PSTricks

# pst-tree

Knoten und Bäume; v.1.12

28. September 2011

Documentation by
**Herbert Voß**
Übersetzung aus dem Englischen:
**Alexander Mischke von der FSU Jena (alex.mischke@web.de)**

Package author(s):
**Timothy Van Zandt**
**Herbert Voß**

# Inhaltsverzeichnis

Knoten und Knotenverbindungen sind perfekte Werkzeuge für die Erstellung von Bäumen, allerdings wäre die Positionierung der Knoten mittels `\rput` ungleich aufwendiger.[1] Die Datei `pst-tree.tex/pstree.sty` enthält eine fortgeschrittene Schnittstelle für die Baumerstellung.

Es sollte angemerkt werden, dass ein korrektes Ergebnis nicht unter jedem `dvips`-Treiber garantiert werden kann. Dieses Paket wurde für das `dvips`-Programm von Rokicki geschrieben, welches praktisch Bestandteil jeder TEX-Distribution ist.

1 Wenn man nicht ein Programm hat, das die Koordinaten erzeugt.

# 1 Überblick

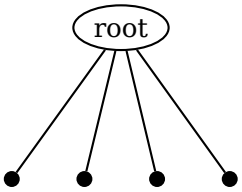
Die Baumbefehle sind

```
\pstree{<root>}{<successors>}
```

| TEX version | LATEX version |
|---|---|
| \psTree{<root>} | \begin{psTree}{root} |
| <successors> | <successors> |
| \endpsTree | \end{psTree} |

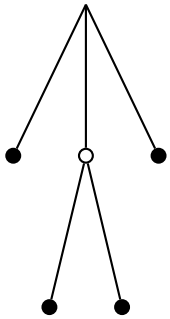
Diese Befehle erfüllen die gleiche Funktion und haben nur eine unterschiedliche Syntax. \psTree ist die „lange“ Version. Diese Makros erstellen eine Box, die alle Knoten einschließt und deren Grundlinie durch den Mittelpunkt des Wurzelknotens verläuft. Die meisten der Knoten haben eine Variante für die Verwendung innerhalb eines Baumes und werden Baumknoten genannt (Siehe Abschnitt 2).

Bäume und Baumknoten werden *Baumobjekte* genannt. Die *Wurzel* eines Baumes sollte ein einzelnes Baumobjekt sein und die *Nachfolger* sollten ein oder mehrere Baumobjekte sein. Hier ist ein Beispiel, das nur aus Knoten besteht:




```
\pstree[radius=3pt]{\Toval{root}}{\TC* \
    TC* \TC* \TC*}
```

Es gibt keinen Unterschied zwischen einem End- und einem Wurzelknoten, abgesehen von der Position im \pstree{}-Befehl. Das folgende Beispiel zeigt einen Baum, der in der Liste der Nachfolger erscheint und so ein Unterbaum wird:



```
\pstree[radius=3pt]{\Tp}{%
  \TC*
  \pstree{\TC}{\TC* \TC*}
  \TC*}
```

# 2 Baumknoten

Der Name des Baumknotens ergibt sich immer durch Auslassen des „node “ vom Ende des Namens und durch Anfügen eines „T “ am Anfang. Beispielsweise wird \psovalnode zu \Toval. Hier ist eine Liste solcher Baumknoten:

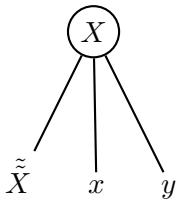
```
\Tp * [Options]
\Tc * [Options] {dim}
\TC * [Options]
\Tf * [Options]
\Tdot * [Options]
\Tr * [Options] {stuff}
\TR * [Options] {stuff}
\Tcircle * [Options] {stuff}
\TCircle * [Options] {stuff}
\Toval * [Options] {stuff}
\Tdia * [Options] {stuff}
\Ttri * [Options] {stuff}
```

Die Syntax eines Baumknotens gleicht der des korrespondierenden „normalen“ Knotens, außer dass:

- es immer ein optionales Argument für die Einstellung von grafischen Parametern gibt, auch wenn der ursprüngliche Knoten keines hatte;
- es kein Argument zum Spezifizieren des Knotennamens gibt;
- es nie ein Koordinatenargument zum Positionieren des Knotens gibt und
- um den Referenzpunkt bei `\Tr` zu setzen, der `ref`-Parameter benötigt wird.

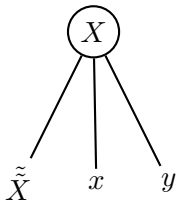
Abbildung 1 auf der nächsten Seite gibt einen Überblick, wie die einzelnen Knoten aussehen.

Der Unterschied zwischen `\Tr` und `\TR` (jeweils Varianten von `\rnode` und `\Rnode`) spielt in Bäumen eine wichtige Rolle. Für gewöhnlich will man `\TR` bei vertikalen Bäumen verwenden, da die Grundlinien des Textes bündig horizontal verlaufen. Zum Beispiel:

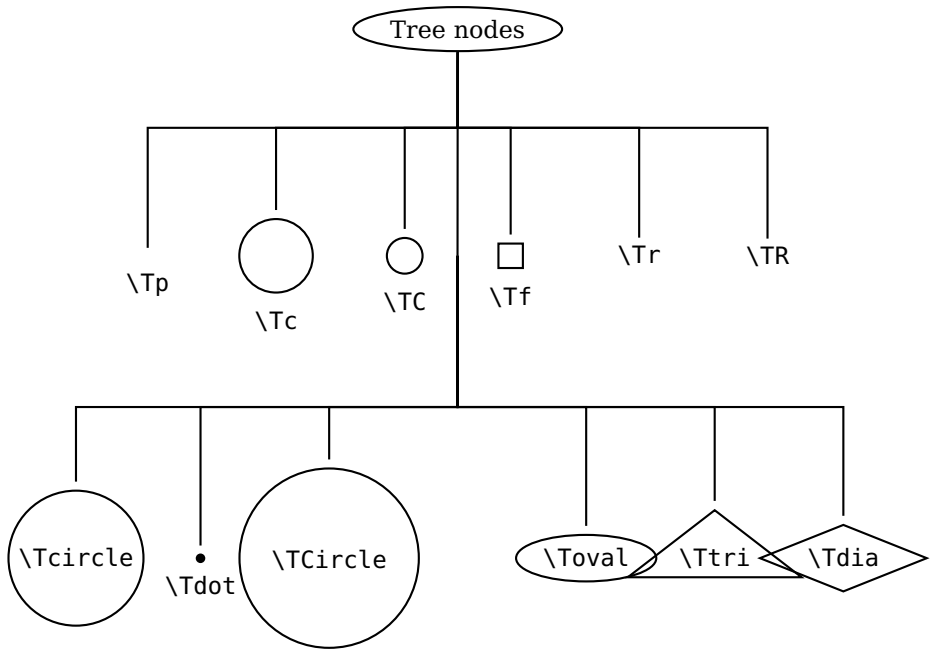



```
$
\pstree[nodesepB=3pt]{\Tcircle{X}}{%
  \TR{\tilde{\tilde{X}}}
  \TR{x}
  \TR{y}}
$
```

Vergleichen Sie mit diesem Beispiel, welches `\Tr` verwendet:




```
$
\pstree[nodesepB=3pt]{\Tcircle{X}}{%
  \Tr{\tilde{\tilde{X}}}
  \Tr{x}
  \Tr{y}}
$
```




```
\small
\psset{armB=1cm, levelsep=3cm, treesep=-3mm, angleB=-90, angleA=90, nodesepA=3pt}
\def\s#1{#1~{\tt\string#1}}\def\b#1{#1{\tt\string#1}}\def\psedge#1#2{\ncangle{#2}{#1}}
\psTree[treenodesize=1cm]{\Toval{Tree nodes}}
 \s\Tp\Tc{.5}~{\tt\string\Tc} \s\TC
 \psTree[levelsep=4cm,armB=2cm]{\Tp[edge=\ncline]}
  \b\Tcircle \s\Tdot
  \TCircle[radius=1.2]{\tt\string\TCircle}
  \Tn \b\Toval \b\Ttri \b\Tdia
 \endpsTree
 \s\Tf \b\Tr \b\TR
\endpsTree
```

**Abbildung 1:** Die Baumknoten.

Es gibt auch einen Null-Baumknoten:

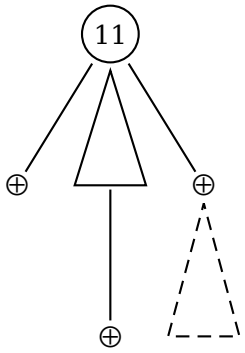
```
\Tn
```

Dieser ist als Platzhalter gedacht. Sehen Sie sich den Baum in Abb. 1 auf Seite 6 an: Bei der unteren Reihe fehlt ein Knoten in der Mitte. `\Tn{}` wurde für diesen fehlenden Knoten verwendet.

Es gibt auch einen besonderen Baumknoten, der keine „normale “Version aufweist und der nicht als Wurzelknoten eines kompletten Baumes genutzt werden kann:

```
\Tfan * [Options]
```

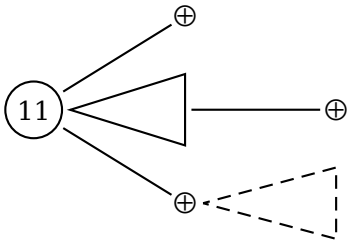
Dieser Befehl zeichnet ein Dreieck, dessen Grundlinie sich aus `fansize` ergibt und dessen gegenüberliegende Ecke (anhand der Werte von `nodesepA` und `offsetA` ausgerichtet) den vorhergehenden Knoten bildet. Zum Beispiel:




```
\pstree[dotstyle=oplus,dotsize=8pt,
    nodesep=2pt]{\Tcircle{11}}{%
   \Tdot
   \pstree{\Tfan}{\Tdot}
   \pstree{\Tdot}{\Tfan[linestyle=dashed
       ]}}
```

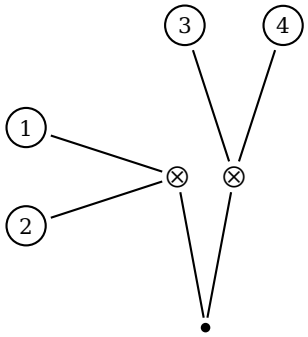
## 3 Baumausrichtung

Bäume können nach unten, oben, rechts oder links wachsen, abhängig vom `treemode=D`, `U`, `R`, oder `L`-Parameter.
So sähe das vorherige Beispiel mit Rechtswachstum aus:




```
\pstree[dotstyle=oplus,dotsize=8pt,
 nodesep=2pt,treemode=R]
 {\Tcircle{11}}{%
   \Tdot
   \pstree{\Tfan}{\Tdot}
   \pstree{\Tdot}{\Tfan[linestyle=dashed]}}
```

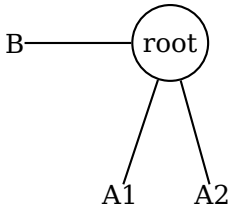
Die Ausrichtung von `treemode` kann auch mitten im Baum erfolgen. Hier ist z. B. ein Baum, der nach oben wächst und dann einen Unterbaum beinhaltet, welcher nach links wächst:




```
\footnotesize
\pstree[treemode=U,dotstyle=otimes,dotsize=8pt,
    nodesep=2pt]
 {\Tdot}{%
   \pstree[treemode=L]{\Tdot}{\Tcircle{1} \Tcircle
       {2}}
   \pstree{\Tdot}{\Tcircle{3} \Tcircle{4}}}
```

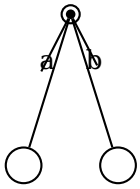
Da man die Ausrichtung eines Baumes ändern kann, kann es sinnvoll sein, einen Baum (<baumA>) als Wurzelknoten (von <baumB>) einzuschließen. Das ergibt einen einzelnen logischen Baum, dessen Wurzel die Wurzel von <baumB> ist und der Nachfolger in

verschiedenen Richtungen hat, abhängig davon, ob diese als Nachfolger von <baumA> oder <baumB> erscheinen.



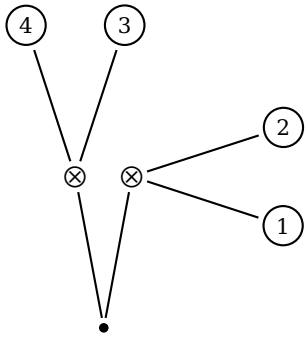

```
\pstree{\pstree[treemode=L]{\Tcircle{root}}{\Tr{B
    }}}{%
 \Tr{A1}
 \Tr{A2}}
```

Beachten Sie als verwandtes Thema, dass jeder Knoten, der eine LR-Box erzeugt, einen Baum enthalten kann. Verschachtelte Bäume dieser Art stehen jedoch in keiner Beziehung zum Rest des Baumes. Hier ist ein Beispiel dafür:




```
\psTree{\Tcircle{%
       \pstree[treesep=0.4,levelsep=0.6,
             nodesepB=-6pt]{\Tdot}{%
        \TR{a} \TR{b}}}}
 \TC
 \TC
\endpsTree
```

Wenn der Baum nach oben oder unten wächst, werden die Nachfolger in der Reihenfolge, in der sie in `\pstree` erscheinen von links nach rechts aufgereiht. Wenn der Baum nach links oder rechts wächst, werden die Nachfolger von oben nach unten aufgereiht. Im Nachhinein könnte man die Reihenfolge der Knoten umdrehen. Der Code `treeflip=true/false` ermöglicht dies. Zum Beispiel:




```
\footnotesize
\pstree[treemode=U,dotstyle=otimes,dotsize=8pt,
 nodesep=2pt,treeflip=true]{\Tdot}{%
  \pstree[treemode=R]{\Tdot}{\Tcircle{1} \Tcircle
      {2}}
  \pstree{\Tdot}{\Tcircle{3} \Tcircle{4}}}
```

Beachten Sie, dass man immer noch zurückgehen und den `treemode` des Unterbaums, der nach links wuchs, ändern müsste.

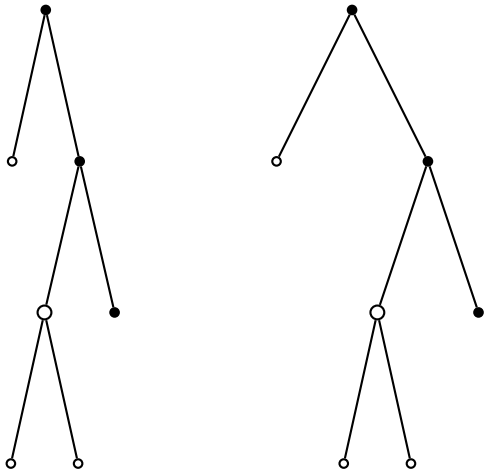
# 4 Der Abstand zwischen Nachfolgern

Der Abstand zwischen Nachfolgern wird durch `treesep` bestimmt. Der Rest dieses Abschnitts zeigt Wege, wie man die Feinabstimmung für die Zwischenräume bei Nachfolgern vornimmt. Die Art und Weise wie die Abstände zwischen Unterbäumen berechnet werden, kann durch den Parameter `treefit=tight/loose` geändert werden. Hier sind zwei Methoden:

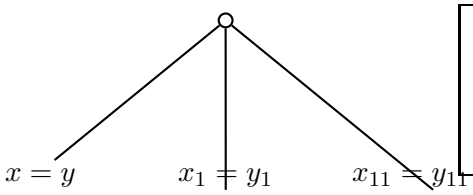
*`tight`* Wenn `treefit=tight` (Normalfall) eingestellt ist, wird `treesep` der Minimalabstand zwischen jeder der Ebenen der Unterbäume.

*loose* Wenn `treefit=loose` eingestellt ist, wird `treesep` der Abstand zwischen den Bounding Boxes der Unterbäume. Außer bei großen Zwischenknoten hat dies zur Folge, dass der horizontale Abstand (oder der vertikale Abstand bei horizontalen Bäumen) zwischen all den Endknoten der gleiche ist (auch wenn sie sich auf verschiedenen Ebenen befinden).[2]

Zum Vergleich:

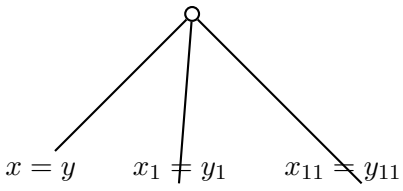


Bei `treefit=loose` benötigen Bäume mehr Platz, die Struktur des Baumes wird dabei manchmal betont.
Manchmal will man den Abstand zwischen den Mittelpunkten der Knoten regelmäßig gestalten, auch wenn die Knoten unterschiedlich groß sind. Wenn man `treenodesize` auf einen positiven Wert setzt, stellt PSTricks die Breite (oder Höhe+Tiefe bei vertikalen Bäumen) auf `treenodesize`, *um den Abstand zwischen den Nachfolgern zu berechnen.* Dreigliedrige Bäume sehen beispielsweise gut aus, wenn sie wie im folgenden Beispiel symmetrisch sind:




```
\pstree[nodesepB=-8pt,treenodesize=.85]{\Tc{3pt}}{
    %
 \TR{$x=y$}
 \TR{$x_1=y_1$}
 \TR{$x_{11}=y_{11}$}}%$
```

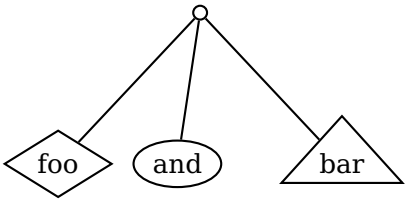
Vergleichen Sie mit folgendem Beispiel, wo sich der Abstand mit der Größe der Knoten ändert:




```
\pstree[nodesepB=-8pt]{\Tc{3pt}}{%
 \TR{$x=y$}
 \TR{$x_1=y_1$}
 \TR{$x_{11}=y_{11}$}}%$
```

2 Wenn alle Endknoten auf der gleichen Ebene liegen und die Zwischenknoten nicht breiter als die Grundlinien ihrer zugehörigen Unterbäume sind, entfällt der Unterschied zwischen beiden Methoden.

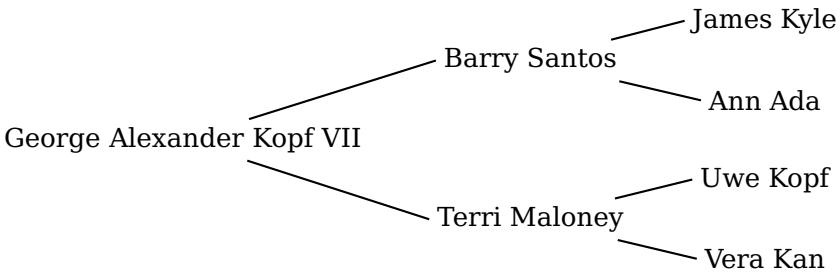
Wenn alles andere nicht funktioniert, kann man den Abstand zwischen zwei Nachfolgern durch das Einfügen von `\tspace{`*length*`}` zwischen beiden ausrichten:




```
\pstree{\Tc{3pt}}{%
  \Tdia{foo}
  \tspace{-0.5}
  \Toval{and}
  \Ttri{bar}}
```

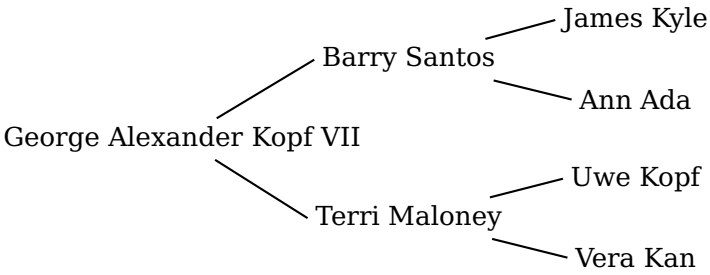
## 5 Abstand zwischen der Wurzel und den Nachfolgern

Der Abstand zwischen den Mittellinien der Baumebenen ist `levelsep`. Wenn man will, dass sich der Abstand zwischen den Ebenen mit deren Größe ändert, verwendet man die *-Konvention. `levelsep` ist dann der Abstand zwischen dem unteren Ende der einen und dem oberen Ende der nächsten Ebene (oder zwischen den Seiten der beiden Ebenen bei horizontalen Bäumen).
Beachten Sie: PSTricks muss Informationen in die `.aux`-Datei (unter LaTeX) oder `\jobname.pst` andererseits schreiben, um den Abstand zu berechnen. Sie müssen die Eingabedatei einige Male kompilieren, bevor PSTricks die Abstände korrekt einrichtet. Vergleichen Sie das folgende Beispiel:




```
\def\psedge#1#2{\ncdiagg[nodesep=3pt,angleA=180,armA=0]{#2}{#1}}
\pstree[treemode=R,levelsep=*1cm]{\Tr{George Alexander Kopf VII}}{%
   \pstree{\Tr{Barry Santos}}{\Tr{James Kyle} \Tr{Ann Ada}}
   \pstree{\Tr{Terri Maloney}}{\Tr{Uwe Kopf} \Tr{Vera Kan}}}
```

mit diesem Beispiel, in dem der Abstand zwischen den Ebenen unverändert ist:




```
\def\psedge#1#2{\ncdiagg[nodesep=3pt,angleA=180,armA=0]{#2}{#1}}
\pstree[treemode=R,levelsep=3cm]{\Tr{George Alexander Kopf VII}}{%
   \pstree{\Tr{Barry Santos}}{\Tr{James Kyle} \Tr{Ann Ada}}
   \pstree{\Tr{Terri Maloney}}{\Tr{Uwe Kopf} \Tr{Vera Kan}}}
```

# 6 Äste

Wenn man einen Baumknoten-Befehl verwendet, ist \pssucc gleich dem Namen des Knoten und \pspred ist gleich dem Namen des Vorgängerknotens. Daher kann man eine Linie zwischen dem Knoten und dessen Vorgänger zeichnen, indem man beispielsweise Folgendes einfügt:

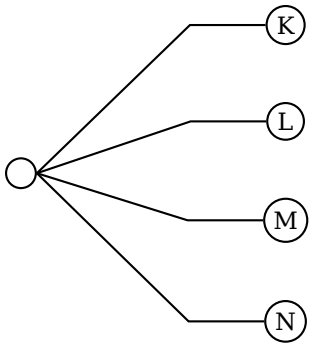
```
\ncline{\pspred}{\pssucc}
```

Um sich den Aufwand für jeden Knoten zu ersparen, führt jeder Knoten folgenden Befehl aus:

```
 \psedge{\pspred}{\pssucc}
```

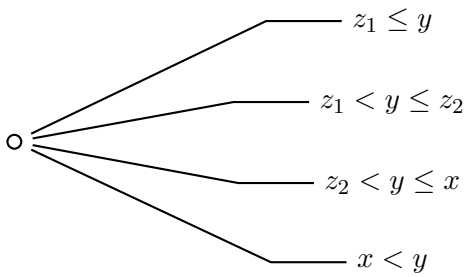
Die Standarddefiniton für \psedge ist \ncline, kann aber nach Belieben mit \def oder \renewcommand von LaTeX redefiniert werden.
Im Folgenden wird beispielsweise \ncdiag mit armA=0 verwendet, um alle Knotenverbindungen vom selben Punkt des Vorgängers ausgehen zu lassen. LaTeX-User können stattdessen eingeben:

```
\renewcommand{\psedge}{\ncdiag[armA=0,angleB=180,armB=1cm]}
```



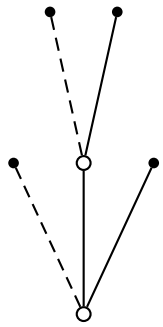

```
\def\psedge{\ncdiag[armA=0,angleB=180,armB=1cm]}
\pstree[treemode=R,levelsep=3.5cm,framesep=2pt]{\
    Tc{6pt}}{%
   \small \Tcircle{K} \Tcircle{L} \Tcircle{M} \
       Tcircle{N}}
```

Hier ist ein Beispiel mit \ncdiagg. Beachten Sie die Verwendung eines negativen Wertes für armA, damit die Ecken der Äste vertikal ausgerichtet sind, auch wenn die Knoten verschieden groß sind:



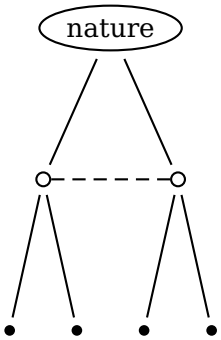

```
$
\def\psedge#1#2{\ncdiagg[angleA=180,armA=1cm,nodesep=4pt]{#2}{#1}}
% Or: \renewcommand{\psedge}[2]{ ... }
\pstree[treemode=R, levelsep=5cm]{\Tc{3pt}}{%
  \Tr{z_1\leq y} \Tr{z_1<y\leq z_2} \Tr{z_2<y\leq x} \Tr{x<y}
}
$
```

Eine andere Methode, `\psedge{}` zu definieren, stellt die Verwendung des `edge`-Parameters dar. Stellen Sie sicher, den Wert in geschweifte Klammern einzuschließen, wenn er Kommata oder andere Parameterbegrenzungen enthält. Dies sieht in langen Befehlen sehr unübersichtlich aus und man kann keine Argumente wie im vorangegangenen Beispiel verwenden, für einfache Änderungen ist es aber nützlich. Wenn man zum Beispiel häufig zwischen einigen Knotenverbindungen umschalten möchte, könnte man einen Befehl für jede Knotenverbindung definieren und dann den `edge`-Parameter benutzen.



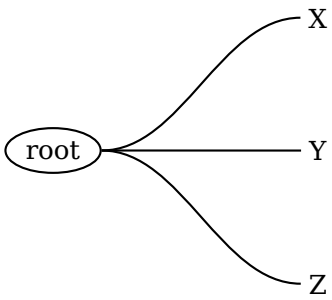
```
\def\dedge{\ncline[linestyle=dashed]}
\pstree[treemode=U,radius=2pt]{\Tc{3pt}}{%
  \TC*[edge=\dedge]
  \pstree{\Tc{3pt}}{\TC*[edge=\dedge] \TC*}
  \TC*}
```

Man kann auch `edge=none` einstellen um die Knotenverbindung zu unterdrücken.
Wenn man eine Knotenverbindung zwischen zwei Knoten zeichnen möchte, die nicht direkt Vorgänger und Nachfolger sind, muss man den Knoten einen Namen geben, auf den man verweisen kann, indem man den `name`-Parameter benutzt. Hier verbinde ich beispielsweise zwei Knoten auf der gleichen Ebene:

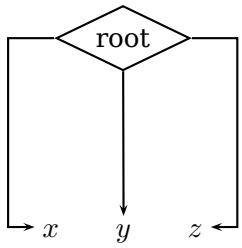



```
\pstree[nodesep=3pt,radius=2pt]{\Toval{nature}}{%
  \pstree{\Tc[name=top]{3pt}}{\TC* \TC*}
  \pstree{\Tc[name=bot]{3pt}}{\TC* \TC*}}
 \ncline[linestyle=dashed]{top}{bot}
```

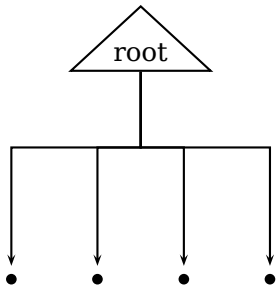
Abschließend noch weitere Beispiele.




```
\def\psedge{\nccurve[angleB=180, nodesepB=3pt]}
\pstree[treemode=R, treesep=1.5, levelsep=3.5]%
  {\Toval{root}}{\Tr{X} \Tr{Y} \Tr{Z}}
```



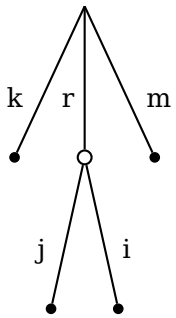

```
\pstree[nodesepB=3pt, arrows=->, xbbl=15pt,
 xbbr=15pt, levelsep=2.5cm]{\Tdia{root}}{%
 $
 \TR[edge={\ncbar[angle=180]}]{x}
 \TR{y}
 \TR[edge=\ncbar]{z}
 $}
```




```
\psset{armB=1cm, levelsep=3cm, treesep=1cm,
 angleB=-90, angleA=90, arrows=<-, nodesepA=3pt}
\def\psedge#1#2{\ncangle{#2}{#1}}
\pstree[radius=2pt]{\Ttri{root}}{\TC* \TC* \TC* \
    TC*}
```

# 7 Labels bei Ästen und Knoten

Typischerweise wird eine Linie gleich nach dem Knoten gezeichnet und man kann Labels mittels `\ncput`, `\tlput` usw. anhängen. Mit `\tlput`, `\trput`, `\taput` und `\tbput` können die Labels vertikal oder horizontal ausgerichtet werden, genau wie bei Knoten. Das mag gut aussehen, zumindest dann, wenn die Neigung der Knotenverbindungen nicht zu verschieden ist.



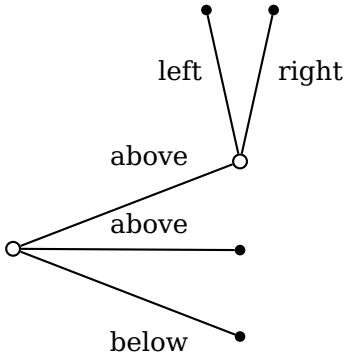

```
\pstree[radius=2pt]{\Tp}{%
 \psset{tpos=.6}
 \TC* \tlput{k}
 \pstree{\Tc{3pt} \tlput[labelsep=3pt]{r}}{%
  \TC* \tlput{j}
  \TC* \trput{i}}
 \TC* \trput{m}}
```

Innerhalb von Bäumen misst der Parameter `tpos` die Entfernung zwischen Vorgänger und Nachfolger, unabhängig von der Ausrichtung des Baumes (außerhalb von Bäumen misst er die Entfernung von oben nach unten verlaufenden bzw. von links nach rechts verlaufenden Knoten).

Weiterhin setzt PSTricks `shortput=tab` innerhalb von Bäumen. Dabei handelt es sich um eine spezielle Option von `shortput`, die nicht außerhalb von Bäumen verwendet werden sollte. Sie implementiert die folgenden Kürzel, die von der Ausrichtung des Baumes abhängen:

| | Kurz für: | |
|---|---|---|
| *Zeichen* | *Vertikale* | *Horizontale* |
| ^ | `\tlput` | `\taput` |
| _ | `\trput` | `\tbput` |

(Das Schema ist umgekehrt, wenn `treeflip=true` ).

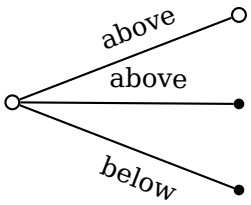



```
\psset{tpos=.6}
\pstree[treemode=R, thistreesep=1cm,
 thislevelsep=3cm,radius=2pt]{\Tc{3pt}}{%
 \pstree[treemode=U, xbbr=20pt]{\Tc{3pt}^{above
     }}{%
  \TC*^{left}
  \TC*_{right}}
 \TC*^{above}
 \TC*_{below}}
```

Geändert werden können die Zeichenkürzel mittels

`\MakeShortTab{`*<char1>*`}{`*<char2>*`}`

Die `\n*put`-Befehle können auch für gute Ergebnisse sorgen:

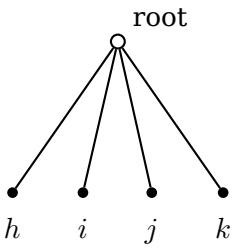



```
\psset{npos=.6,nrot=:U}
\pstree[treemode=R, thistreesep=1cm,
 thislevelsep=3cm]{\Tc{3pt}}{%
 \Tc{3pt}\naput{above}
 \Tc*{2pt}\naput{above}
 \Tc*{2pt}\nbput{below}}
```

Mittels `\nput` kann man Knoten mit Labels versehen. Jedoch lässt `\pstree` diese Labels außer acht, wenn es die Bounding Boxes berechnet.
Es gibt eine spezielle Knotenlabel-Option für Bäume, die die Bounding Boxes berücksichtigt:

`~*[Options]{`*stuff*`}`

Nennen wir dies ein „Baumknoten-Label “
Setzen Sie ein Baumknoten-Label gleich nach dem Knoten, auf den es sich beziehen soll, bevor Sie dies für die anderen Knotenverbindungs-Labels tun (Knotenverbindungs-Labels, inklusive der Kurzformen, können aber auf ein Baumknoten-Label folgen). Das Label wird in vertikalen Bäumen direkt unter dem Knoten positioniert; in anderen Bäumen verläuft dies ähnlich. Zum Beispiel:




```
\pstree[radius=2pt]{\Tc{3pt}\nput{45}{\pssucc}{root}}{%
 \TC*~{$h$} \TC*~{$i$} \TC*~{$j$} \TC*~{$k$}}
```

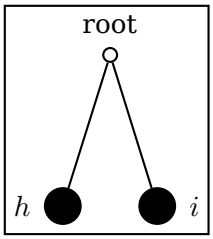
Beachten Sie, dass es keine „Langform“ für dieses Baumknoten-Label gibt. Man kann jedoch das einzelne Zeichen für die Begrenzung des Labels folgendermaßen ändern:

`\MakeShortTnput{<char1>}`

Wenn Sie es verwirrend finden, ein Einzelzeichen zu verwenden, können Sie auch auf eine Befehlssequenz zurückgreifen. Zum Beispiel:

```
\MakeShortTnput{\tnput}
```

Man kann mehrere Labels setzen, aber jedes folgende Label wird relativ zur die vorherigen Labels beherbergenden Bounding Box positioniert. Daher macht es einen Unterschied, in welcher Reihenfolge die Labels platziert werden und nicht alle Kombinationen ergeben zufriedenstellende Ergebnisse.
Sie werden wahrscheinlich finden, dass die Baumknoten-Labels gut bei Endknoten funktionieren, auch ohne Ihr Zutun. Wie dem auch sei: Man kann die Baumknoten-Labels durch das Setzen verschiedener Parameter kontrollieren.
Um das Label auf einer beliebigen Seite des Knotens zu platzieren („l“eft, „r“ight, „a“bove oder „b“elow), stellt man `tnpos=l/r/a/b` ein.




```
\psframebox{%
  \pstree{\Tc{3pt}~[tnpos=a,tndepth=0pt,radius=4pt]{
      root}}{%
    \TC*~[tnpos=l]{$h$}
    \TC*~[tnpos=r]{$i$}}}
```

Wenn das Argument leer gelassen wird (Normalfall), wählt PSTricks die Labelposition automatisch.
Um den Abstand zwischen Knoten und Label zu ändern, setzen sie `tnsep` auf den entsprechenden Wert. Wenn das Argument leer gelassen wird (Normalfall), verwendet PSTricks den Wert von `labelsep`. Bei einem negativen Wert wird der Abstand vom Mittelpunkt des Knotens aus gemessen.
Wenn Labels unterhalb eines Knotens positioniert werden, erhält das Label eine Minimalhöhe von `tnheight`. Wenn man also Labels mehreren Knoten zuweist, die horizontal ausgerichtet sind, und wenn diese Knoten entweder die gleiche Tiefe besitzen oder `tnsep` oder diese negativ ist und die Höhe jedes der Labels nicht größer als `tnheight` ist, werden die Labels ebenfalls anhand ihrer Grundlinien ausgerichtet. Der Normalfall ist `\ht\strutbox`, welcher in den meisten TEX-Formaten die Höhe einer typischen Textlinie in der gebräuchlichen Schriftart darstellt. Beachten Sie, dass der Wert von `tnheight` nicht berechnet wird, bevor er verwendet wird.
Die Positionierung ähnelt der von Labels, die unter einem Knoten stehen. Das Label erhält eine *Mindesttiefe* von `tndepth`. Für oberhalb oder unterhalb platzierte Labels wird der horizontale Referenzpunkt des Labels, d. h. der Punkt im Label direkt über oder unter dem Mittelpunkt des Knotens durch den `href`-Parameter gesetzt.
Wenn Labels rechts oder links platziert werden, wird die rechte oder linke Kante des Labels in der Entfernung `tnsep` vom Knoten positioniert. Der vertikale Punkt, der im Zentrum des Knotens ausgerichtet wird, ergibt sich aus `tnyref`. Wenn dies leer bleibt, wird stattdessen `vref` verwendet. Behalten Sie im Gedächtnis, dass `vref` den vertikalen Abstand von der Grundlinie vorgibt. Ansonsten funktioniert der `tnyref`-Parameter wie

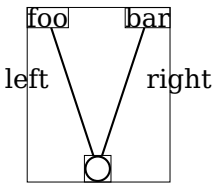
der `yref`-Parameter, der das Stück der Distanz von der Ober- zur Unterkante des Labels liefert.

# 8 Details

PSTricks ist recht zuverlässig, was die Platzierung der Knoten und das Anlegen einer Box betrifft, deren Größe der Bounding Box des Baumes ziemlich genau gleicht. Wie dem auch sei, PSTricks lässt die Knotenverbindungen oder Labels außer acht (mit Ausnahme der Baumknoten-Labels), wenn es die Bounding Boxes berechnet.
Wenn Sie aus diesen oder anderen Gründen eine Feinabstimmung für die Bounding Boxes der Knoten vornehmen wollen, können Sie die folgenden Parameter auf den entsprechenden Wert setzen:

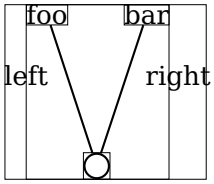
| *Name* | *Voreinstellung* |
|---|---|
| `bbl` | 0pt |
| `bbr` | 0pt |
| `bbh` | 0pt |
| `bbd` | 0pt |
| `xbbl` | 0pt |
| `xbbr` | 0pt |
| `xbbh` | 0pt |
| `xbbd` | 0pt |

Die „x“-Versionen vergrößern die Bounding Box um den jeweiligen Wert, die anderen setzen die Box auf den entsprechenden Wert. Es gibt einen Parameter für jede Richtung vom Mittelpunkt des Knotens aus: **l**eft, **r**ight, **h**eight (Höhe) und **d**epth (Tiefe).
Diese Parameter beeinflussen Bäume und Knoten sowie Unterbäume, die ihre Richtung ändern, jedoch keine Unterbäume, welche in die gleiche Richtung verlaufen, wie deren übergeordneter Baum (solche Unterbäume haben ein Profil anstelle einer Bounding Box und sollten über die Änderung der den einzelnen Knoten zugehörigen Bounding Boxes angepasst werden).
Heben Sie sich die Feineinstellungen der Bounding Box für den Schluss auf, wenn sie ansonsten mit dem Baum fertig sind.
Sie können die Bounding Box sehen, wenn sie den Parameter `showbbox=true/false` auf `true` setzen. Um die Bounding Boxes aller Knoten im Baum zu sehen, muss dieser Parameter vor dem Baum gesetzt werden.
Im folgenden Beispiel ragen die Labels aus der Bounding Box heraus:



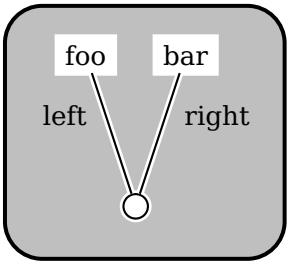

```
\psset{tpos=.6,showbbox=true}
\pstree[treemode=U]{\Tc{5pt}}{%
   \TR{foo}^{left}
   \TR{bar}_{right}}
```

So schaffen wir Abhilfe:




```
\psset{tpos=.6,showbbox=true}
\pstree[treemode=U,xbbl=8pt,xbbr=14pt]{\Tc{5pt}}{%
   \TR{foo}^{left}
   \TR{bar}_{right}}
```

Nun können wir den Baum umrahmen:

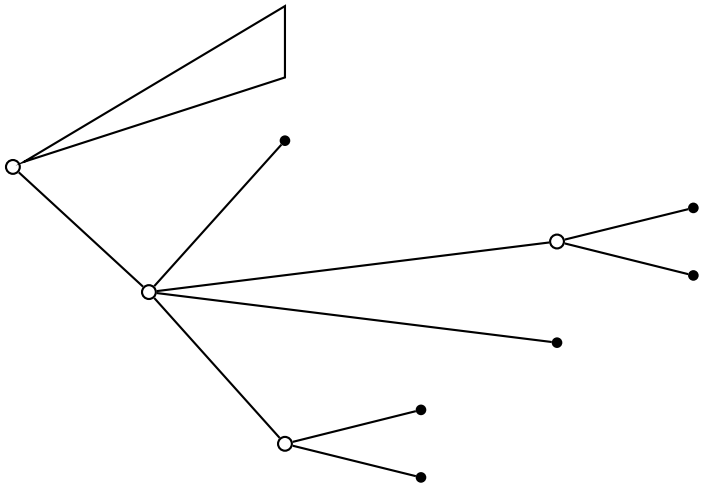



```
\psframebox[fillstyle=solid,fillcolor=lightgray,
    framesep=14pt,
 linearc=14pt,cornersize=absolute,linewidth=1.5pt]{
     %
 \psset{tpos=.6,border=1pt,nodesepB=3pt}
 \pstree[treemode=U,xbbl=8pt,xbbr=14pt]{%
  \Tc[fillcolor=white,fillstyle=solid]{5pt}}{%
  \TR*{foo}^{left}
  \TR*{bar}_{right}}}
```

Wir hätten das gleiche Ergebnis auch über das Ändern der Bounding Box der zwei Endknoten erhalten können.
Um Ebenen zu überspringen, verwendet man

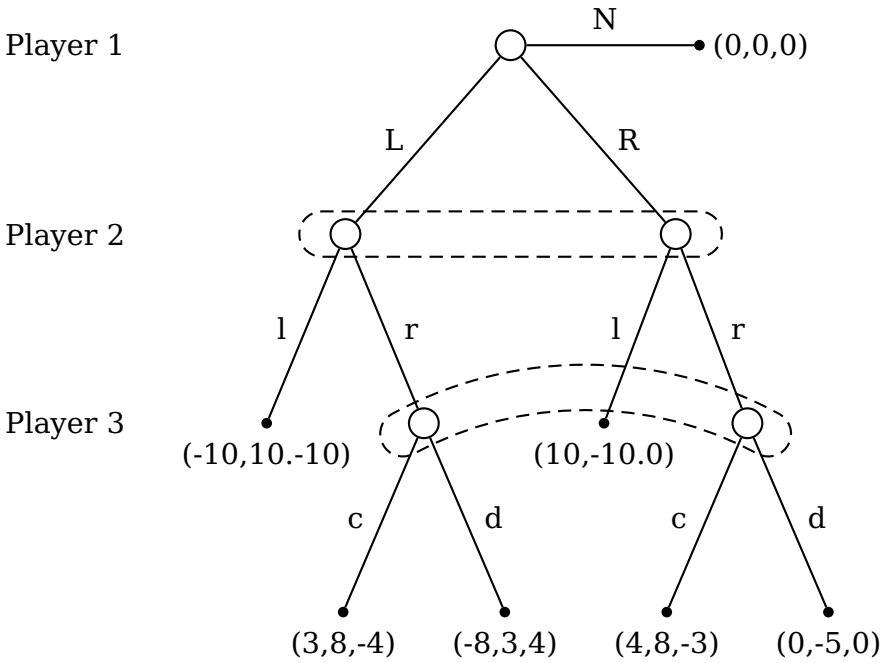
```
\skiplevel * [Options] {nodes or subtrees}
\skiplevels * [Options] {int}
    <nodes or subtrees>
\endskiplevels
```

Dies sind eine Art Unterbäume, nur ohne Wurzelknoten.



```
\pstree[treemode=R,levelsep=1.8,radius=2pt]{\Tc{3pt}}{%
 \skiplevel{\Tfan}
 \pstree{\Tc{3pt}}{%
  \TC*
  \skiplevels{2}
   \pstree{\Tc{3pt}}{\TC* \TC*}
   \TC*
  \endskiplevels
  \pstree{\Tc{3pt}}{\TC* \TC*}}}
```

Das Profil bei den fehlenden Ebenen ist das gleiche wie bei der ersten nicht fehlenden Ebene. Man kann dies über die Parameter der Bounding Box anpassen. Die maximale Kontrolle erhält man über den Einsatz verschachtelter `\skiplevel`-Befehle anstelle von `\skiplevels`.



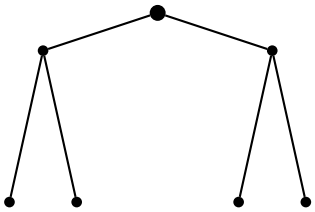

```
\large\psset{radius=6pt, dotsize=4pt}
\pstree[thislevelsep=0,edge=none,levelsep=2.5cm]{\Tn}{%
   \pstree{\TR{Player 1}}{\pstree{\TR{Player 2}}{\TR{Player 3}}}
   \psset{edge=\ncline}
   \pstree{\pstree[treemode=R]{\TC}{\Tdot ~{(0,0,0)} ^{N}}}{%
       \pstree{\TC[name=A] ^{L}}{%
         \Tdot ~{(-10,10.-10)} ^{l}
         \pstree{\TC[name=C] _{r}}{%
           \Tdot ~{(3,8,-4)} ^{c}
           \Tdot ~{(-8,3,4)} _{d}}}
       \pstree{\TC[name=B] _{R}}{%
         \Tdot ~{(10,-10.0)} ^{l}
         \pstree{\TC[name=D]_{r}}{%
           \Tdot ~{(4,8,-3)} ^{c}
           \Tdot ~{(0,-5,0)} _{d}}}}}
\ncbox[linearc=.3,boxsize=.3,linestyle=dashed,nodesep=.4]{A}{B}
\ncarcbox[linearc=.3,boxsize=.3,linestyle=dashed,arcangle=25,nodesep=.4]{D}{C}
```

# 9 Der Spielraum bei Parameteränderungen

`edge` ist der einzige Parameter, der, wenn im Parameterargument des Baumknotens gesetzt, das Zeichnen der Knotenverbindung beeinflusst (wenn man beispielsweise `nodesep` ändern möchte, muss die Linie die Parameteränderung beinhalten oder man muss sie vor den Knoten setzen).
Wie zu Beginn dieses Abschnitts angesprochen, beeinflussen mit `\pstree` vollzogene Parameteränderungen alle Unterbäume. Es gibt jedoch Varianten von einigen dieser

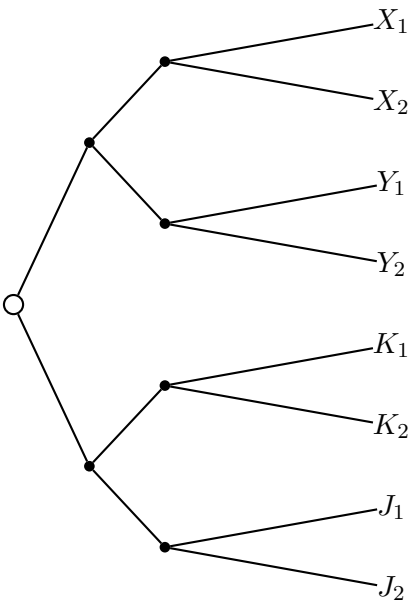
Parameter, um lokale Änderungen herbeizuführen, d. h. Änderungen, die nur die aktuelle Ebene betreffen: `thistreesep`, `thistreenodesize`, `thistreefit=tight/loose` und `thislevelsep`.

Zum Beispiel:



```
\pstree[thislevelsep=.5cm,thistreesep=2cm,
  radius=2pt]{\Tc*{3pt}}{%
  \pstree{\TC*}{\TC* \TC*}
  \pstree{\TC*}{\TC* \TC*}}
```

Es gibt einige Dinge (wie `levelsep`), die Sie vielleicht einheitlich für eine Baumebene einstellen wollen. Auf der Ebene `<n>` wird der Befehl `\pstreehook<roman(n)>` (z. B. `\pstreehookii`) ausgeführt, wenn er definiert ist (der Wurzelknoten des gesamten Baumes ist Ebene 0, die nachfolgenden Baumobjekte und die Knotenverbindungen vom Wurzelknoten zu diesen Nachfolgern ist Ebene 1 usw.). Im folgenden Beispiel wird `levelsep` für Ebene 2 geändert, ohne dabei den `thislevelsep`-Parameter für jeden der drei Unterbäume der zweiten Ebene gesondert einstellen zu müssen:




```
\[
\def\pstreehookiii{\psset{thislevelsep=3cm}}
\pstree[treemode=R,levelsep=1cm,radius=2pt]{\Tc{4pt}}{%
  \pstree{\TC*}{%
    \pstree{\TC*}{\Tr{X_1} \Tr{X_2}}
    \pstree{\TC*}{\Tr{Y_1} \Tr{Y_2}}}
  \pstree{\TC*}{%
    \pstree{\TC*}{\Tr{K_1} \Tr{K_2}}
    \pstree{\TC*}{\Tr{J_1} \Tr{J_2}}}}
\]
```

# 10 Liste aller optionalen Argumente für pst-tree

Der Standardwert wird eingestellt, wenn ein optionales Argument ohne Wert aufgerufen wird, z. B. ist \pstree[*levelsep*] das gleiche Argument wie \pstree[*levelsep=2cm*]. Alle booleschen Argumente sind per Voreinstellung false.

| Key |
|---|
| treefit |
| thistreefit |
| treemode |
| treesep |
| thistreesep |
| treenodesize |
| thistreenodesize |
| levelsep |
| thislevelsep |
| treeflip |
| showbbox |
| edge |
| bbl |
| bbr |
| bbh |
| bbd |
| xbbl |
| xbbr |
| xbbh |
| xbbd |
| fansize |
| tnsep |
| tnyref |
| tnheight |
| tndepth |
| tnpos |

# Literatur

[1] Denis Girou. Présentation de PSTricks. *Cahier GUTenberg*, 16:21–70, April 1994.

[2] Michel Goosens, Frank Mittelbach, Sebastian Rahtz, Dennis Roegel, and Herbert Voß. *The LaTeX Graphics Companion*. Addison-Wesley Publishing Company, Boston, Mass., second edition, 2007.

[3] Nikolai G. Kollock. *PostScript richtig eingesetzt: vom Konzept zum praktischen Einsatz*. IWT, Vaterstetten, 1989.

[4] Herbert Voß. *PSTricks – Grafik für TeX und LaTeX*. DANTE – Lehmanns, Heidelberg/Hamburg, sixth edition, 2010.

[5] Timothy Van Zandt. *multido.tex - a loop macro, that supports fixed-point addition*. CTAN:/macros/generic/multido.tex, 1997.

[6] Timothy Van Zandt and Denis Girou. Inside PSTricks. *TUGboat*, 15:239–246, September 1994.

# Index