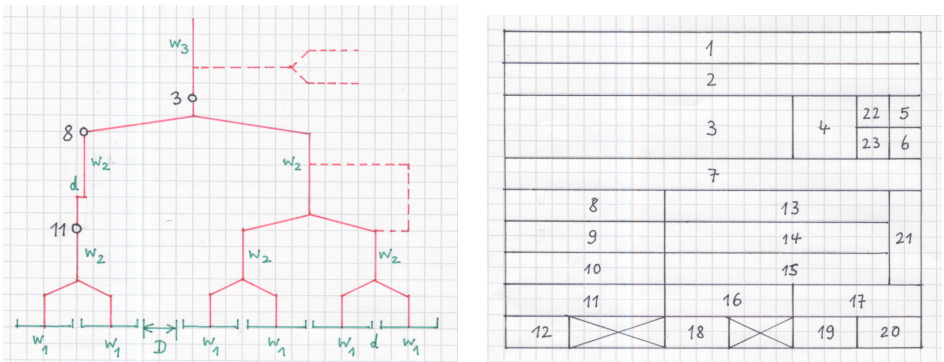


Bild 15.5: Skizzen der Anordnung: (a) Baumstruktur, (b) als Tabelle.

Baumstruktur wurde so in ein Koordinatensystem eingefügt, dass alle Boxen pro Zeile nebeneinander Platz finden. Das Diagramm kann dann in 5 Zeilen und 6 Spalten untergebracht werden. Um Platz zu sparen, wurde versucht, Boxbreiten zu verwenden, die sich in etwa an den Texten orientieren. Ausgehend von der Zeile 3 mit 6 Boxen der angegebenen Boxbreiten w_i und dem jeweiligen Abstand d ergeben sich die x-Positionen der restlichen Boxen, indem man die Boxen 5, 11, 15 und 6, 12, 16 jeweils übereinander und links bzw. rechts einer vertikalen Linie anordnet. Unter einigen Boxen stehen noch zusätzliche Bemerkungen, auch hier ist die Beschriftung in und außerhalb der Box in einer einzigen mehrzeiligen Tabelle mit einem zusätzlichen Zeilenabstand formatiert. Die Beschriftung (Picture) wird mit dem Label-Befehl in die linke obere Ecke der Box platziert gemäß

```
label.lrt(lab[i], point 3 of p[i]);
```

Die Ausrichtung der Boxen pro Zeile erfolgt so, dass sie mit ihrer jeweiligen Oberkante auf einer horizontalen Fluchtlinie liegen.

15.1.2 Pakete für Boxdiagramme

boxes.mp Das Zeichnen von Boxdiagrammen ist eine Standardaufgabe. Es gibt daher eine ganze Reihe von allgemeinen und speziellen Paketen, die diese Arbeit unterstützen. Am bekanntesten ist das Paket boxes.mp von John D. Hobby [19], das wie das Paket graph zur Standard-Distribution von MP gehört. Das Paket enthält eine Reihe von Befehlen zur Platzierung, Beschriftung und Verbindung von „Boxen“ als Rechtecke oder Kreise. Das Blockdiagramm in Bild 15.9 wurde mit dem Paket boxes entworfen. Dazu muss man zu Beginn des MP-Quellfiles das Paket mit `input boxes` laden.

```
input boxes;
beginfig(1)
  a=10bp;
  boxit.one(btex  \bfseries  Breite QRS-Komplexe etex);
  boxit.twoi(btex Ventrikul\"are Tachykardie  etex);
```

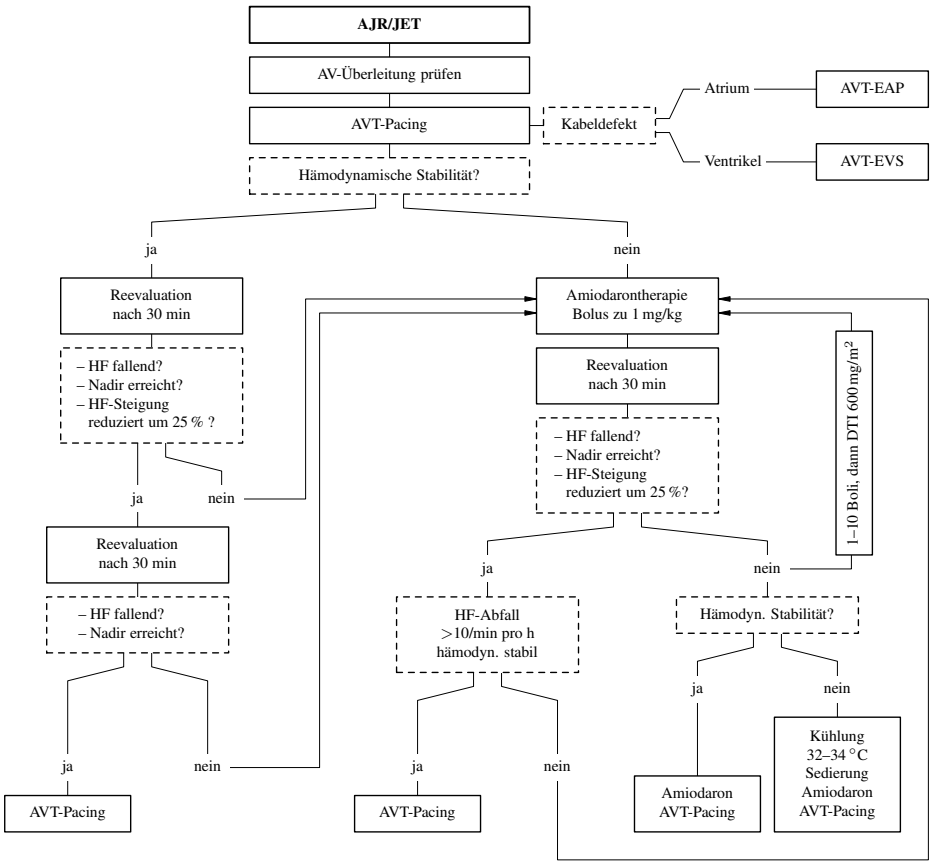


Bild 15.6: Ausführung des Diagramms.

```
boxit.twoii(btex Supraventrikul\"are Tachykardie etex);

boxit.Twoi(btex \vbox{\hbox{\strut}\hbox{- h\"aufig}
  \hbox{\strut}
  \hbox{- QRS $>$ 0.12 s}
  \hbox{\strut}
  \hbox{- QRS-Morphologie:}
  \hbox{\phantom{-} RSB + R/S in V6 $<$ 1 mV}
  \hbox{\phantom{-} LSB + Knotung von S in V1--3}
  \hbox{\strut}
  \hbox{- AV-Dissoziation (beweisend,}
  \hbox{\phantom{-} Ausnahme: junktional ektope}
  \hbox{\phantom{-} Tachykardie)}
  \hbox{\strut}
  \hbox{- Fusionskomplexe (beweisend)}} etex);

boxit.Twoii(btex \vbox{\hbox{\strut}\hbox{- selten}
```

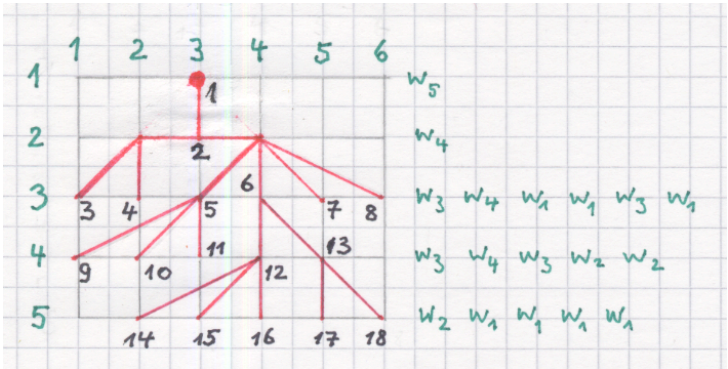


Bild 15.7: Skizze der Anordnung als Baumstrukur in einer Tabelle.

```
\hbox{\strut}
\hbox{- QRS $<$ 0.12 s (vorbestehend?)}
\hbox{\strut}
\hbox{- QRS-Morphologie:}
\hbox{\phantom{-} QRS haupts\"achlich positiv und}
\hbox{\phantom{-} mit \"ahnlicher Morphologie in}
\hbox{\phantom{-} allen Brustwandableitungen}
```

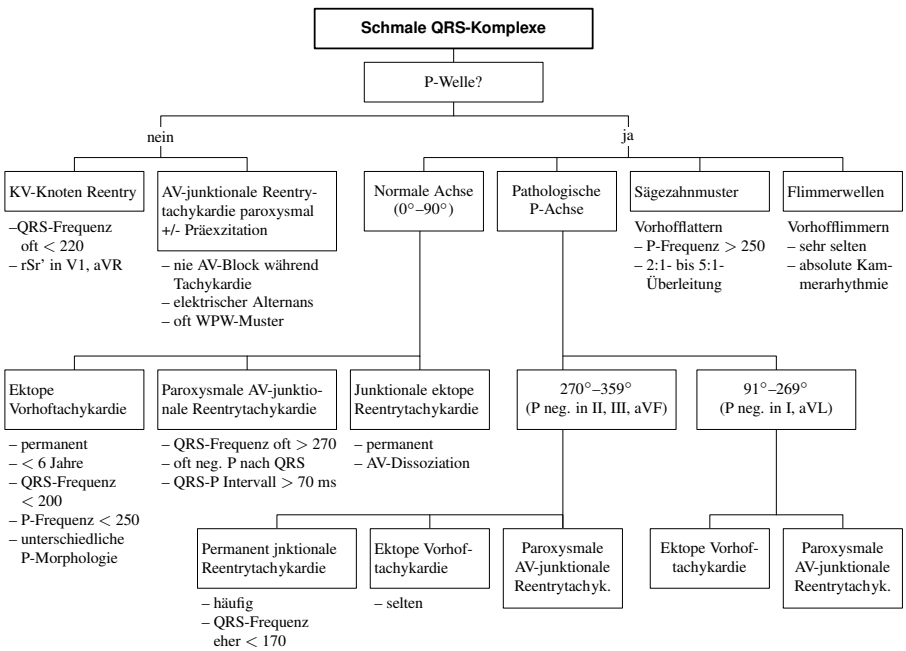


Bild 15.8: Ausführung des Flussdiagramms.

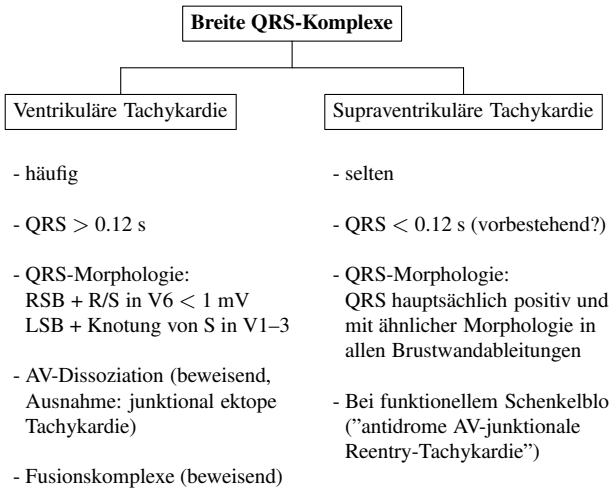


Bild 15.9: Flussdiagramm mit dem Paket boxes.

```
    \hbox{\strut}
    \hbox{- Bei funktionellem Schenkelblock}
    \hbox{\phantom{-} (''antidrome AV-junktionale}
    \hbox{\phantom{-} Reentry-Tachykardie'')}} etex);

  Twoii.nw=Twoi.ne+(a,0);
  drawunboxed(Twoi,Twoii);
  twoi.sw=Twoi.nw;
  twoii.sw=Twoii.nw;
  drawboxed(twoi,twoii);

  one.s=0.5[twoi.n,twoii.n]+(0,2*a);
  drawboxed(one);

  draw (twoi.n+(0,a))--(twoii.n+(0,a));
  draw one.s--(one.s-(0,a));
  draw (twoi.n+(0,a))--twoi.n;
  draw (twoii.n+(0,a))--twoii.n;
endfig;
```

Als weitere Pakete seien nur erwähnt, das Paket `metaobj` von Denis Roegel [36] und die Pakete `automata`, `blockdraw_mp`, `expressg` und `metauml`, die alle auf CTAN zu finden sind.

Klassische Flussdiagramme zur Veranschaulichung des Ablaufs von Assembler- und Fortran-Programmen, die Nassi-Shneiderman Diagramme für blockorientierte Sprachen wie Algol sowie die Syntaxdiagramme zur Definition einer Programmiersprache kommen heute kaum noch zur Anwendung. Sie wurden durch neuere Entwicklungen wie die Literate Programming Methode [25, 28] oder die Backus-Naur-Notation weitgehend

ersetzt.

15.2 Einfügen von externen eps-Bildern in MP

Es besteht oft der Wunsch, in eine MP-Zeichnung ein Bild als externes eps-File einzubinden, beispielsweise als Hintergrundbild, auf das man mit MP-Zeichenbefehlen Linien und Texte als Overlay zeichnen kann. Dies ist mit Bordmitteln von MP nicht möglich, weil der `\special`-Befehl aus TEX nicht ausgewertet wird. So funktioniert auch nicht die Konstruktion
`label(btex \includegraphics{bildfile.eps} etex,z0)`.

Zur Lösung des Problems gibt es das Paket epsincl [3]. Allerdings benötigt man zusätzlich das Linux-Shell-Kommando `gawk`, welches nur auf Linux-basierten Systemen (Ubuntu,…), aber nicht auf BSD-basierten Systemen wie Mac OSX, standardmäßig installiert ist. Als Alternative eignet sich das Paket mplatex von Dag Langmyhr [29], das auch epsincl verwendet, aber ohne `gawk` auskommt, weil die entsprechende Funktionalität in das Paket integriert ist. mplatex läuft auf beiden Plattformen.

15.2.1 Das Paket epsincl

Gleich zu Beginn des MP-Quellfiles muss das Paket epsincl geladen werden mit

```
input epsincl;
```

Ein externes eps-Bildfile, z. B. `puls.eps`, wird mit dem Kommando

```
use_eps("puls.eps")
```

als Picture eingebunden. Der Nullpunkt liegt in der linken unteren Ecke der Boundingbox des eingefügten Bildes und dient als Referenzpunkt für das Overlay.

Beispiel: Quellfile `pulseps.mp`

```
1  input graph;
2  input epsincl;
3  verbatimtex
4  \documentclass{article}
5  \usepackage[T1]{fontenc}
6  \usepackage{helvet}
7  \begin{document}
8  \sffamily
9  \footnotesize
10 etex
11 defaultfont:="phvr8r";
12 defaultscale:=8pt/fontsize defaultfont;
13 Fmfont_:=defaultfont;
14 Fmscale_:=defaultscale;
15 beginfig(1)
```

```
  ut:=1mm; a:=10ut;
  picture pic;
  pic:=use_eps("puls.eps");
  draw pic shifted (-0.02a,-0.02a);
  draw origin withpen pencircle scaled 3bp;
  label.top(btex P  etex,(1.75a,2.5a));
  label.bot(btex Q  etex,(3a,1.5a));
  label.top(btex R  etex,(3.25a,5.75a));
  label.bot(btex S  etex,(3.5a,0.75a));
  label.top(btex T  etex,(5.25a,2.5a));
  label.bot(btex P-Welle  etex,(1.75a,0));
  label.bot(btex \begin{tabular}{c}QRS-\\Komplex\end{tabular} etex,
    (3.25a,0));
  label.bot(btex T-Welle  etex,(5.25a,0));
  drawdblarrow((5a,4a)--(6a,4a));
  label.top(btex 10\,ms  etex,(5.5a,4a));
endfig;
verbatimtex
\end{document}
etex;
end
```

Das File wird mit

```
mpost -tex=latex pulseps
```

bearbeitet. Als Zwischenergebnis erhält man das File `pulseps.1`. Files, die eps-Bilder enthalten, müssen nun einzeln mit `gawk` und dem Programmfile `epsincl.awk` des Pakets nachbearbeitet werden

```
gawk -f epsincl.awk pulseps.1 >pulseps1.eps
```

Das File `epsincl.awk` muss im Arbeitsverzeichnis liegen. Auf Linux-basierten Systemen (Ubuntu, …) kann das Shell-Kommando `gawk`, falls nicht vorhanden, im Ubuntu-Software-Center nachinstalliert werden. Auf BSD-basierten Systemen wie Mac OSX gibt es `gawk` standardmäßig nicht. Eine Nachinstallation der GNU Shell-Kommandos auf Mac OSX ist problematisch, sodass hier die gawk-Kommando nicht ausgeführt und der weitere Weg nicht beschritten werden kann. Abhilfe: Das im nächsten Abschnitt beschriebene Paket mplatex.

Unter Linux fügen wir das mit `gawk` bearbeitete File `pulseps1.eps` gegebenenfalls zusammen mit den anderen Ausgabefiles, wie üblich, in ein LaTeX-Dummy-File ein, um die Grafiken anzuzeigen. Man beachte den unterschiedlichen Filenamen, `pulseps1.eps` gegenüber `pulseps.1`.

15.2.2 Das Paket mplatex

Das von Dag Langmyhr [29] entwickelte Paket mplatex erhält man von der Homepage des Autors. Die Distribution besteht aus der Dokumentation `mplatex.pdf`, die wir nach

~/texmf/doc/ verschieben und dem ausführbaren Perl-Programm mplatex, das nach /usr/local/bin kommt. Eventuell muss man in der ersten Zeile von mplatex den Pfad aktualisieren, wo der Compiler/Interpreter perl gefunden wird.

mplatex ist ein Perl-Programm zur Bearbeitung von MP-Quellfiles, die LaTeX-Code innerhalb von btex-etex-Umgebungen enthalten und in die externe eps-Bilder mit dem epsincl-Befehl use_eps eingebunden sind. Es enthält alle Bearbeitungsschritte und ersetzt damit mpost. Im Gegensatz zu der Nachbearbeitung mit gawk ist diese Funktionalität bereits integriert und die Filenamen der Ausgabefiles werden nicht verändert. Der Shell-Befehl export TEX=latex wird automatisch ausgeführt. Dadurch vereinfacht sich die Anwendung für den Benutzer erheblich. Auch das MP-Quellfile ist einfacher, weil die LaTeX-Präambel und Postambel entfallen. Diese werden von mplatex automatisch hinzugefügt anhand der beim Aufruf des Programms anzugebenden Optionen. Voreingestellt ist die Dokumentklasse article, ohne Klassenoptionen.

Aufruf:

```
mplatex [-C class][-co class option]...
   [[-P package][-po package option]...]... [-v] file
```

Die Option -v (verbous) dient nur zu debugging-Zwecken.

file ist der vollständige Filename des MP-Quellfiles einschließlich Extension mp.

Weitere LaTeX-Anweisungen kann man mit Hilfe eines eigenen LaTeX-Stilfiles einfügen, das man beim Aufruf von mplatex als letztes Paket mit der Option -P lädt. Mit dem Kommando \AtBeginDocument{} kann man auch LaTeX-Befehle in das Stilfile eintragen, die unmittelbar nach \begin{document} ausgeführt werden sollen. Allerdings handelt es sich auch dabei um Präambel-Befehle, die keinen ausführbaren Code erzeugen dürfen.

Beispiel:
Wir schreiben im folgenden das MP-Quellfile puls.mp und ein Stilfile puls.sty, um die Schrift einzustellen. Das File puls.mp ergibt sich aus dem File pulseps.mp durch Streichen der Zeilen 3–10 und 33–35. Wir wollen in LaTeX-Texten den \color-Befehl und die Schrift Helvetica mit Schriftgröße \footnotesize verwenden. In die Grafik soll das externe eps-File puls.eps als Hintergrundbild eingebunden werden.

```
input graph;
input epsincl;
defaultfont:="phvr8r";
defaultscale:=8pt/fontsize defaultfont;
Fmfont_:=defaultfont;
Fmscale_:=defaultscale;
beginfig(1)
  ut:=1mm; a:=10ut;
  picture pic;
  pic:=use_eps("puls.eps");
  draw pic shifted (-0.02a,-0.02a);
  draw origin withpen pencircle scaled 3bp;
```

```
  label.top(btex P  etex,(1.75a,2.5a));
  label.bot(btex Q  etex,(3a,1.5a));
  label.top(btex R  etex,(3.25a,5.75a));
  label.bot(btex S  etex,(3.5a,0.75a));
  label.top(btex T  etex,(5.25a,2.5a));
  label.bot(btex P-Welle  etex,(1.75a,0));
  label.bot(btex \begin{tabular}{c}QRS-\\Komplex\end{tabular} etex,
    (3.25a,0));
  label.bot(btex T-Welle  etex,(5.25a,0));
  drawdblarrow((5a,4a)--(6a,4a));
  label.top(btex 10\,ms  etex,(5.5a,4a));
endfig;
end
```

Das Stilfile `puls.sty` enthält nur die Einstellung des Schriftfonts und der Schriftgröße gemäß

```
\AtBeginDocument{\usefont{T1}{phv}{m}{n}\footnotesize}
```

Bearbeitung:

```
mplatex -P color -P helvet -P puls puls.mp
```

Als Ergebnis erhält man das Ausgabefile `puls.1`, in welches das eps-Bild bereits eingearbeitet ist, sodass wir es ohne weitere Nachbearbeitung, wie üblich, in ein LaTeX-Dummy-File einbinden und die Grafik sichtbar machen können.

Bild 15.10 zeigt das identische Ergebnis für die beiden MP-Programmbeispiele `pulseps` (nur unter Linux wegen `gawk`) und `puls`.

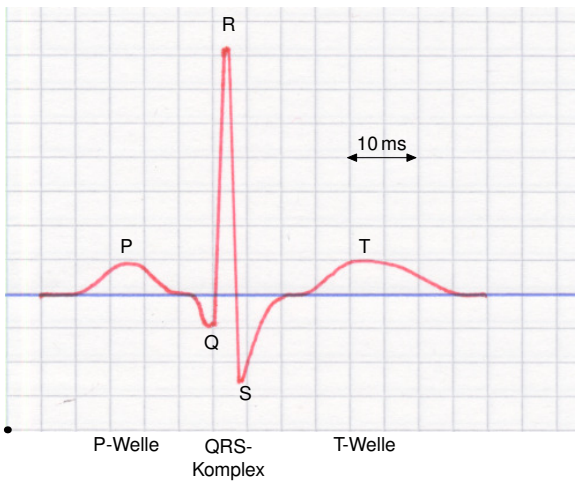


Bild 15.10: Overlay: Externes eps-Bild eingefügt in ein MP-Bild mit den Paketen epsincl und mplatex.

15.3 Text auf Pfad

Mit dem Paket textpath [17] von Stephan Hennig kann man LaTeX-Text entlang eines Pfades setzen. Das Paket ist auf CTAN erhältlich und in der TeX Live-Distribution enthalten. Das Paket basiert auf latexmp und soul.

Beispiel: Das MP-Quellfile, z. B. `textpfad.mp`, hat folgende Struktur

```
% Textpfad
input latexmp;
setupLaTeXMP(preamblefile="mytextpath");
input textpath;
beginfig(1) % Beispiel 1 aus textpathfigs
  path p;
  p:=reverse fullcircle rotated -90 scaled 50bp;
  draw textpath ("Greetings from MetaPost!", p, 0.5);
endfig;
```

Das benötigte Präambel-File, z. B. `mytextpath.tex`, enthält folgende Anweisungen und lädt auch das Stilfile `textpathmp.sty` der Distribution.

```
\RequirePackage{fix-cm}
\documentclass{article}
\usepackage{textpathmp}
```

Zur Anzeige des erzeugten Bildes am Bildschirm verwenden wir, wie üblich, ein LaTeX-Dummy-File.

Damit ergibt sich folgender Workflow:

```
mpost <mpfile>
mpost <mpfile>
latex <dummyfile>
dvips <dummyfile>
gv <dummyfile>.ps
```

Man beachte, dass der mpost-Befehl **zweimal** aufgerufen werden muss, um das zunächst erzeugte File `ltx-<mpfile>.tmp` mit LaTeX zu bearbeiten und das entsprechende File `ltx-<mpfile>.mpx` zu erzeugen.

Der Wert der Umgebungsvariablen `TEX` ist irrelevant. Bei einer eventuellen Fehlermeldung muss man die alten Hilfsfiles `ltx-<mpfile>.tmp` und `ltx-<mpfile>.mpx` löschen.

Die Anwendung des Pakets ist sehr einfach. Der eigentliche Zeichenbefehl lautet

`draw textpath(`⟨ text string ⟩`,`⟨ pfad ⟩`,`⟨ pos ⟩`)`⟨ options ⟩`;`

<text string> enthält den LaTeX-Text als String.
<pfad> ist der Pfad entlang dessen der Text gesetzt werden soll.
<pos> ist ein Wert zwischen 0 und 1, der die Positionierung des Textes auf dem Pfad

zwischen Anfang (0) und Ende (1) angibt. Der Text wird immer in Pfadrichtung links, rechtsbündig ab der angegebenen Position gesetzt.

textpath kennt noch die weiteren Befehle

```
draw textpathRaw(⟨ text string ⟩,⟨ pfad ⟩,⟨ pos ⟩)⟨ options ⟩;
draw textpathFont(⟨ font string ⟩,⟨ text string ⟩,⟨ pfad ⟩,⟨ pos ⟩)
⟨ options ⟩;
```

wobei <font string> die in LaTeX üblichen Befehle zur Festlegung von Font- und Schriftgröße enthält, z. B.

```
"\usefont{T1}{pzc}{m}{n}\fontsize{22pt}{22pt}\selectfont"
```

Der Raw-Befehl eignet sich vor allem zum Setzen von mathematischen Formeln.

Die weitere Gestaltung wird durch interne Variable gesteuert, die innerhalb einer Gruppe mit `interim` lokal geeignet besetzt werden können. Diese sind in Tabelle 15.1 zusammengestellt.

Tabelle 15.1: Interne Variable von `textpath`.

Variable	Voreinstellung	gültig in Makro	
		textpath, textpathFont	textpathRaw
textpathRepeat	1	✓	✓
textpathStretch	1	✓	✓
textpathHSpace	0pt	✓	✓
textpathShift	0pt	✓	✓
textpathLetterSpace	0pt	✓	
textpathRotation	0	✓	✓
textpathAbsRotation	0	✓	✓
textpathClip	1	✓	✓
textpathAutoScale	0	✓	✓
textpathDraft	0	✓	✓
textpathFancyStrokes	1		✓
textpathStrokePrecision	10		✓
textpathCureSqrt	1		✓
textpathDebug	1	✓	✓

Bild 15.11 zeigt einige Beispiele aus der Dokumentation des Pakets zusammen mit den entsprechenden Listings. Viele weitere Bildbeispiele findet man in der Distribution im File `textpathfigs.mp`.

```
% Textpfad
input latexmp;
setupLaTeXMP(preamblefile="mytextpath");
input textpath;
beginfig(1) % Beispiel 3 aus textpathfigs
  path p;
  p:=reverse fullcircle rotated -90;
```

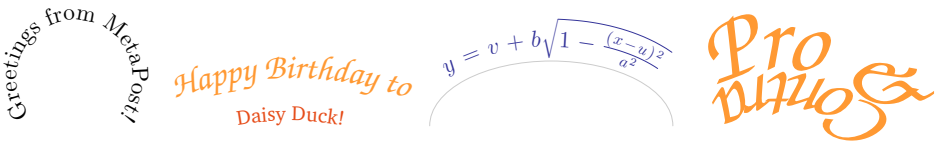


Bild 15.11: LaTeX-Schrift auf einem Pfad mit dem Paket textpath.

```
  draw textpathFont("\usefont{T1}{pzc}{m}{n}\huge",
    "Happy Birthday to",
    p scaled 400bp,0.5) withcolor(1,0.6,0.2);
  draw textpathFont("\usefont{T1}{bch}{m}{n}\large","Daisy Duck!",
    p scaled 350bp,0.5) withcolor(0.9,0.3,0.1);
endfig;

% Textpfad
input latexmp;
setupLaTeXMP(preamblefile="mytextpath");
input textpath;
beginfig(1) % Beispiel 17 aus textpathfigs
  path p;
  p:=reverse halfcircle xscaled 150bp yscaled 80;
  draw p withcolor 0.8white;
  interim textpathShift:=9pt;
  interim textpathFancyStrokes:=1;
  interim textpathCureSqrt:=1;
  draw textpathRaw("\Large$y=v+b\sqrt{1-\frac{(x-u)^2}{a^2}}$",
    p,0.5) withcolor(0.2,0.2,0.6);
endfig;

% Textpfad
input latexmp;
setupLaTeXMP(preamblefile="mytextpath");
input textpath;
beginfig(1) % Beispiel 18 aus textpathfigs
  string f; path p;
  f:="\usefont{T1}{pzc}{m}{it}\fontsize{56pt}{56pt}\selectfont";
  p:=reverse quartercircle rotated 33 scaled 150bp;
  interim textpathLetterSpace:=-3pt;
  draw textpathFont(f,"Pro \&",p,1) withcolor(1,0.6,0.2);
  interim textpathLetterSpace:=0pt;
  f:="\usefont{T1}{pzc}{m}{it}\fontsize{48pt}{48pt}\selectfont";
  p:=quartercircle rotated 33 scaled 145bp;
  draw textpathFont(f,"Con\<{\kern -2pt}tra",p,0)
    withcolor(1,0.6,0.2);
endfig;
```