

PSTricks

pst-tree

Nodes and Trees; v.1.10

January 25, 2009

Documentation by
Herbert Voß

Package author(s):
Timothy Van Zandt
Herbert Voß

Contents

1 Overview	4
2 Tree Nodes	4
3 Tree orientation	7
4 The distance between successors	8
5 Spacing between the root and successors	10
6 Edges	10
7 Edge and node labels	13
8 Details	15
9 The scope of parameter changes	18
10 List of all optional arguments for pst-tree	20
References	20

The `node` and `node connections` are perfect tools for making trees, but positioning the nodes using `\rput` would be rather tedious.¹ The file `pst-tree.tex/pstree.sty` contains a high-level interface for making trees.

It should be noted that the correct result is not guaranteed with every `dvips` driver. This package was written for Rokicki's `dvips` programme, which is practically part of every `TEX` distribution.

¹ Unless you have a computer program that generates the coordinates.

1 Overview

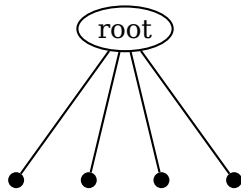
The tree commands are

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

T _E X version	L ^A T _E X version
\psTree{<root>}	\begin{psTree}{root}
<successors>	<successors>
\endpsTree	\end{psTree}

These do the same thing, but just have different syntax. `\psTree` is the “long” version. These macros make a box that encloses all the nodes, and whose baseline passes through the center of the root. Most of the nodes has a variant for use within a tree and are called tree nodes (see Section 2).

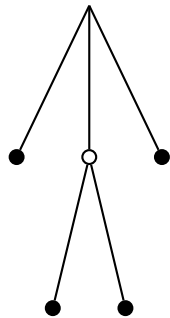
Trees and tree nodes are called *tree objects*. The *root* of a tree should be a single tree object, and the *successors* should be one or more tree objects. Here is an example with only nodes:



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

There is no difference between a terminal node and a root node, other than their position in the `\pstree{}` command.

Here is an example where a tree is included in the list of successors, and hence becomes subtree:



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

2 Tree Nodes

In each case, the name of the tree node is formed by omitting “node” from the end of the name and adding “T” at the beginning. For example, `\psovalnode` becomes `\Toval`. Here is the list of such tree nodes:

```

\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}

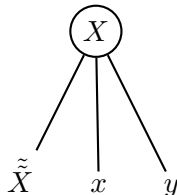
```

The syntax of a tree node is the same as of its corresponding “normal” node, except that:

- There is always an optional argument for setting graphics parameters, even if the original node did not have one;
- There is no argument for specifying the name of the node;
- There is never a coordinate argument for positioning the node; and
- To set the reference point with `\Tr`, set the `ref` parameter.

Figure 1 gives a reminder of what the nodes look like.

The difference between `\Tr` and `\TR` (variants of `\rnode` and `\Rnode`, respectively) is important with trees. Usually, you want to use `\TR` with vertical trees because the baselines of the text in the nodes line up horizontally. For example:

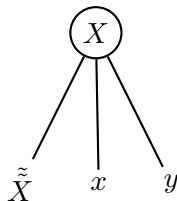


```

1 $
2 \pstree[nodesepB=3pt]{\Tcircle{X}}{%
3   \TR{\tilde{\tilde{X}}}
4   \TR{x}
5   \TR{y}}
6 $

```

Compare with this example, which uses `\Tr`:



```

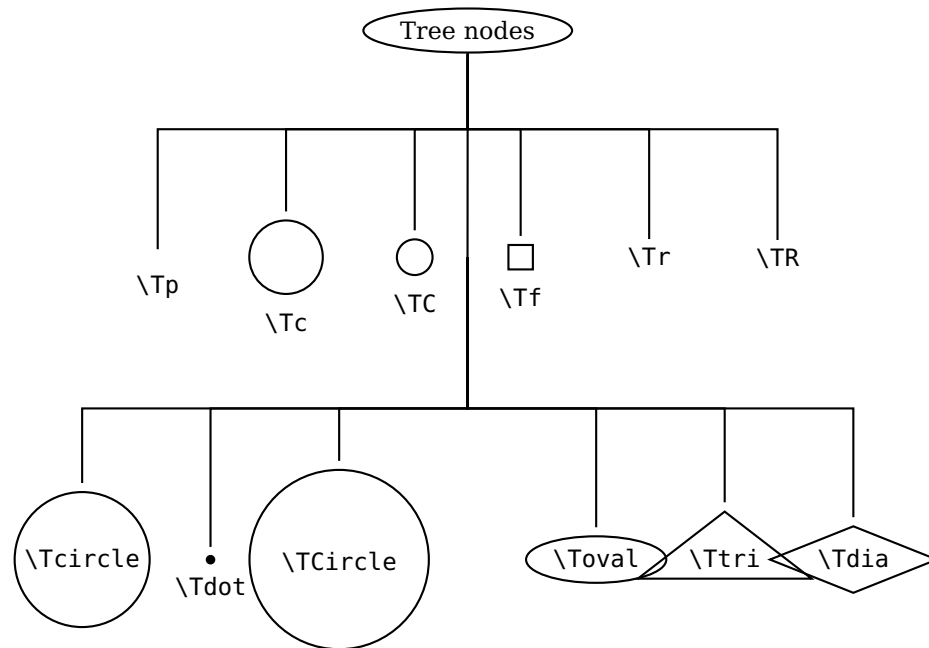
1 $
2 \pstree[nodesepB=3pt]{\Tcircle{X}}{%
3   \Tr{\tilde{\tilde{X}}}
4   \Tr{x}
5   \Tr{y}}
6 $

```

There is also a null tree node:

```
\Tn
```

It is meant to be just a place holder. Look at the tree in Figure page 6. The bottom row has a node missing in the middle. `\Tn{}` was used for this missing node.



```

1 \small
2 \psset{armB=1cm, levelsep=3cm, treesep=-3mm, angleB=-90, angleA=90, nodesepA=3pt}
3 \def\s#1{#1~{\tt\string#1}}\def\b#1{#1{\tt\string#1}}\def\psedge#1#2{\ncangle
4   {#2}{#1}}
5 \psTree[treenodesize=1cm]{\Toval{Tree nodes}}
6   \s\Tp\Tc{.5}~{\tt\string\Tc} \s\TC
7   \psTree[levelsep=4cm,armB=2cm]{\Tp[edge=\nc\line]}
8     \b\Tcircle \s\Tdot
9     \TCircle[radius=1.2]{\tt\string\TCircle}
10    \Tn \b\Toval \b\Ttri \b\Tdia
11  \endspsTree
12  \s\Tf \b\Tr \b\TR
13 \endspsTree

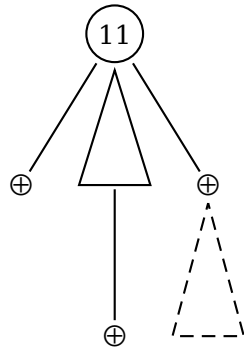
```

Figure 1: The tree nodes.

There is also a special tree node that doesn't have a "normal" version and that can't be used as the root node of a whole tree:

`\Tfan* [Options]`

This draws a triangle whose base is fansize and whose opposite corner is the predecessor node, adjusted by the value of nodesepA and offsetA. For example:



```

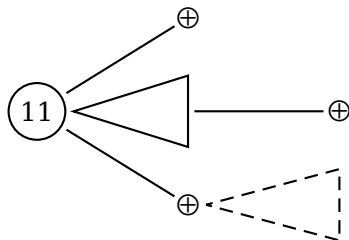
1 \pstree[dotstyle=oplus,dotsize=8pt,
2   nodesep=2pt]{\Tcircle{11}}{%
3   \Tdot
4   \pstree{\Tfan}{\Tdot}
5   \pstree{\Tdot}{\Tfan[linestyle=
6     dashed]}}

```

3 Tree orientation

Trees can grow down, up, right or left, depending on the `treemode= D, U, R, or L` parameter.

Here is what the previous example looks like when it grows to the right:

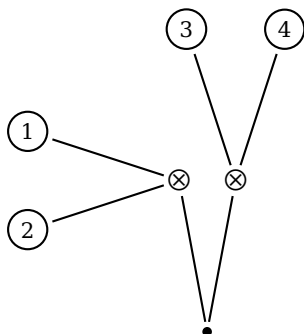


```

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

```

You can change the `treemode` in the middle of the tree. For example, here is a tree that grows up, and that has a subtree which grows to the left:

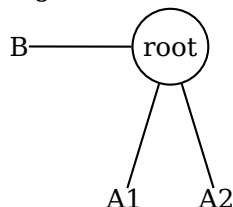


```

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

```

Since you can change a tree's orientation, it can make sense to include a tree (`<treeB>`) as a root node (of `<treeA>`). This makes a single logical tree, whose root is the root of `<treeB>`, and that has successors going off in different directions, depending on whether they appear as a successor to `<treeA>` or to `<treeB>`.

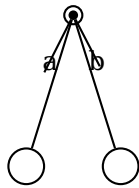


```

1 \pstree{\pstree[treemode=L]{\Tcircle{root}}{\
2   Tr{B}}}{%
3   \Tr{A1}
4   \Tr{A2}}

```

On a semi-related theme, note that any node that creates an LR-box can contain a tree. However, nested trees of this kind are not related in any way to the rest of the tree. Here is an example:

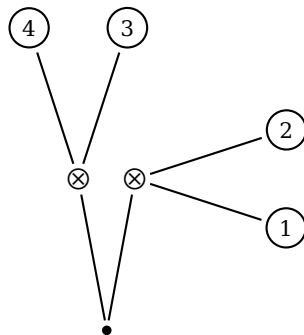


```

1 \psTree{\Tcircle{%
2     \pstree[treeseq=0.4,levelsep=0.6,
3         nodesepB=-6pt]{\Tdot}{%
4             \TR{a} \TR{b}}}}
5 \TC
6 \TC
7 \endpsTree

```

When the tree grows up or down, the successors are lined up from left to right in the order they appear in `\pstree`. When the tree grows to the left or right, the successors are lined up from top to bottom. As an afterthought, you might want to flip the order of the nodes. The keyword `treeflip=true/false` lets you do this. For example:



```

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

```

Note that I still have to go back and change the `treemode` of the subtree that used to grow to the left.

4 The distance between successors

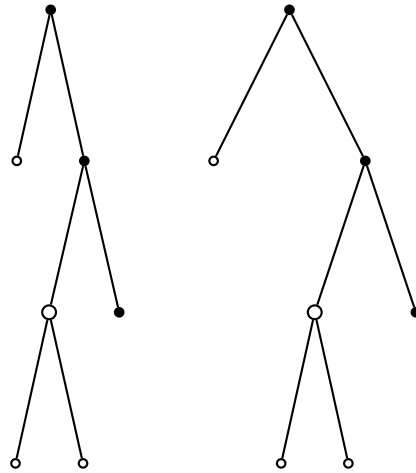
The distance between successors is set by the key `treeseq`. The rest of this section describes ways to fine-tune the spacing between successors. You can change the method for calculating the distance between subtrees by setting the `treefit=tight/loose` parameter. Here are the two methods:

tight When `treefit=tight`, which is the default, `treeseq` is the minimum distance between each of the levels of the subtrees.

loose When `treefit=loose`, `treeseq` is the distance between the subtrees' bounding boxes. Except when you have large intermediate nodes, the effect is that the horizontal distance (or vertical distance, for horizontal trees) between all the terminal nodes is the same (even when they are on different levels).²

Compare:

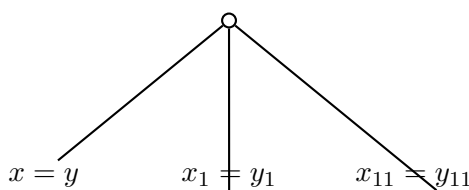
² When all the terminal nodes are on the same level, and the intermediate nodes are not wider than the base of their corresponding subtrees, then there is no difference between the two methods.



With `treefit=loose`, trees take up more space, but sometimes the structure of the tree is emphasized.

Sometimes you want the spacing between the centers of the nodes to be regular even though the nodes have different sizes. If you set `treenodesize` to a non-negative value, then PSTricks sets the width (or height+depth for vertical trees) to `treenodesize`, for the purpose of calculating the distance between successors.

For example, ternary trees look nice when they are symmetric, as in the following example:

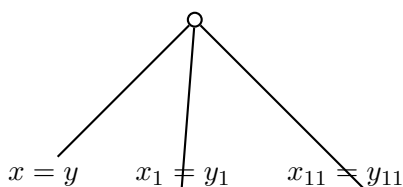


```

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

```

Compare with this example, where the spacing varies with the size of the nodes:

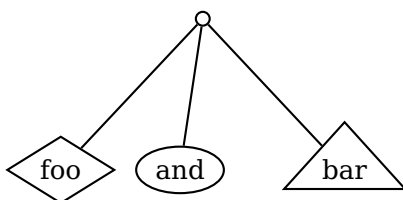


```

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

```

Finally, if all else fails, you can adjust the distance between two successors by inserting `\tspace{length}` between them:



```

1 \pstree{\Tc{3pt}}{%
2 \Tdia{foo}
3 \tspace{-0.5}
4 \Toval{and}
5 \Ttri{bar}}

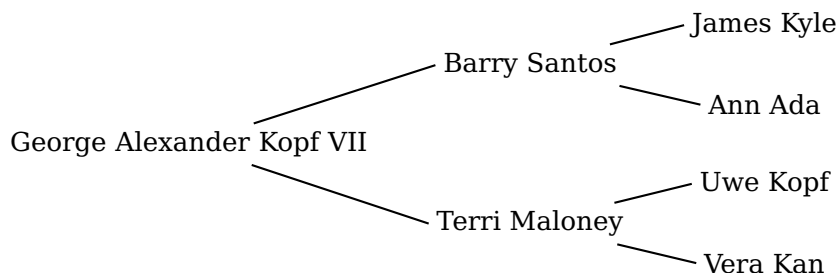
```

5 Spacing between the root and successors

The distance between the center lines of the tree levels is `levelsep`. If you want the spacing between levels to vary with the size of the levels, use the `*` convention. Then `levelsep` is the distance between the bottom of one level and the top of the next level (or between the sides of the two levels, for horizontal trees).

Note: PSTricks has to write some information to your `.aux` file if using \LaTeX , or to `\jobname.pst` otherwise, in order to calculate the spacing. You have to run your input file a few times before PSTricks gets the spacing right.

trees. Compare the following example:

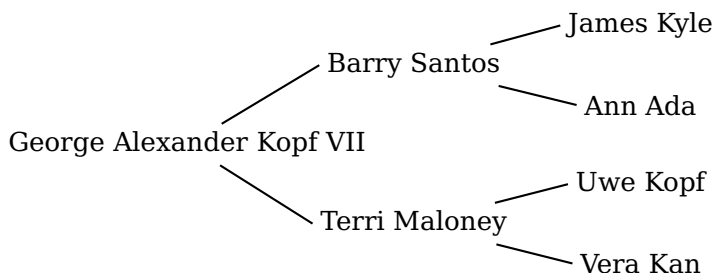


```

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

```

with this one, where the spacing between levels is fixed:



```

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

```

6 Edges

Right after you use a tree node command, `\pssucc` is equal to the name of the node, and `\pspred` is equal to the name of the node's predecessor. Therefore, you can draw a line between the node and its predecessor by inserting, for example,

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

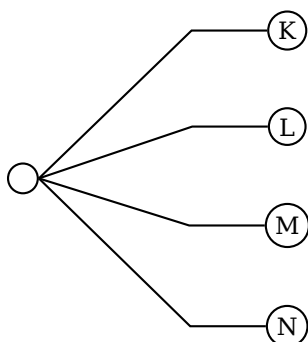
To save you the trouble of doing this for every node, each tree node executes

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

The default definition of `\psedge` is `\ncline`, but you can redefine it as you please with `\def` or L^AT_EX's `\renewcommand`.

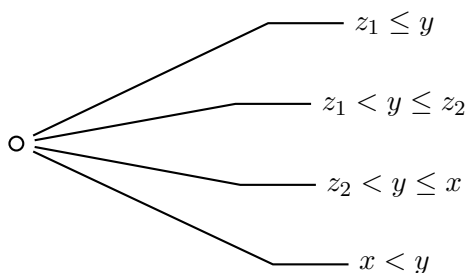
For example, here I use `\ncdiag`, with `armA=0`, to get all the node connections to emanate from the same point in the predecessor. L^AT_EX users can instead type:

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



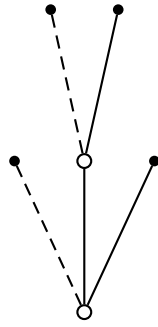
```
1 \def\psedge{\ncdiag[armA=0,angleB=180,armB=1cm
2 ]}
3 \pstree[treemode=R,levelsep=3.5cm,framesep=2pt
4 ]{\Tc{6pt}}{\%
5 \small \Tcircle{K} \Tcircle{L} \Tcircle{M}
6 \Tcircle{N}}
```

Here is an example with `\ncdiagg`. Note the use of a negative `armA` value so that the corners of the edges are vertically aligned, even though the nodes have different sizes:



```
1 $
2 \def\psedge#1#2{\ncdiagg[angleA=180,armA=1cm,nodesep=4pt]{#2}{#1}}
3 % Or: \renewcommand{\psedge}[2]{ ... }
4 \pstree[treemode=R, levelsep=5cm]{\Tc{3pt}}{\%
5 \Tr{z_1\leq y} \Tr{z_1<y\leq z_2} \Tr{z_2<y\leq x} \Tr{x<y}
6 }
7 $
```

Another way to define `\psedge{}` is with the edge parameter. Be sure to enclose the value in braces "" if it contains commas or other parameter delimiters. This gets messy if your command is long, and you can't use arguments like in the preceding example, but for simple changes it is useful. For example, if I want to switch between a few node connections frequently, I might define a command for each node connection, and then use the edge parameter.



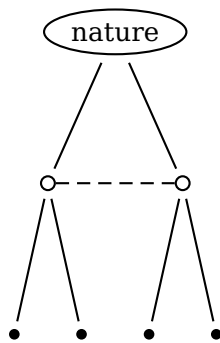
```

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

```

You can also set `edge=none` to suppress the node connection.

If you want to draw a node connection between two nodes that are not direct predecessor and successor, you have to give the nodes a name that you can refer to, using the name parameter. For example, here I connect two nodes on the same level:

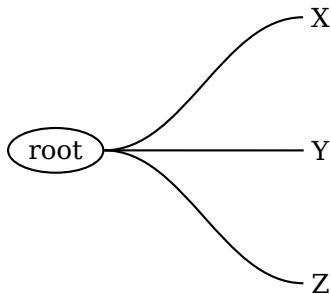


```

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

```

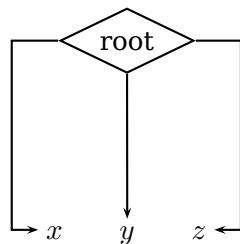
We conclude with the more examples.



```

1 \def\psedge{\nccurve[angleB=180, nodesepB=3pt
2   ]}
3 \pstree[treemode=R, treesep=1.5, levelsep=3.5]
4   %
5   {\Toval{root}}{\Tr{X} \Tr{Y} \Tr{Z}}

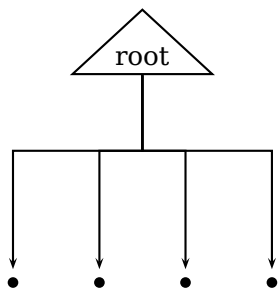
```



```

1 \pstree[nodesepB=3pt, arrows=->, xlbl=15pt,
2   xbr=15pt, levelsep=2.5cm]{\Tdia{root}}{%
3   $
4   \TR[edge={\ncbar[angle=180]}]{x}
5   \TR{y}
6   \TR[edge=\ncbar]{z}
7   $}

```



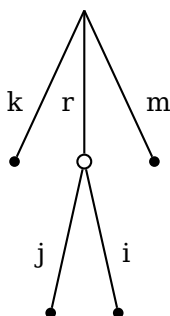
```

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

```

7 Edge and node labels

Right after a node, an edge has typically been drawn, and you can attach labels using `\ncput`, `\tlput`, etc. With `\tlput`, `\trput`, `\taput`, and `\tbput`, you can align the labels vertically or horizontally, just like the nodes. This can look nice, at least if the slopes of the node connections are not too different.



```

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

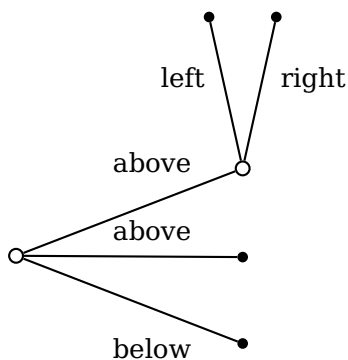
```

Within trees, the `tpos` parameter measures this distance from the predecessor to the successor, whatever the orientation of the tree. (Outside of trees it measures the distance from the top to bottom or left to right nodes.)

PSTricks also sets `shortput=tab` within trees. This is a special `shortput` option that should not be used outside of trees. It implements the following abbreviations, which depend of the orientation of the tree:

	Short for:	
Char.	Vert.	Horiz.
^	<code>\tlput</code>	<code>\taput</code>
-	<code>\trput</code>	<code>\tbput</code>

(The scheme is reversed if `treeflip=true`.)



```

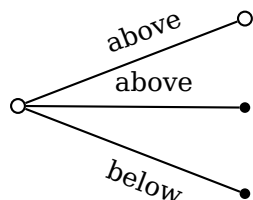
1 \psset{tpos=.6}
2 \pstree[treemode=R, thistreesep=1cm,
3   thislevelsep=3cm, radius=2pt]{\Tc{3pt}}{%
4   \pstree[treemode=U, xbr=20pt]{\Tc{3pt}^
5     above}}{%
6     \TC*^{left}
7     \TC*_{right}}
8 \TC*^{above}
9 \TC*_{below}

```

You can change the character abbreviations with

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

The `\n*put` commands can also give good results:



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

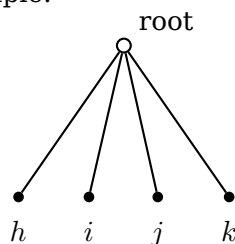
You can put labels on the nodes using `\nput`. However, `\pstree` won't take these labels into account when calculating the bounding boxes.

There is a special node label option for trees that does keep track of the bounding boxes:

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

Call this a “tree node label”.

Put a tree node label right after the node to which it applies, before any node connection labels (but node connection labels, including the short forms, can follow a tree node label). The label is positioned directly below the node in vertical trees, and similarly in other trees. For example:



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

Note that there is no “long form” for this tree node label. However, you can change the single character used to delimit the label with

```
\MakeShortTnput{<char1>}
```

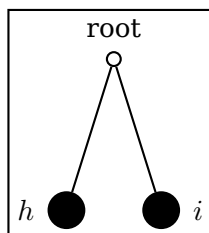
If you find it confusing to use a single character, you can also use a command sequence. E.g.,

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

You can have multiple labels, but each successive label is positioned relative to the bounding box that includes the previous labels. Thus, the order in which the labels are placed makes a difference, and not all combinations will produce satisfactory results.

You will probably find that the tree node label works well for terminal nodes, without your intervention. However, you can control the tree node labels by setting several parameters.

To position the label on any side of the node ("l"eft, "r"ight, "a"bove or "b"elow), set: `tnpos=l/r/a/b`



```

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

When you leave the argument empty, which is the default, PSTricks chooses the label position is automatically.

To change the distance between the node and the label, set `tnsep` to a dimension. When you leave the argument empty, which is the default, PSTricks uses the value of `labelsep`. When the value is negative, the distance is measured from the center of the node.

When labels are positioned below a node, the label is given a minimum height of `tnheight`. Thus, if you add labels to several nodes that are horizontally aligned, and if either these nodes have the same depth or `tnsep` is negative, and if the height of each of the labels is no more than `tnheight`, then the labels will also be aligned by their baselines. The default is `\ht\strutbox`, which in most T_EX formats is the height of a typical line of text in the current font. Note that the value of `tnheight` is not evaluated until it is used.

The positioning is similar for labels that go below a node. The label is given a minimum *depth* of `tndepth`. For labels positioned above or below, the horizontal reference point of the label, i.e., the point in the label directly above or below the center of the node, is set by the `href` parameter.

When labels are positioned on the left or right, the right or left edge of the label is positioned distance `tnsep` from the node. The vertical point that is aligned with the center of the node is set by `tnyref`. When you leave this empty, `vref` is used instead. Recall that `vref` gives the vertical *distance* from the baseline. Otherwise, the `tnyref` parameter works like the `yref` parameter, giving the fraction of the distance from the bottom to the top of the label.

8 Details

PSTricks does a pretty good job of positioning the nodes and creating a box whose size is close to the true bounding box of the tree. However, PSTricks does not take into account the node connections or labels when calculating the bounding boxes, except the tree node labels.

If, for this or other reasons, you want to fine tune the bounding box of the nodes, you can set the following parameters to a dimension:

<i>name</i>	<i>default</i>
bbL	0pt
bbR	0pt
bbH	0pt
bbD	0pt
xbbL	0pt
xbbR	0pt
xbbH	0pt
xbbD	0pt

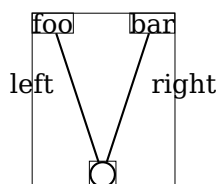
The "x" versions increase the bounding box by <dim>, and the others set the bounding box to the dimension. There is one parameter for each direction from the center of the node, **left**, **right**, **height**, and **depth**.

These parameters affect trees and nodes, and subtrees that switch directions, but not subtrees that go in the same direction as their parent tree (such subtrees have a profile rather than a bounding box, and should be adjusted by changing the bounding boxes of the constituent nodes).

Save any fiddling with the bounding box until you are otherwise finished with the tree.

You can see the bounding boxes by setting the `showbbox=true/false` parameter to true. To see the bounding boxes of all the nodes in a tree, you have to set this parameter before the tree.

In the following example, the labels stick out of the bounding box:

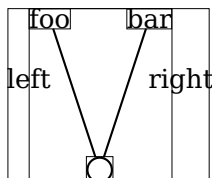


```

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

```

Here is how we fix it:

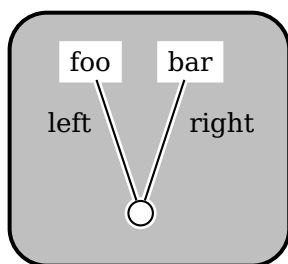


```

1 \psset{tpos=.6, showbbox=true}
2 \pstree[treemode=U, xbbL=8pt, xbbR=14pt]{\Tc{5pt}}{%
3   \TR{foo}^{left}
4   \TR{bar}_{right}}

```

Now we can frame the tree:



```

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

```

We would have gotten the same result by changing the bounding box of the two terminal nodes.

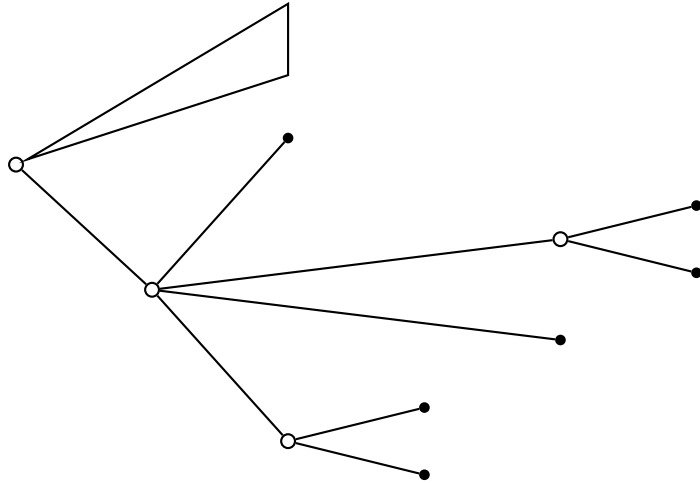
To skip levels, use


```

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

```

These are kind of like subtrees, but with no root node.

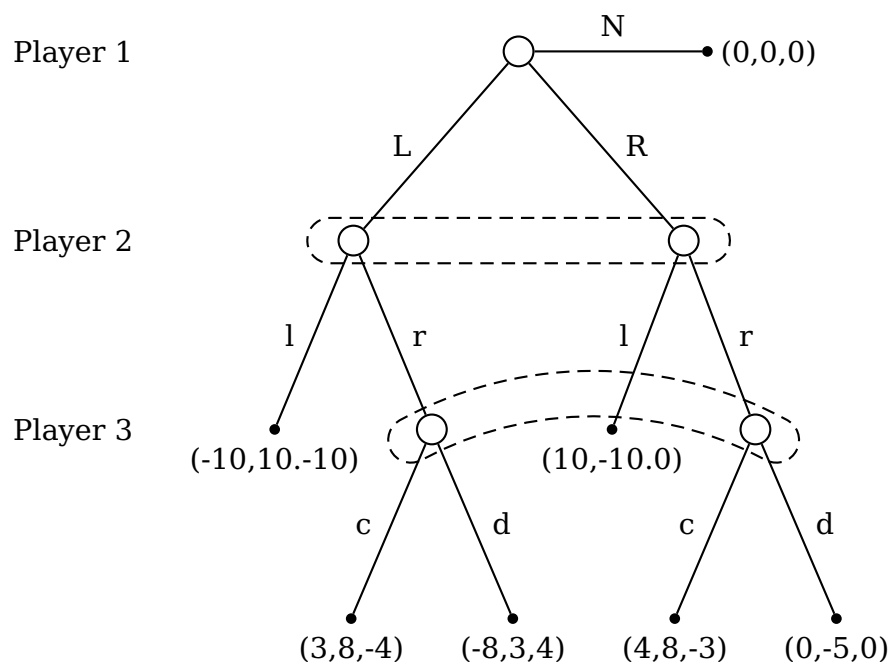


```

1 \pstree[treemode=R, levelsep=1.8, radius=2pt]{\Tc{3pt}}{%
2   \skiplevel{\Tfan}
3   \pstree{\Tc{3pt}}{%
4     \TC*
5     \skiplevels{2}
6     \pstree{\Tc{3pt}}{\TC* \TC*}
7     \TC*
8   \endskiplevels
9   \pstree{\Tc{3pt}}{\TC* \TC*}}

```

The profile at the missing levels is the same as at the first non-missing level. You can adjust this with the bounding box parameters. You get greatest control if you use nested `\skiplevel` commands instead of `\skiplevels`.



```

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

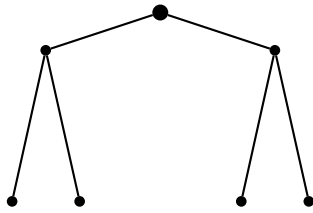
```

9 The scope of parameter changes

edge is the only parameter which, when set in a tree node's parameter argument, affects the drawing of the node connection (e.g., if you want to change the nodesep, your edge has to include the parameter change, or you have to set it before the node).

As noted at the beginning of this section, parameter changes made with `\pstree` affect all subtrees. However, there are variants of some of these parameters for making local changes, i.e. changes that affects only the current level: `thisreesep`, `thisreenodesize`, `thisreefit=tight/loose`, and `thislevelsep`.

For example:

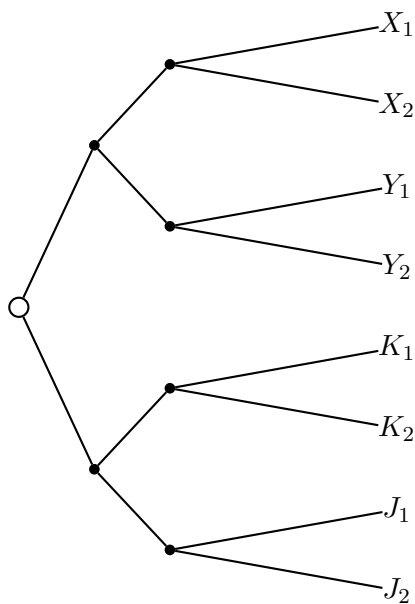


```

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

```

There are some things you may want set uniformly across a level in the tree, such as the `levelsep`. At level $\langle n \rangle$, the command `\pstreehook<roman(n)>` (e.g., `\pstreehookii`) is executed, if it is defined (the root node of the whole tree is level 0, the successor tree objects and the node connections from the root node to these successors is level 1, etc.). In the following example, the `levelsep` is changed for level 2, without having to set the `thislevelsep` parameter for each of the three subtrees that make of level 2:



```

1 \[
2 \def\pstreehookiii{\psset{thislevelsep=3cm}}
3 \pstree[treemode=R,levelsep=1cm,radius=2pt]{\Tc{4pt}}{%
4   \pstree{\Tc*}{%
5     \pstree{\Tc*}{\Tr{X_1} \Tr{X_2}}
6     \pstree{\Tc*}{\Tr{Y_1} \Tr{Y_2}}}
7   \pstree{\Tc*}{%
8     \pstree{\Tc*}{\Tr{K_1} \Tr{K_2}}
9     \pstree{\Tc*}{\Tr{J_1} \Tr{J_2}}}}
10 \]

```

10 List of all optional arguments for pst-tree

The default value ist set when an optional argument is called without any value, e. g. `\pstree[levelsep]` is the same as `\pstree[levelsep=2cm]`. All boolean arguments are preset to false.

Key	Type	Default
<code>treefit</code>	ordinary	<code>tight</code>
<code>thistreefit</code>	ordinary	<code>tight</code>
<code>treemode</code>	ordinary	<code>0</code>
<code>treeseq</code>	ordinary	<code>0.75cm</code>
<code>thistreeseq</code>	ordinary	<code>[none]</code>
<code>treenodesize</code>	ordinary	<code>-1pt</code>
<code>thistreenodesize</code>	ordinary	<code>-1pt</code>
<code>levelsep</code>	ordinary	<code>2cm</code>
<code>thislevelsep</code>	ordinary	<code>[none]</code>
<code>treeflip</code>	boolean	<code>true</code>
<code>showbbox</code>	boolean	<code>true</code>
<code>edge</code>	ordinary	<code>\ncline</code>
<code>ttl</code>	ordinary	<code>[none]</code>
<code>tbr</code>	ordinary	<code>[none]</code>
<code>tbb</code>	ordinary	<code>[none]</code>
<code>tbd</code>	ordinary	<code>[none]</code>
<code>xttl</code>	ordinary	<code>[none]</code>
<code>xtbr</code>	ordinary	<code>[none]</code>
<code>xtbb</code>	ordinary	<code>[none]</code>
<code>xtbd</code>	ordinary	<code>[none]</code>
<code>fansize</code>	ordinary	<code>1cm</code>
<code>tnsep</code>	ordinary	
<code>tnyref</code>	ordinary	
<code>tnheight</code>	ordinary	<code>\ht \strutbox</code>
<code>tndepth</code>	ordinary	<code>\dp \strutbox</code>
<code>tnpos</code>	ordinary	

References

- [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 L^AT_EX 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 T_EX und L^AT_EX*. DANTE – Lehmanns, Heidelberg/Hamburg, fifth edition, 2008.

-
- [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

Symbols

~, 14

A

a, 15

armA, 11

.aux, 10

B

b, 15

bbd, 16

bbh, 16

bbL, 16

bbr, 16

D

D, 7

\def, 11

dvips, 3

E

edge, 11, 12, 18

\endpsTree, 4

\endskiplevels, 17

Environment

– psTree, 4

Extension

– .aux, 10

F

fansize, 6

H

href, 15

J

\jobname, 10

K

Keyvalue

– a, 15

– b, 15

– D, 7

– L, 7

– l, 15

– loose, 8, 18

– R, 7

– r, 15

– tight, 8, 18

– U, 7

Keyword

– armA, 11

– bbd, 16

– bbh, 16

– bbL, 16

– bbr, 16

– edge, 11, 12, 18

– fansize, 6

– href, 15

– labelsep, 15

– levelsep, 10, 19

– name, 12

– nodesep, 18

– nodesepA, 6

– offsetA, 6

– ref, 5

– shortcut, 13

– showbbox, 16

– thislevelsep, 18, 19

– thistreefit, 18

– thistreenodesize, 18

– thistreesep, 18

– tndepth, 15

– tnheight, 15

– tnpos, 15

– tnsep, 15

– tnyref, 15

– tpos, 13

– treefit, 8, 9

– treeflip, 8, 13

– treemode, 7, 8

– treemode=, 7

– treenodesize, 9

– treesep, 8

– vref, 15

– xbbd, 16

– xbbh, 16

– xbbL, 16

– xbbR, 16

– yref, 15

L

L, 7
 l, 15
 labelsep, 15
 levelsep, 10, 19
 loose, 8, 9, 18

M

Macro

- \def, 11
 - \endpsTree, 4
 - \endskipllevels, 17
 - \jobname, 10
 - \MakeShortTab, 14
 - \MakeShortTinput, 14
 - \ncdiag, 11
 - \ncdiagg, 11
 - \ncline, 11
 - \ncput, 13
 - \nput, 14
 - \psedge, 11
 - \psovalnode, 4
 - \pspred, 10
 - \pssucc, 10
 - \psTree, 4
 - \pstree, 4, 8, 14, 18, 20
 - \pstreehookii, 19
 - \renewcommand, 11
 - \Rnode, 5
 - \rnode, 5
 - \rput, 3
 - \skipllevel*, 17
 - \skipllevel, 17
 - \skipllevels*, 17
 - \skipllevels, 17
 - \strutbox, 15
 - \taput, 13
 - \tbput, 13
 - \TC*, 5
 - \Tc*, 5
 - \TCircle*, 5
 - \Tcircle*, 5
 - \Tdia*, 5
 - \Tdot*, 5
 - \Tf*, 5
 - \Tfan*, 6
 - \tlput, 13

- \Tn, 5
 - \Toval*, 5
 - \Toval, 4
 - \Tp*, 5
 - \TR*, 5
 - \TR, 5
 - \Tr*, 5
 - \Tr, 5
 - \trput, 13
 - \tspace, 9
 - \Ttri*, 5
 \MakeShortTab, 14
 \MakeShortTinput, 14

N

name, 12
 \ncdiag, 11
 \ncdiagg, 11
 \ncline, 11
 \ncput, 13
 nodesep, 18
 nodesepA, 6
 none, 12
 \nput, 14

O

offsetA, 6

P

Program

- dvips, 3
 \psedge, 11
 \psovalnode, 4
 \pspred, 10
 \pssucc, 10
 \psTree, 4
 psTree, 4
 \pstree, 4, 8, 14, 18, 20
 \pstreehookii, 19

R

R, 7
 r, 15
 ref, 5
 \renewcommand, 11
 \Rnode, 5
 \rnode, 5
 Rokicki, 3

\rput, 3

S

shortput, 13

showbbox, 16

\skiplevel, 17

\skiplevel*, 17

\skiplevels, 17

\skiplevels*, 17

\strutbox, 15

subtree, 4, 8

Syntax

– ~, 14

T

tab, 13

\taput, 13

\tbput, 13

\TC*, 5

\Tc*, 5

\TCircle*, 5

\Tcircle*, 5

\Tdia*, 5

\Tdot*, 5

\Tf*, 5

\Tfan*, 6

thislevelsep, 18, 19

thisreefit, 18

thisreenodesize, 18

thisreeseq, 18

tight, 8, 18

\tlput, 13

\Tn, 5

tndepth, 15

tnheight, 15

tnpos, 15

tnsep, 15

tnyref, 15

\Toval, 4

\Toval*, 5

\Tp*, 5

tpos, 13

\TR, 5

\Tr, 5

\TR*, 5

\Tr*, 5

tree objects, 4

treefit, 8, 9

treeflip, 8, 13

treemode, 7, 8

treemode=, 7

treenodesize, 9

treeseq, 8

\trput, 13

true, 13

\tspace, 9

\Ttri*, 5

U

U, 7

V

Value

– loose, 8, 9

– none, 12

– tab, 13

– tight, 8

– true, 13

vref, 15

X

xbbd, 16

xbbh, 16

xbbl, 16

xbbr, 16

Y

yref, 15