1 Dots

The graphics objects

> **\psdot***[*par*](*x1*,*y1*)
> **\psdots***[*par*]**(*x1*,*y1*)**(*x2*,*y2*)…(*xn*,*yn*)

put a dot at each coordinate.

What a “dot” is depends on the value of the

> **dotstyle=*style*** **Default: ***

parameter. This also determines the dots you get when **showpoints=true**.

The dot styles are also pretty intuitive:

Style	*Example*	*Style*	*Example*
*	● ● ● ● ●	square	□ □ □ □ □
o	○ ○ ○ ○ ○	square*	■ ■ ■ ■ ■
+	+ + + + +	diamond	♦ ♦ ♦ ♦ ♦
x	× × × × ×	diamond*	◊ ◊ ◊ ◊ ◊
asterisk	∗ ∗ ∗ ∗ ∗	triangle	△ △ △ △ △
oplus	⊕ ⊕ ⊕ ⊕ ⊕	triangle*	▲ ▲ ▲ ▲ ▲
otimes	⊗ ⊗ ⊗ ⊗ ⊗	pentagon	⬠ ⬠ ⬠ ⬠ ⬠
|	| | | | |	pentagon*	⬟ ⬟ ⬟ ⬟ ⬟

Except for diamond, the center of dot styles with a hollow center is colored **fillcolor**.

Here are the parameters for changing the size and orientation of the dots:

dotsize=*dim* ‘*num*’ **Default: 2pt 2**

> The diameter of a circle or disc is *dim* plus *num* times **linewidth** (if the optional *num* is included). The size of the other dots styles is similar (except for the size of the | dot style, which is set by the **tbarsize** parameter described on page **??**).

dotscale=*num1* ‘*num2*’ **Default: 1**

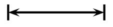
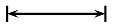
> The dots are scaled horizontally by *num1* and vertically by *num2*. If you only include *num1*, the arrows are scaled by *num1* in both directions.

dotangle=*angle* **Default: 0**

After setting the size and scaling the dots, the dots are rotated by *angle*.

2 Arrowheads and such

New arrows:

Value	*Example*	*Name*
|<->|		T-bars and arrowheads.
|<*->|*		T-bars and arrowheads, flush.

The size of these line terminators is controlled by the following parameters. In the description of the parameters, the width always refers to the dimension perpendicular to the line, and length refers to a dimension in the direction of the line.

arrowsize=*dim ‘num’* **Default: 1.5pt 2**

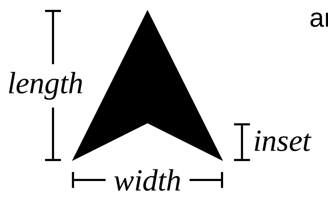
The width of arrowheads is *dim* plus *num* times **linewidth** (if the optional ‘num’ is inclued). See the diagram below.

arrowlength=*num* **Default:**

Length of arrowheads, as a fraction of the width, as shown below.

arrowinset=*num* **Default:**

Size of inset for arrowheads, as a fraction of the length, as shown below.



arrowsize	=	*dim num*
width	=	*num* x **linewidth** + *dim1*
length	=	**arrowlength** x *width*
inset	=	**arrowinset** x *height*

tbarsize=*dim ‘num’* **Default:**

The width of a t-bar, square bracket or rounded bracket is *dim* plus *num* times **linewidth** (if the optional ‘num’ is included).

bracketlength=*num* **Default:**

The height of a square bracket is *num* times its width.

rbracketlength=*num* **Default:**

The height of a round bracket is *num* times its width.

arrowscale=*arrowscale=num1 'num2'* **Default:**

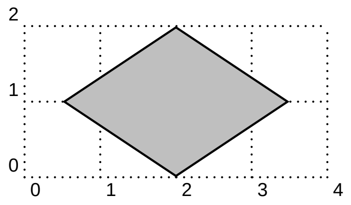
Imagine that arrows and such point down. This scales the width of the arrows by *num1* and the length (height) by *num2*. If you only include one number, the arrows are scaled the same in both directions. Changing **arrowscale** can give you special effects not possible by changing the parameters described above. E.g., you can change the width of lines used to draw brackets.

The size of dots is controlled by the **dotsize** parameter.

3 Lines and polygons

\psdiamond*[*par*](*x0*,*y0*)**(*x1*,*y1*)**

\psdiamond draws a diamond centered at (*x0*,*y0*), and with the half width and height equal to *x1* and *y1*, respectively.



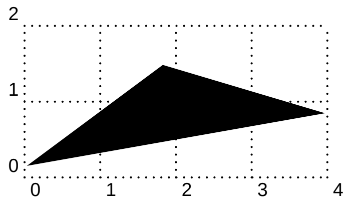
```
\psdiamond[framearc=.3,fillstyle=solid,
    fillcolor=lightgray](2,1)(1.5,1)
```

The diamond is rotated about the center by

gangle=*gangle* **Default: 0**

\pstriangle*[*par*](*x0*,*y0*)**(*x1*,*y1*)**

\pstriangle draws an isosceles triangle with the base centered at (*x0*,*y0*), and with width (base) and height equal to *x1* and *y1*, respectively.

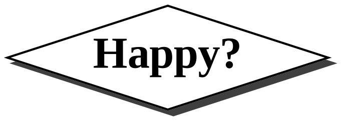


```
\pstriangle*[gangle=10](2,.5)(4,1)
```

4 Framed boxes

\psdiabox*[*par*]**{*stuff*}**

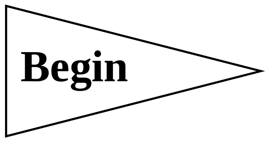
\psdiabox draws a diamond.



```
\psdiabox[shadow=true]{\Large\bf  Happy?}
```

\pstribox*[*par*]**{*stuff*}**

\pstribox draws a triangle.

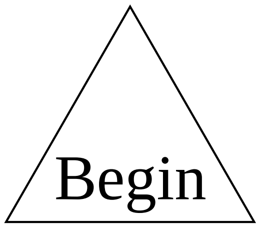


```
\pstribox[trimode=R,framesep=5pt]{\Large\bf  Begin}
```

The triangle points in the direction:

trimode=*U/D/R/L* **Default: U**

If you include the optional *, then an equilateral triangle is drawn, otherwise, you get the minimum-area isosceles triangle.

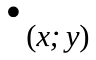


```
\pstribox[trimode=*U]{\Huge  Begin}
```

5 Obsolete put commands

There is an obsolete command **\Rput** that has the same syntax as **\uput** and that works almost the same way, except the *refangle* argument has the syntax of **\rput**'s *refpoint* argument, and it gives the point in *stuff* that should be aligned with (x,y). E.g.,

```
\qdisk(4,0){2pt}
\Rput[tl](4,0){$(x,y)$}
```



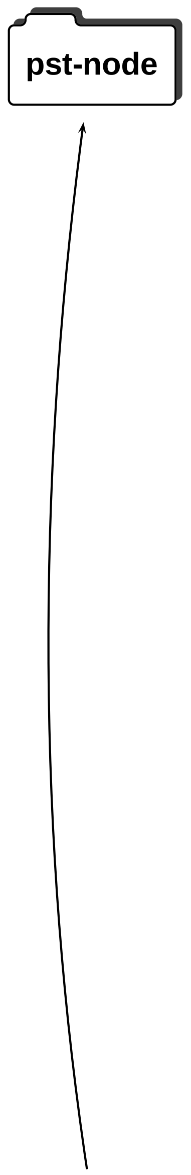
Here is the equivalence between **\uput**'s *refangle* abbreviations and **\Rput**'s *refpoint* abbreviations:

\uput	r	u	l	d	ur	ul	dr	dl
\Rput	l	b	r	t	bl	br	tr	rl

Some people prefer **\Rput**'s convention for specifying the position of *stuff* over **\uput**'s.

Once upon a time there was \psput instead of **\nput**. This feature is still supported for backwards compatibility.

Nodes and Node Connections



All the commands described in this part are contained in the file pst-node.tex/pst-node.sty.

The node and node connection macros let you connect information and place labels, without knowing the exact position of what you are connecting or where the lines should connect. These macros are useful for making graphs and trees, mathematical diagrams, linguistic syntax diagrams, and connecting ideas of any kind. They are the trickiest tricks in PSTricks!

There are three components to the node macros:

Node definitions The node definitions let you assign a name and shape to an object. See Section 6.

Node connections The node connections connect two nodes, identified by their names. See Section 7.

Node labels The node label commands let you affix labels to the node connections. See Section 8.

You can use these macros just about anywhere. The best way to position them depends on the application. For greatest flexibility, you can use the nodes in a **\pspicture**, positioning and rotating them with **\rput**. You can also use them in alignment environments. pst-node.tex contains a special alignment environment, **\psmatrix**, which is designed for positioning nodes in a grid, such as in mathematical diagrams and some graphs. **\psmatrix** is described in Section **??**. pst-node.tex also contains high-level macros for trees. These are described in Part **??**.

But don't restrict yourself to these more obvious uses. For example:

I made the file symbol a node. Now I can draw an arrow so that you know what I am talking about.

```
\rnode{A}{%
  \parbox{4cm}{\raggedright
    I made the file symbol a node. Now I can draw an
    arrow so that you know what I am talking about.}}
\ncarc[nodesep=8pt]{->}{A}{file}
```

6 Nodes

Nodes have a name. a boundary and a center.

The name is for refering to the node when making node connections and labels. You specify the name as an argument to the node commands. The name must contain only letters and numbers, and must begin with a letter. Bad node names can cause PostScript errors.

The center of a node is where node connections point to. The boundary is for determining where to connect a node connection. The various nodes differ in how they determine the center and boundary. They also differ in what kind of visable object they create.

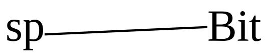
Here are the nodes:

\rnode[*refpoint*]***{name}{stuff}***

\rnode puts *stuff* in a box. The center of the node is *refpoint*, which you can specify the same way as for **\rput**.

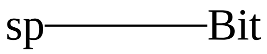
\Rnode*[*par*]***{name}{stuff}***

\Rnode also makes a box, but the center is set differently. If you align **\rnode**'s by their baseline, differences in the height and depth of the nodes can cause connecting lines to be not quite parallel, such as in the following example:



```
\Large
\rnode{A}{sp} \hskip 2cm \rnode{B}{Bit}
\ncline{A}{B}
```

With **\Rnode**, the center is determined relative to the baseline:



```
\Large
\Rnode{A}{sp} \hskip 2cm \Rnode{B}{Bit}
\ncline{A}{B}
```

You can usually get by without fiddling with the center the center of the node, but to modify it you set the

href=*num*	**Default: 0**
vref=*dim*	**Default: .7ex**

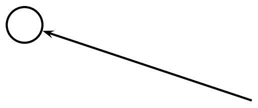
parameters. In the horizontal direction, the center is located fraction **href** from the center to the edge. E.g, if **href=-1**, the center is on the left edge of the box. In the vertical direction, the center is located distance **vref** from the baseline. The **vref** parameter is evaluated each time **\Rnode** is used, so that you can use ex units to have the distance adjust itself to the size of the current font (but without being sensitive to differences in the size of letters within the current font).

\pnode(*x*,*y*)***{name}***

This creates a zero dimensional node at (*x*,*y*).

\cnode*[*par*](*x*,*y*)***{radius}{name}***

This draws a circle. Here is an example with **\pnode** and **\cnode**:



```
\cnode(0,1){.25}{A}
\pnode(3,0){B}
\ncline{<-}{A}{B}
```

\Cnode*[*par*](*x*,*y*)***{name}***

This is like **\cnode**, but the radius is the value of

radius=*dim* **Default: 2pt**

This is convenient when you want many circle nodes of the same radius.

\circlenode*[*par*]***{name}{stuff}***

This is a variant of **\pscirclebox** that gives the node the shape of the circle.

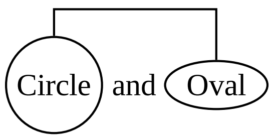
\cnodeput*[*par*]{*angle*}(*x*,*y*)***{name}{stuff}***

This is a variant of **\cput** that gives the node the shape of the circle. That is, it is like

\rput{*angle*}(*x*,*y*){\circlenode{*name*}{*stuff*}

\ovalnode*[*par*]***{name}{stuff}***

This is a variant of **\psovalbox** that gives the node the shape of an ellipse. Here is an example with **\circlenode** and **\ovalnode**:



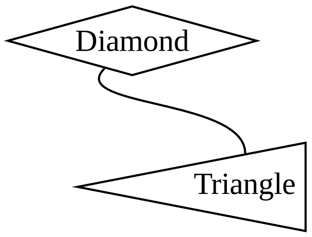
```
\circlenode{A}{Circle}  and  \ovalnode{B}{Oval}
\ncbar[angle=90]{A}{B}
```

\dianode*[*par*]***{name}{stuff}***

This is like **\diabox**.

\trinode*[*par*]**{*name*}{*stuff*}**

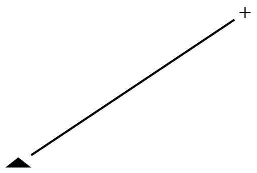
This is like **\tribox**.



```
\rput[tl](0,3){\dianode{A}{Diamond}}
\rput[br](4,0){\trinode[trimode=L]{B}{Triangle}}
\nccurve[angleA=-135,angleB=90]{A}{B}
```

\dotnode*[*par*](*x*,*y*)**{*name*}**

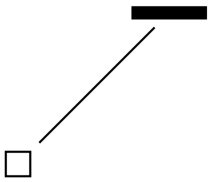
This is a variant of **\psdot**. For example:



```
\dotnode[dotstyle=triangle*,dotscale=2  1](0,0){A}
\dotnode[dotstyle=+](3,2){B}
\ncline[nodesep=3pt]{A}{B}
```

\fnode*[*par*](*x*,*y*)**{*name*}**

The f stands for "frame". This is like, but easier than, putting a **\psframe** in an **\rnode**.



```
\fnode{A}
\fnode*[framesize=1  5pt](2,2){B}
\ncline[nodesep=3pt]{A}{B}
```

There are two differences between **\fnode** and **\psframe**:

- There is a single (optional) coordinate argument, that gives the *center* of the frame.
- The width and height of the frame are set by the

  framesize=*dim1 'dim2'* **Default: 10pt**

  parameter. If you omit *dim2*, you get a square frame.

7 Node connections

All the node connection commands begin with nc, and they all have the same syntax:[1]

[1]The node connections can be used with **\pscustom**. The beginning of the node connection is attached to the current point by a straight line, as with **\psarc**.[2]

\nodeconnection[par]{arrows}{nodeA}{nodeB}

A line of some sort is drawn from *nodeA* to *nodeB*. Some of the node connection commands are a little confusing, but with a little experimentation you will figure them out, and you will be amazed at the things you can do. When we refer to the A and B nodes below, we are referring only to the order in which the names are given as arguments to the node connection macros.[3]

The node connections use many of the usual graphics parameters, plus a few special ones. Let's start with one that applies to all the node connections:

nodesep=*dim* **Default: 0pt**

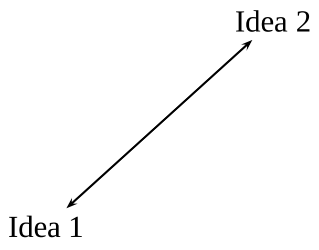
nodesep is the border around the nodes that is added for the purpose of determining where to connect the lines.

For this and other node connection parameters, you can set different values for the two ends of the node connection. Set the parameter **nodesepA** for the first node, and set **nodesepB** for the second node.

The first two node connections draw a line or arc directly between the two nodes:

\ncline*[*par*]{*arrows*}***{*nodeA*}{*nodeB*}**

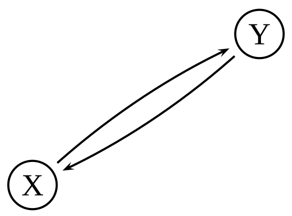
This draws a straight line between the nodes. For example:



```
\rput[bl](0,0){\rnode{A}{Idea  1}}
\rput[tr](4,3){\rnode{B}{Idea  2}}
\ncline[nodesep=3pt]{<->}{A}{B}
```

\ncarc*[*par*]{*arrows*}***{*nodeA*}{*nodeB*}**

This connects the two nodes with an arc.



```
\cnodeput(0,0){A}{X}
\cnodeput(3,2){B}{Y}
\psset{nodesep=3pt}
\ncarc{->}{A}{B}
\ncarc{->}{B}{A}
```

[3]When a node name cannot be found on the same page as the node connection command, you get either no node connection or a nonsense node connection. However, TEX will not report any errors.

The angle between the arc and the line between the two nodes is[4]

arcangle=*angle* **Default: 8**

\ncline and **\ncarc** both determine the angle at which the node connections join by the relative position of the two nodes. With the next group of node connections, you specify one or both of the angles in absolute terms, by setting the

angle=*angle* **Default: 0**

(and **angleA** and **angleB**) parameter.

You also specify the length of the line segment where the node connection joins at one or both of the ends (the "arms") by setting the

arm=*dim* **Default: 10pt**

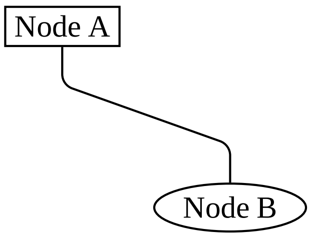
(and **armA** and **armB**) parameter.

These node connections all consist of several line segments, including the arms. The value of **linearc** is used for rounding the corners.

Here they are, starting with the simplest one:

\ncdiag*[*par*]{*arrows*}**{*nodeA*}{*nodeB*}**

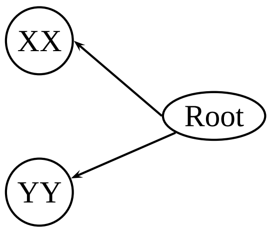
An arm is drawn at each node, joining at angle **angleA** or **angleB**, and with a length of **armA** or **armB**. Then the two arms are connected by a straight line, so that the whole line has three line segments. For example:



```
\rput[tl](0,3){\rnode{A}{\psframebox{Node  A}}}
\rput[br](4,0){\ovalnode{B}{Node  B}}
\ncdiag[angleA=-90,  angleB=90,  arm=.5,  linearc=.2]{A}{B}
```

[4]Rather than using a true arc, **\ncarc** actually draws a bezier curve. When connecting two circular nodes using the default parameter values, the curve will be indistinguishable from a true arc. However, **\ncarc** is more flexible than an arc, and works right connecting nodes of different shapes and sizes. You can set **arcangleA** and **arcangleB** separately, and you can control the curvature with the **ncurv** parameter, which is described on page **??**.

You can also set one or both of the arms to zero length. For example, if you set **arm=0**, the nodes are connected by a straight line, but you get to determine where the line connects (whereas the connection point is determined automatically by **\ncline**). Compare this use of **\ncdiag** with **\ncline** in the following example:

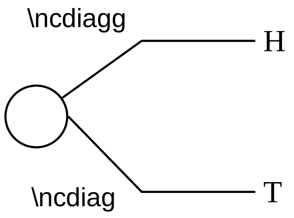


```
\rput[r](4,1){\ovalnode{R}{Root}}
\cnodeput(1,2){A}{XX}
\cnodeput(1,0){B}{YY}
\ncdiag[angleB=180, arm=0]{<-}{A}{R}
\ncline{<-}{B}{R}
```

(Note that in this example, the default value **angleA=0** is used.)

\ncdiagg*[*par*]{*arrows*}***{*nodeA*}{*nodeB*}**

\ncdiagg is similar to **\ncdiag**, but only the arm for node A is drawn. The end of this arm is then connected directly to node B. Compare **\ncdiagg** with **\ncdiag** when **armB=0**:

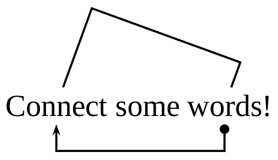


```
\cnode(0,0){12pt}{a}
\rput[l](3,1){\rnode{b}{H}}
\rput[l](3,-1){\rnode{c}{T}}
\ncdiagg[angleA=180, armA=1.5, nodesepA=3pt]{b}{a}
\ncdiag[angleA=180, armA=1.5, armB=0, nodesepA=3pt]{c}{a}
```

You can use **\ncdiagg** with **armA=0** if you want a straight line that joins to node A at the angle you specify, and to node B at an angle that is determined automatically.

\ncbar*[*par*]{*arrows*}***{*nodeA*}{*nodeB*}**

This node connection consists of a line with arms dropping "down", at right angles, to meet two nodes at an angle **angleA**. Each arm is at least of length **armA** or **armB**, but one may be need to be longer.

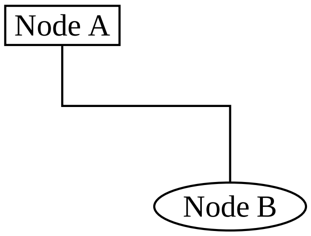


```
\rnode{A}{Connect} some \rnode{B}{words}!
\ncbar[nodesep=3pt,angle=-90]{<-**}{A}{B}
\ncbar[nodesep=3pt,angle=70]{A}{B}
```

Generally, the whole line has three straight segments.

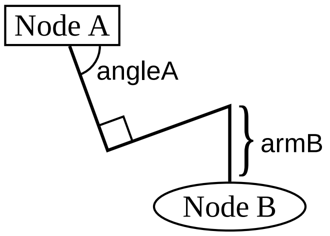
\ncangle*[*par*]{*arrows*}***{nodeA}{nodeB}***

Now we get to a more complicated node connection. **\ncangle** typically draws three line segments, like **\ncdiag**. However, rather than fixing the length of arm A, we adjust arm A so that the line joining the two arms meets arm A at a right angle. For example:



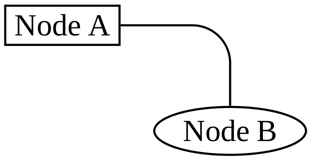
```
\rput[tl](0,3){\rnode{A}{\psframebox{Node  A}}}
\rput[br](4,0){\ovalnode{B}{Node  B}}
\ncangle[angleA=-90,angleB=90,armB=1cm]{A}{B}
```

Now watch what happens when we change **angleA**:



```
\rput[tl](0,3){\rnode{A}{\psframebox{Node  A}}}
\rput[br](4,0){\ovalnode{B}{Node  B}}
\ncangle[angleA=-70,angleB=90,armB=1cm,linewidth=1.2pt]{A}{B}
```

\ncangle is also a good way to join nodes by a right angle, with just two line segments, as in this example:

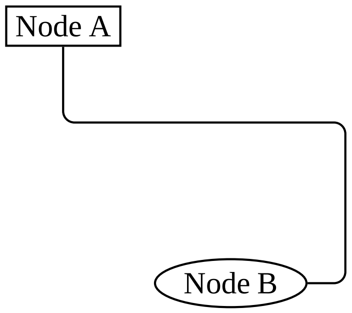


```
\rput[tl](0,2){\rnode{A}{\psframebox{Node  A}}}
\rput[br](4,0){\ovalnode{B}{Node  B}}
\ncangle[angleB=90,  armB=0,  linearc=.5]{A}{B}
```

\ncangles*[*par*]{*arrows*}***{nodeA}{nodeB}***

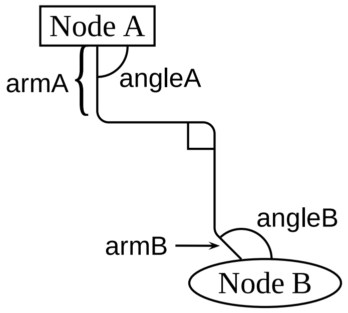
\ncangles is similar to **\ncangle**, but the length of arm A is fixed by the **armA** parameter. Arm A is connected to arm B by two line segments that meet arm A and each other at right angles. The angle at which they join arm B, and the length of the connecting segments, depends on the positions of the two arms. **\ncangles** generally draws a total of four line segments.[5] For example:

[5]Hence there is one more angle than **\ncangle**, and hence the s in **\ncangles**.



```
\rput[tl](0,4){\rnode{A}{\psframebox{Node  A}}}
\rput[br](4,0){\ovalnode{B}{Node  B}}
\ncangles[angleA=-90,  armA=1cm,  armB=.5cm,  linearc=.15]{A}{B}
```

Let’s see what happens to the previous example when we change **angleB**:



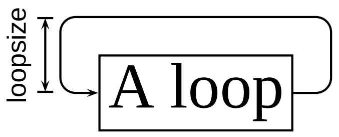
```
\rput[tl](0,4){\rnode{A}{\psframebox{Node  A}}}
\rput[br](4,0){\ovalnode{B}{Node  B}}
\ncangles[angleA=-90,  angleB=135,  armA=1cm,  armB=.5cm,
    linearc=.15]{A}{B}
```

\ncloop*[*par*]{*arrows*}***{nodeA}{nodeB}***

\ncloop is also in the same family as **\ncangle** and **\ncangles**, but now typically 5 line segments are drawn. Hence, **\ncloop** can reach around to opposite sides of the nodes. The lengths of the arms are fixed by **armA** and **armB**. Starting at arm A, **\ncloop** makes a 90 degree turn to the left, drawing a segment of length

loopsize=*dim* **Default: 1cm**

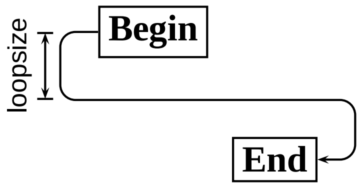
This segment connects to arm B the way arm A connects to arm B with **\ncline**; that is, two more segments are drawn, which join the first segment and each other at right angles, and then join arm B. For example:



```
\rnode{a}{\psframebox{\Huge  A  loop}}
\ncloop[angleB=180,loopsize=1,arm=.5,linearc=.2]{->}{a}{a}
```

In this example, node A and node B are the same node! You can do this with all the node connections (but it doesn’t always make sense).

Here is an example where **\ncloop** connects two different nodes:

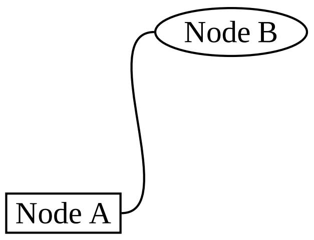


```
\parbox{3cm}{%
\rnode{A}{\psframebox{\large\bf Begin}}
\vspace{1cm}\hspace*{\fill}
\rnode{B}{\psframebox{\large\bf End}}
\ncloop[angleA=180,loopsize=.9,arm=.5,linearc=.2]{->}{A}{B}}
```

The next two node connections are a little different from the rest.

\nccurve*[*par*]{*arrows*}***{nodeA}{nodeB}***

\nccurve draws a bezier curve between the nodes.



```
\rput[bl](0,0){\rnode{A}{\psframebox{Node A}}}
\rput[tr](4,3){\ovalnode{B}{Node B}}
\nccurve[angleB=180]{A}{B}
```

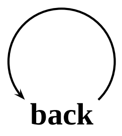
You specify the angle at which the curve joins the nodes by setting the **angle** (and **angleA** and **angleB**) parameter. The distance to the control points is set with the

ncurv=*num* **Default: .67**

(and **ncurvA** and **ncurvB**) parameter. A lower number gives a tighter curve. (The distance between the beginning of the arc and the first control point is one-half **ncurvA** times the distance between the two endpoints.)

\nccircle*[*par*]{*arrows*}***{node}{radius}***

\nccircle draws a circle, or part of a circle, that, if complete, would pass through the center of the node counterclockwise, at an angle of **angleA**.



```
\rnode{A}{\bf back}
\nccircle[nodesep=3pt]{->}{A}{.7cm}
\kern 5pt
```

\nccircle can only connect a node to itself; it is the only node connection with this property. **\nccircle** is also special because it has an additional argument, for specifying the radius of the circle.

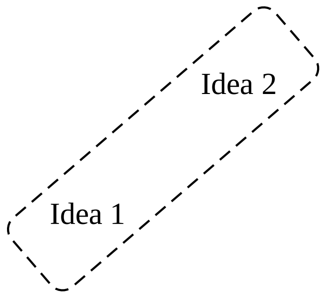
The last two node connections are also special. Rather than connecting the nodes with an open curve, they enclose the nodes in a box or curved box. You can think of them as variants of **\ncline** and **\ncarc**. In both cases, the half the width of the box is

boxsize=*dim* **Default: .4cm**

You have to set this yourself to the right size, so that the nodes fit inside the box. The **boxsize** parameter actually sets the **boxheight** and **boxdepth** parameters. The ends of the boxes extend beyond the nodes by **nodesepA** and **nodesepB**.

\ncbox*[*par*]**{*nodeA*}{*nodeB*}**

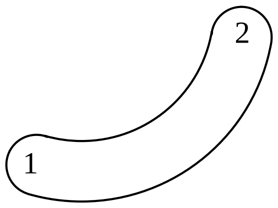
\ncbox encloses the nodes in a box with straight sides. For example:



```
\rput[bl](.5,0){\rnode{A}{Idea 1}}
\rput[tr](3.5,2){\rnode{B}{Idea 2}}
\ncbox[nodesep=.5cm,boxsize=.6,linearc=.2,
   linestyle=dashed]{A}{B}
```

\ncarcbox*[*par*]**{*nodeA*}{*nodeB*}**

\ncarcbox encloses the nodes in a curved box that is **arcangleA** away from the line connecting the two nodes.



```
\rput[bl](.5,0){\rnode{A}{1}}
\rput[tr](3.5,2){\rnode{B}{2}}
\ncarcbox[nodesep=.2cm,boxsize=.4,linearc=.4,
   arcangle=50]{<->}{A}{B}
```

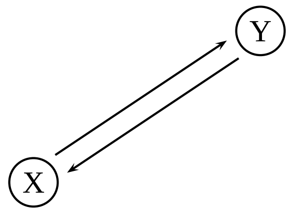
The arc is drawn counterclockwise from node A to node B.

There is one other node connection parameter that applies to all the node connections, except **\ncarcbox**:

offset=*dim* **Default: 0pt**

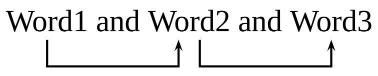
(You can also set **offsetA** and **offsetB** independently.) This shifts the point where the connection joins up by *dim* (given the convention that connections go from left to right).

There are two main uses for this parameter. First, it lets you make two parallel lines with **\ncline**, as in the following example:



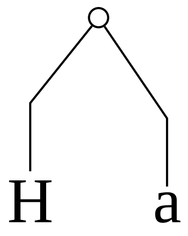
```
\cnodeput(0,0){A}{X}
\cnodeput(3,2){B}{Y}
\psset{nodesep=3pt,offset=4pt,arrows=->}
\ncline{A}{B}
\ncline{B}{A}
```

Second, it lets you join a node connection to a rectangular node at a right angle, without limiting yourself to positions that lie directly above, below, or to either side of the center of the node. This is useful, for example, if you are making several connections to the same node, as in the following example:



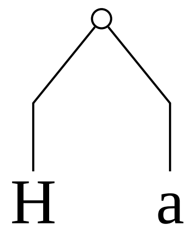
```
\rnode{A}{Word1} and \rnode{B}{Word2} and \rnode{C}{Word3}
\ncbar[offsetB=4pt,angleA=-90,nodesep=3pt]{->}{A}{B}
\ncbar[offsetA=4pt,angleA=-90,nodesep=3pt]{->}{B}{C}
```

Sometimes you might be aligning several nodes, such as in a tree, and you want to ends or the arms of the node connections to line up. This won't happen naturally if the nodes are of different size, as you can see in this example:



```
\Huge
\cnode(1,3){4pt}{a}
\rput[B](0,0){\Rnode{b}{H}}
\rput[B](2,0){\Rnode{c}{a}}
\psset{angleA=90,armA=1,nodesepA=3pt}
\ncdiagg{b}{a}
\ncdiagg{c}{a}
```

If you set the **nodesep** or **arm** parameter to a negative value, PSTricks will measure the distance to the beginning of the node connection or to the end of the arm relative to the center of the node, rather than relative to the boundary of the node or the beginning of the arm. Here is how we fix the previous example:



```
\Huge
\cnode(1,3){4pt}{a}
\rput[B](0,0){\Rnode{b}{H}}
\rput[B](2,0){\Rnode{c}{a}}
\psset{angleA=90,armA=1,nodesepA=-12pt}
\ncdiagg{b}{a}
\ncdiagg{c}{a}
```

Note also the use of **\Rnode**.

One more paramter trick: By using the **border** parameter, you can create the impression that one node connection passes over another.

The node connection commands make interesting drawing tools as well, as an alternative to **\psline** for connecting two points. There are variants of the node connection commands for this purpose. Each begins with pc (for "point connection") rather than nc. E.g.,

```
\pcarc{<->}(3,4)(6,9)
```

gives the same result as

```
\pnode(3,4){A}
\pnode(6,9){B}
\pcarc{<->}{A}{B}
```

Only **\nccircle** does not have a pc variant:

Command	*Corresponds to:*
\pcline{*arrows*}**(*x1*,*y1*)(*x2*,*y2*)**	**\ncline**
\pccurve{*arrows*}**(*x1*,*y1*)(*x2*,*y2*)**	**\nccurve**
\pcarc{*arrows*}**(*x1*,*y1*)(*x2*,*y2*)**	**\ncarc**
\pcbar{*arrows*}**(*x1*,*y1*)(*x2*,*y2*)**	**\ncbar**
\pcdiag{*arrows*}**(*x1*,*y1*)(*x2*,*y2*)**	**\ncdiag**
\pcangle{*arrows*}**(*x1*,*y1*)(*x2*,*y2*)**	**\ncangle**
\pcloop{*arrows*}**(*x1*,*y1*)(*x2*,*y2*)**	**\ncloop**
\pcbox(*x1*,*y1*)(*x2*,*y2*)	**\ncbox**
\pcarcbox(*x1*,*y1*)(*x2*,*y2*)	**\ncarcbox**

8 Node connections labels: I

Now we come to the commands for attaching labels to the node connections. The label command must come right after the node connection to which the label is to be attached. You can attach more than one label to a node connection, and a label can include more nodes.

The node label commands must end up on the same TEX page as the node connection to which the label corresponds.

There are two groups of connection labels, which differ in how they select the point on the node connection. In this section we describe the first group:

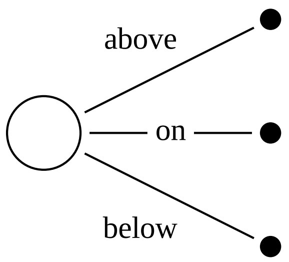
\ncput*[*par*]**{*stuff*}**
\naput*[*par*]**{*stuff*}**
\nbput*[*par*]**{*stuff*}**

These three command differ in where the labels end up with respect to the line:

\ncput	*on* the line
\naput	*above* the line
\nbput	*below* the line

(using the convention that node connections go from left to right).

Here is an example:



```
\cnode(0,0){.5cm}{root}
\cnode*(3,1.5){4pt}{A}
\cnode*(3,0){4pt}{B}
\cnode*(3,-1.5){4pt}{C}
\psset{nodesep=3pt}
\ncline{root}{A}
\naput{above}
\ncline{root}{B}
\ncput*{on}
\ncline{root}{C}
\nbput{below}
```

\naput and **\nbput** use the same algorithm as **\uput** for displacing the labels, and the distance beteen the line and labels is **labelsep** (at least if the lines are straight).

\ncput uses the same system as **\rput** for setting the reference point. You change the reference point by setting the

ref=*ref* **Default: c**

parameter.

Rotation is also controlled by a graphics parameter:

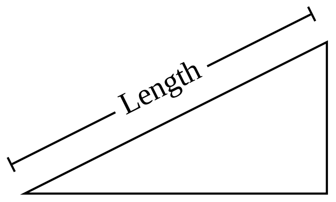
nrot=*rot* **Default: 0**

rot can be in any of the forms suitable for **\rput**, and you can also use the form

```
{:angle}
```

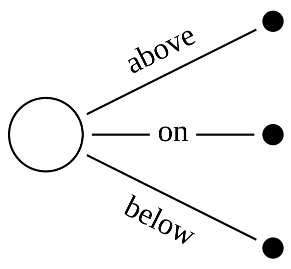
The angle is then measured with respect to the node connection. E.g., if the angle is {:U}, then the label runs parallel to the node connection. Since the label can include other put commands, you really have a lot of control over the label position.

The next example illustrates the use {:*angle*}, the **offset** parameter, and **\pcline**:



```
\pspolygon(0,0)(4,2)(4,0)
\pcline[offset=12pt]{|-|}(0,0)(4,2)
\ncput*[nrot=:U]{Length}
```

Here is a repeat of an earlier example, now using {:*angle*}:



```
\cnode(0,0){.5cm}{root}
\cnode*(3,1.5){4pt}{A}
\cnode*(3,0){4pt}{B}
\cnode*(3,-1.5){4pt}{C}
\psset{nodesep=3pt,nrot=:U}
\ncline{root}{A}
\naput{above}
\ncline{root}{B}
\ncput*{on}
\ncline{root}{C}
\nbput{below}
```

The position on the node connection is set by the

npos=*num* **Default:**

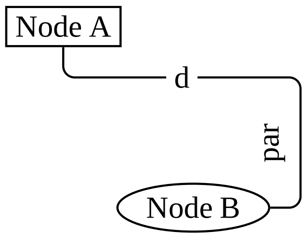
parameter, roughly according to the following scheme: Each node connection has potentially one or more segments, including the arms and connecting lines. A number **npos** between 0 and 1 picks a point on the first segment from node A to B (fraction **npos** from the beginning to the end of the segment), a number between 1 and 2 picks a number on the second segment, and so on.

Each node connection has its own default value of **npos**. If you leave the **npos** parameter value empty (e.g., [npos=]), then the default is substituted. This is the default mode.

Here are the details for each node connection:

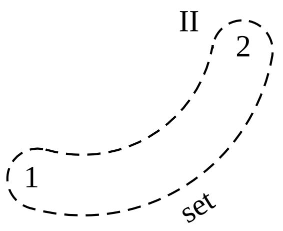
Connection	*Segments*	*Range*	*Default*
\ncline	1	$0 \leq pos \leq 1$	0.5
\nccurve	1	$0 \leq pos \leq 1$	0.5
\ncarc	1	$0 \leq pos \leq 1$	0.5
\ncbar	3	$0 \leq pos \leq 3$	1.5
\ncdiag	3	$0 \leq pos \leq 3$	1.5
\ncdiagg	2	$0 \leq pos \leq 2$	0.5
\ncangle	3	$0 \leq pos \leq 3$	1.5
\ncangles	4	$0 \leq pos \leq 4$	1.5
\ncloop	5	$0 \leq pos \leq 5$	2.5
\nccircle	1	$0 \leq pos \leq 1$	0.5
\ncbox	4	$0 \leq pos \leq 4$	0.5
\ncarcbox	4	$0 \leq pos \leq 4$	0.5

Here is an example:



```
\rput[tl](0,3){\rnode{A}{\psframebox{Node  A}}}
\rput[br](3.5,0){\ovalnode{B}{Node  B}}
\ncangles[angleA=-90,arm=.4cm,linearc=.15]{A}{B}
\ncput*{d}
\nbput[nrot=:D,npos=2.5]{par}
```

With **\ncbox** and **\ncarcbox**, the segments run counterclockwise, starting with the lower side of the box. Hence, with **\nbput** the label ends up outside the box, and with **\naput** the label ends up inside the box.

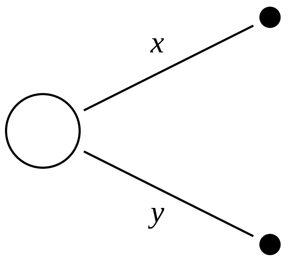


```
\rput[bl](.5,0){\rnode{A}{1}}
\rput[tr](3.5,2){\rnode{B}{2}}
\ncarcbox[nodesep=.2cm,boxsize=.4,linearc=.4,
   arcangle=50,linestyle=dashed]{<->}{A}{B}
\nbput[nrot=:U]{set}
\nbput[npos=2]{II}
```

If you set the parameter

shortput=*none/nab/tablr/tab* **Default: none**

to nab, then immediately following a node connection or another node connection label you can use ˆ instead of **\naput** and _ instead of **\nbput**.



```
\cnode(0,0){.5cm}{root}
\cnode*(3,1.5){4pt}{A}
\cnode*(3,-1.5){4pt}{C}
\psset{nodesep=3pt,shortput=nab}
\ncline{root}{A}^{$x$}
\ncline{root}{C}_{$y$}
```

You can still have parameter changes with the short ˆ and _ forms. Another example is given on page 27.

If you have set **shortput=nab**, and then you want to use a true ˆ or _ character right after a node connection, you must precede the ˆ or _ by {} so that PSTricks does not convert it to \naput or \nbput.

You can change the characters that you use for the short form with the

\MakeShortNab{*char1*}{*char2*}

command.[6]

The **shortput=tablr** and **shortput=tab** options are described on pages 24 and **??**, respectively.

9 Node connection labels: II

Now the second group of node connections:

[6]You can also use **\MakeShortNab** if you want to use ˆ and _ with non-standard category codes. Just invoke the command after you have made your \catcode changes.

\tvput*[*par*]**{*stuff*}**
\tlput*[*par*]**{*stuff*}**
\trput*[*par*]**{*stuff*}**
\thput*[*par*]**{*stuff*}**
\taput*[*par*]**{*stuff*}**
\tbput*[*par*]**{*stuff*}**

The difference between between these commands and the \n*put commands is that these find the position as an intermediate point between the centers of the nodes, either in the horizontal or vertical direction. These are good for trees and mathematical diagrams, where it can sometimes be nice to have the labels be horizontally or vertically aligned. The t stands for "tree".

You specify the position by setting the

tpos=*num* **Default: .5**

parameter.

\tvput, **\tlput** and **\trput** find the position that lies fraction *tpos* in the *vertical* direction from the upper node to the lower node. **\thput**, **\taput** and **\tbput** find the position that lies fraction *tpos* in the *horizontal* direction from the left node to the right node. Then the commands put the label on or next to the line, as follows:

Command	*Direction*	*Placement*
\tvput	vertical	middle
\tlput	vertical	left
\trput	vertical	right
\thput	horizontal	middle
\taput	horizontal	above
\tbput	horizontal	below

Here is an example:

```
\[
  \setlength{\arraycolsep}{1.1cm}
  \begin{array}{cc}
    \Rnode{a}{(X-A)}  &  \Rnode{b}{A}  \\[1.5cm]
    \Rnode{c}{x}  &  \Rnode{d}{\tilde{X}}
  \end{array}
```

```
    \psset{nodesep=5pt,arrows=->}
    \everypsbox{\scriptstyle}
    \ncline{a}{c}\tlput{r}
    \ncline{a}{b}\taput{u}
    \ncline[linestyle=dashed]{c}{d}\tbput{b}
    \ncline{b}{d}\trput{s}
\]
```

$$\begin{tikzcd}
(X-A) \arrow[r, "u"] \arrow[d, "r"'] & A \arrow[d, "s"] \\
x \arrow[r, dashed, "b"'] & \tilde{X}
\end{tikzcd}
\qquad
\begin{tikzcd}
(X-A) \arrow[r, "u"] \arrow[d, "r"'] & a \arrow[d, "s"] \\
x \arrow[r, dashed, "b"'] & \tilde{X}
\end{tikzcd}$$

On the left is the diagram with **\tlput**, **\trput** **\tbput** and **\Rnode**, as shown in the code. On the right is the same diagram, but with **\naput** **\nbput** and **\rnode**.

These do not have a rotation argument or parameter. However, you can rotate *stuff* in 90 degree increments using box rotations (e.g., **\rotateleft**).

If you set **shortput=tablr**, then you can use the following single-character abbreviations for the t put commands:

Char.	*Short for:*
ˆ	**\taput**
_	**\tbput**
<	**\tlput**
>	**\trput**

You can change the character abbreviations with

\MakeShortTablr{*char1*}{*char2*}{*char3*}{*char4*}

The t put commands, including an example of **shortput=tablr**, will be shown further when we get to mathematical diagrams and trees.

Driver notes: The node macros use **\pstVerb** and **\pstverbscale**.

10 Attaching labels to nodes

The command

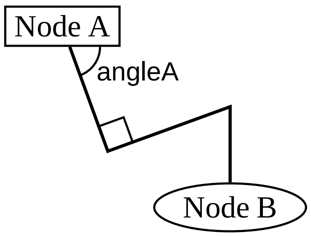
\nput*[*par*]**{*refangle*}{*name*}{*stuff*}**

affixes *stuff* to node *name*. It is positioned distance **labelsep** from the node, in the direction *refangle* from the center of the node. The algorithm is the same as for **\uput**. If you want to rotate the node, set the

rot=*rot* **Default: 0**

parameter, where *rot* is a rotation that would be valid for **\rput**.[7] The position of the label also takes into account the **offsetA** parameter. If **labelsep** is negative, then the distance is from the center of the node rather than from the boundary, as with **nodesep**.

Here is how I used **\nput** to mark an angle in a previous example:



```
\rput[br](4,0){\ovalnode{B}{Node  B}}
\rput[tl](0,3){\rnode{A}{\psframebox{Node  A}}}
\nput[labelsep=0]{-70}{A}{%
   \psarcn(0,0){.4cm}{0}{-70}
   \uput{.4cm}[-35](0,0){{\tt  angleA}}}
\ncangle[angleA=-70,angleB=90,armB=1cm,linewidth=1.2pt]{A}{B}
\ncput[nrot=:U,npos=1]{\psframe[dimen=middle](0,0)(.35,.35)}
```

11 Mathematical diagrams and graphs

For some applications, such as mathematical diagrams and graphs, it is useful to arrange nodes on a grid. You can do this with alignment environments, such as TEX's \halignprimitive, LATEX's tabular environment, and AMS-TEX's \matrix, but PSTricks contains its own alignment environment that is especially adapted for this purpose:

\psmatrix ... \endpsmatrix

Here is an example

$$\begin{array}{ccc} & A & \\ & & \\ B & E & C \\ & & \\ & D & \end{array}$$

```
$
\psmatrix[colsep=1cm,rowsep=1cm]
     & A \\
  B & E & C \\
     & D &
\endpsmatrix
$
```

[7]Not to be confused with the nput parameter.

As an alignment environment, **\psmatrix** is similar to AMS-TEX's \matrix. There is no argument for specifying the columns. Instead, you can just use as many columns as you need. The entries are horizontally centered. Rows are ended by \\. **\psmatrix** can be used in or out of math mode.

Our first example wasn't very interesting, because we didn't make use of the nodes. Actually, each entry is a node. The name of the node in row *row* and column *col* is {*row*,*col*}, with no spaces. Let's see some node connections:

$$\begin{tikzcd}
 & X \arrow[dl, "f"'] \arrow[d, "g"] \\
Y \arrow[r, dotted, "h"'] & Z
\end{tikzcd}$$

```
$
\psmatrix[colsep=1cm]
     & X \\
  Y & Z
\endpsmatrix
\everypsbox{\scriptstyle}%
\psset{nodesep=3pt,arrows=->}
\ncline{1,2}{2,1}
\tlput{f}
\ncline{1,2}{2,2}
\trput{g}
\ncline[linestyle=dotted]{2,1}{2,2}
\tbput{h}
$
```

You can include the node connections inside the **\psmatrix**, in the last entry and right before **\endpsmatrix**. One advantage to doing this is that **shortput=tab** is the default within a **\psmatrix**.

$$\begin{tikzcd}
U \arrow[dr, "y"'] \arrow[drr, Rightarrow, dashed, "x"] & & \\
 & X \times_Z Y \arrow[r, "p"'] \arrow[d, "q"'] & X \arrow[d, "f"] \\
 & Y \arrow[r, "g"'] & Z
\end{tikzcd}$$

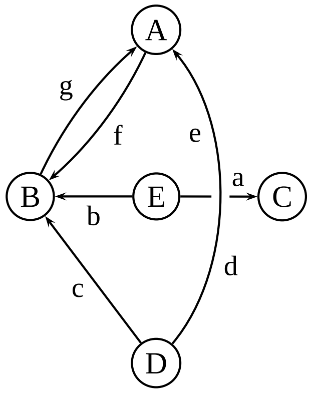
```
$
  \begin{psmatrix}
     U \\
       &  X\times_Z  Y  &  X  \\
       &        Y          &  Z
     \psset{arrows=->,nodesep=3pt}
     \everypsbox{\scriptstyle}
     \ncline{1,1}{2,2}_{y}
     \ncline[doubleline=true,linestyle=dashed]{-}{1,1}{2,3}^{x}
     \ncline{2,2}{3,2}<{q}
     \ncline{2,2}{2,3}_{p}
     \ncline{2,3}{3,3}>{f}
     \ncline{3,2}{3,3}_{g}
  \end{psmatrix}
$
```

You can change the kind of nodes that are made by setting the

mnode=*type* **Default: R**

parameter. Valid types are R, r, C, f, p, circle, oval, dia, tri, dot and none, standing for **\Rnode**, **\rnode**, **\Cnode**, **\fnode**, **\pnode**, **\circlenode**, **\ovalnode**, **\dotnode** and no node, respectively. Note that for circles, you use **mnode=C** and set the radius with the **radius** parameter.

For example:



```
\psmatrix[mnode=circle,colsep=1]
     & A \\
  B & E & C \\
     & D &
\endpsmatrix
\psset{shortput=nab,arrows=->,labelsep=3pt}
\small
\ncline{2,2}{2,3}^[npos=.75]{a}
\ncline{2,2}{2,1}^{b}
\ncline{3,2}{2,1}^{c}
\ncarc[arcangle=-40,border=3pt]{3,2}{1,2}
  _[npos=.3]{d}^[npos=.7]{e}
\ncarc[arcangle=12]{1,2}{2,1}^{f}
\ncarc[arcangle=12]{2,1}{1,2}^{g}
```

Note that a node is made only for the non-empty entries. You can also specify a node for the empty entries by setting the

emnode=*type* **Default: none**

parameter.

You can change parameters for a single entry by starting with entry with the parameter changes, enclosed in square brackets. Note that the changes affect the way the node is made, but not contents of the entry (use **\psset** for this purpose). For example:

$$\begin{tikzcd}
 & X \arrow[dl] \arrow[d] \\
Y \arrow[r, dotted] & Z
\end{tikzcd}$$

```
$
\psmatrix[colsep=1cm]
     &  [mnode=circle]  X  \\
   Y  &  Z
\endpsmatrix
\psset{nodesep=3pt,arrows=->}
\ncline{1,2}{2,1}
\ncline{1,2}{2,2}
\ncline[linestyle=dotted]{2,1}{2,2}
$
```

If you want your entry to begin with a [that is not meant to indicate parameter changes, the precede it by {}.

You can assign your own name to a node by setting the

name=*name* **Default:**

parameter at the beginning of the entry, as described above. You can still refer to the node by {*row*,*col*}, but here are a few reasons for giving your own name to a node:

- The name may be easier to keep track of;
- Unlike the {*row*,*col*} names, the names you give remain valid even when you add extra rows or columns to your matrix.
- The names remain valid even when you start a new **\psmatrix** that reuses the {*row*,*col*} names.

Here a few more things you should know:

- The baselines of the nodes pass through the centers of the nodes. **\psmatrix** achieves this by setting the

  nodealign=*true/false* **Default: false**

  parameter to true. You can also set this parameter outside of **\psmatrix** when you want this kind of alignment.

- You can left or right-justify the nodes by setting the

  mcol=*l/r/c* **Default: c**

  parameter. l, r and c stand for left, right and center, respectively.

- The space between rows and columns is set by the

rowsep=*dim* **Default: 1.5cm**
colsep=*dim* **Default: 1.5cm**

parameters.

- If you want all the nodes to have a fixed with, set

  mnodesize=*dim* **Default: -1pt**

  to a positive value.

- If **\psmatrix** is used in math mode, all the entries are set in math mode, but you can switch a single entry out of math mode by starting and ending the entry with $.

- The radius of the c **mnode** (corresponding to **\cnode**) is set by the

  radius=*dim* **Default: 2pt**

  parameter.

- Like in LaTeX, you can end a row with \\\\[*dim*] to insert an extra space *dim* between rows.

- The command \psrowhookii is executed, if defined, at the beginning of every entry in row ii (row 2), and the command \pscolhookv is executed at athe beginning of every entry in column v (etc.). You can use these hooks, for example, to change the spacing between two columns, or to use a special **mnode** for all the entries in a particular row.

- An entry can itself be a node. You might do this if you want an entry to have two shapes.

- If you want an entry to stretch across several (*int*) columns, use the

  \psspan{*int*}

  at the end of the entry. This is like Plain TeX's **\multispan**, or LaTeX's **\multicolumn**, but the template for the current column (the first column that is spanned) is still used. If you want wipe out the template as well, use \multispan{*int*} *at the beginning of the entry* instead. If you just want to wipe out the template, use \omit before the entry.

- **\psmatrix** can be nested, but then all node connections and other references to the nodes in the {*row*,*col*} form for the nested matrix *must go inside* the **\psmatrix**. This is how PSTricks decides which matrix you are referring to. It is still neatest to put all the node connections towards the end; just be sure to put them before **\endpsmatrix**. Be careful also not to refer to a node until it actually appears. The whole matrix can itself go inside a node, and node connections can be made as usual. This is not the same as connecting nodes from two different **\psmatrix**'s. To do this, you must give the nodes names and refer to them by these names.

12 Obsolete put commands

This is old documentation, but these commands will continue to be supported.

There is also an obsolete command **\Lput** for putting labels next to node connections. The syntax is

```
\Lput{labelsep}[refpoint]{rotation}(pos){stuff}
```

It is a combination of **\Rput** and **\lput**, equivalent to

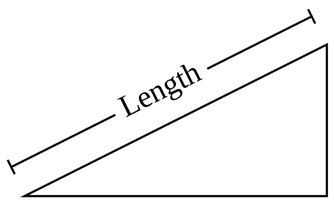
```
\lput(pos){\Rput{labelsep}[refpoint]{rotation}(0,0){stuff}}
```

\Mput is a short version of **\Lput** with no {*rotation*} or (*pos*) argument. **\Lput** and **\Mput** remain part of PSTricks only for backwards compatibility.

Here are the node label commands:

\lput*[*refpoint*]{*rotation*}***(pos){stuff}***

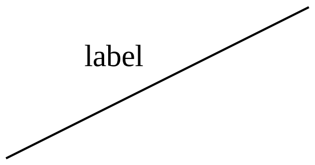
The l stands for "label". Here is an example illustrating the use of the optional star and :*angle* with **\lput**, as well as the use of the **offset** parameter with **\pcline**:



```
\pspolygon(0,0)(4,2)(4,0)
\pcline[offset=12pt]{|-|}(0,0)(4,2)
\lput*{:U}{Length}
```

(Remember that with the put commands, you can omit the coordinate if you include the angle of rotation. You are likely to use this feature with the node label commands.)

With **\lput** and **\rput**, you have a lot of control over the position of the label. E.g.,

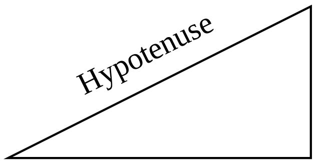


```
\pcline(0,0)(4,2)
\lput{:U}{\rput[r]{N}(0,.4){label}}
```

puts the label upright on the page, with right side located .4 centimeters "above" the position .5 of the node connection (above if the node connection points to the right). However, the **\aput** and **\bput** commands described below handle the most common cases without **\rput**.[8]

\aput*[*labelsep*]{*angle*}***(*pos*){*stuff*}**

stuff is positioned distance **\pslabelsep** *above* the node connection, given the convention that node connections point to the right. \aput is a node-connection variant of **\uput**. For example:



```
\pspolygon(0,0)(4,2)(4,0)
\pcline[linestyle=none](0,0)(4,2)
\aput{:U}{Hypotenuse}
```

\bput*[*labelsep*]{*angle*}***(*pos*){*stuff*}**

This is like **\aput**, but *stuff* is positioned *below* the node connection.

It is fairly common to want to use the default position and rotation with these node connections, but you have to include at least one of these arguments. Therefore, PSTricks contains some variants:

[8]There is also an obsolete command **\Lput** for putting labels next to node connections. The syntax is

\Lput{*labelsep*}[*refpoint*]{*rotation*}(*pos*){*stuff*}

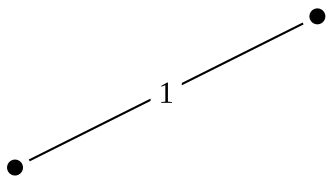
It is a combination of **\Rput** and **\lput**, equivalent to

\lput(*pos*){\Rput{*labelsep*}[*refpoint*]{*rotation*}(0,0){*stuff*}}

\Mput is a short version of **\Lput** with no {*rotation*} or (*pos*) argument. **\Lput** and **\Mput** remain part of PSTricks only for backwards compatibility.

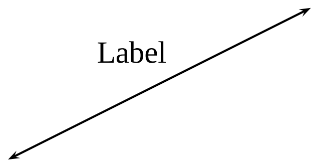
\mput*[*refpoint*]**{*stuff*}**
\Aput*[*labelsep*]**{*stuff*}**
\Bput*[*labelsep*]**{*stuff*}**

of **\lput**, **\aput** and **\bput**, respectively, that have no angle or positioning argument. For example:



```
\cnode*(0,0){3pt}{A}
\cnode*(4,2){3pt}{B}
\ncline[nodesep=3pt]{A}{B}
\mput*{1}
```

Here is another:



```
\pcline{<->}(0,0)(4,2)
\Aput{Label}
```

II Trees

13 Overview

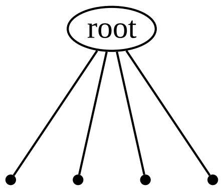
pstree

The node and node connections are perfect tools for making trees. The file pstree.tex / pstree.sty contains a high-level interface for positioning the nodes in a tree.

The main tree macro is

\pstree{*(root)node*}{*(sub)trees and (terminal)nodes*}

This positions the root node above its successors.

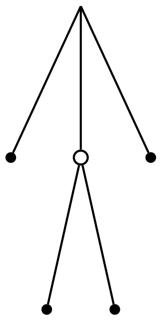


```
\pstree{\Toval{root}}{\TC*  \TC*  \TC*  \TC*}
```

\pstree produces a box that encloses all the nodes, and whose baseline passes through the center of the root node.

For most of the nodes described in Section 6 (e.g., **\ovalnode**), there is a variant for use within a tree (e.g., **\Toval**). Note that there is no distinction between a terminal node and a root node, other than their position in the **\pstree** command.

A tree, when included in the list or successors, becomes a subtree.



```
\pstree{\Tp}{%
   \TC*
   \pstree{\Tc{3pt}}{\TC*  \TC*}
   \TC*}
```

14 Tree Nodes

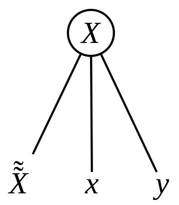
For most nodes described in Section 6, you can add strip `node` from the end of the name and add `T` add the beginning to obtain a node for use in trees. The syntax of a tree node is the same as of its corresponding “normal” node, except that:

- 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, and
- there is never a coordinate argument for positioning the node.
- to set the reference point with **\Tr**, set the **ref** parameter.

Here is the list of such tree nodes:

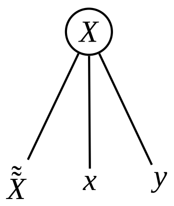
\Tp*[*par*]
\Tc*[*par*]**{*dim*}**
\TC*[*par*]
\Tf*[*par*]
\Tdot*[*par*]
\Tr*[*par*]**{*stuff*}**
\TR*[*par*]**{*stuff*}**
\Tcircle*[*par*]**{*stuff*}**
\Toval*[*par*]**{*stuff*}**
\Tdia*[*par*]**{*stuff*}**
\Ttri*[*par*]**{*stuff*}**

\Rnode is a good choice when you want the baselines of the text in the nodes to line up horizontally.



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

Compare the preceding example with the next one, which uses **\rnode**:

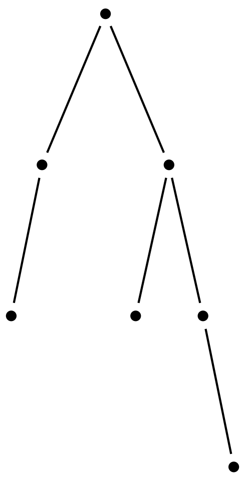


```
$
\pstree[nodesepB=3pt]{\Tcircle{X}}{%
   \Tr{\tilde{\tilde{X}}}
   \Tr{x}
   \Tr{y}}
$
```

There is also a null tree node

\Tn

It is meant to be just a place holder.



```
\pstree[nodesep=3pt]{\TC*}{%
   \pstree{\TC*}{\TC*  \Tn}
   \pstree{\TC*}{%
      \TC*
      \pstree{\TC*}{\Tn\TC*}}}
```

Actually, if I was going to do this a lot I would define some short-cuts:

```
\def\mytree{\pstree{\TC*}}
\def\ltree#1{\mytree{#1\Tn}}
\def\rtree#1{\mytree{\Tn#1}}
\psset{nodesep=3pt}
\mytree{%
   \ltree{\TC*}
   \mytree{%
      \TC*
      \rtree{\TC*}}}
```

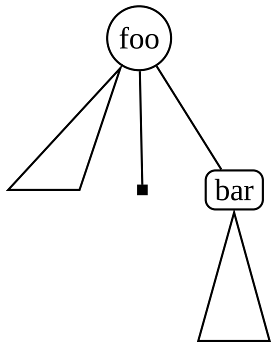
There is also a special tree node that doesn’t have a “normal” version and that can’t be used as the root node of a whole tree:

\Tfan*[*par*]

This draws a triangle whose base is

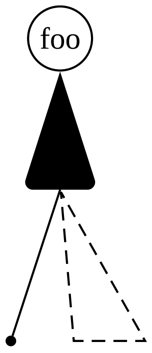
fansize=*dim* **Default: 1cm**

and whose opposite corner is the predecessor node, adjusted by the value of **nodesepA** and **offsetA**. For example:



```
\pstree{\Tcircle{foo}}{%
  \Tfan
  \Tf*[framesize=4pt]
  \pstree{\Tr{\psframebox[framearc=.5]{bar}}}{\Tfan}}
```

Here is another example illustrating that a **\Tfan** can have successors:



```
\pstree{\Tcircle{foo}}{%
  \pstree{\Tfan*[linearc=.1]}{%
    \Tc*{2pt}
    \Tfan[linestyle=dashed]}}
```

15 Trees

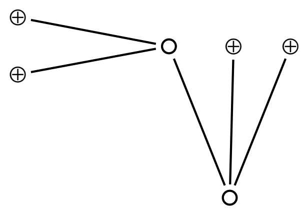
This section describes several graphics parameters for **\pstree**. Any settings of graphics parameters for **\pstree** affects all of its successors, including subtrees. but not the root node.

The

treemode=*R/L/U/D* **Default:**

parameter controls the direction in which the tree grows. R, L, U and D stand for “right”, “left”, “up” and “down”, respectively. When you change the **treemode**, the **treemode** of all nested trees changes as well.

For example, here is a tree that grows up, and then to the left:



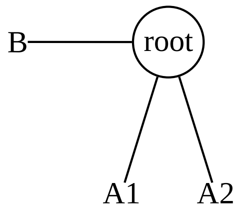
```
\pstree[treemode=U,dotstyle=oplus,dotsize=6pt,
   nodesep=2pt]{\Tc{3pt}}{%
   \pstree[treemode=L]{\Tc{3pt}}{%
      \Tdot
      \Tdot}
   \Tdot
   \Tdot}
```

When the tree goes up or down, the successors are lined up from left to right in the order they appear in **\pstree**'s argument. When the tree goes 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

treeflip=*true/false* **Default: false**

let's you do this.

A tree can also be root node. This is useful when the nested tree goes off in a different direction. If *treeB* is the root node of *treeA*, then the root of *treeB* is also the root node *treeA*.



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

A node can also contain a tree, but that is another story.

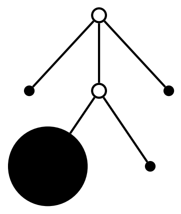
The distance between successors and between levels is given by the

treesep=*dim* **Default: .75cm**
levelsep=dim*** **Default: 2cm**

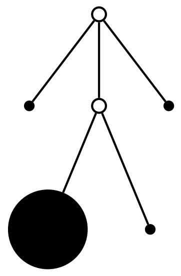
parameters.

The distance between successors takes into account the size of the nodes, but the distance between levels does not, at least by default. If you include the optional * when setting **levelsep**, the level sep is in addition to the size of the nodes. However, PSTricks needs a second run through TEX (without any changes between runs) to get the spacing right, and it writes to the .aux file with LaTEX, and to the file \jobname.pst with other macro packages. (Even then, there is no guarantee it will get the spacing right.)

Here are two exaggerated examples that illustrates the difference between relative and absolute spacing between levels:



```
\pstree[levelsep=1cm,radius=2pt]{\Tc{3pt}}{%
  \TC*
  \pstree{\Tc{3pt}}{%
    \Tc*{15pt}
    \TC*}
  \TC*}
```

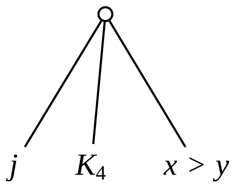


```
\psset{levelsep=*1cm,radius=2pt}
\pstree{\Tc{3pt}}{%
  \TC*
  \pstree{\Tc{3pt}}{%
    \Tc*{15pt}
    \TC*}
  \TC*}
```

If you set the

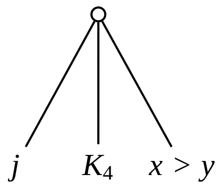
treenodesize=*dim* **Default: -1pt**

to a non-negative value, then PSTricks uses **treenodesize** as a fixed size of the successors (in the direction of their neighbors, i.e., a fixed width for vertical trees and a fixed height/depth for horizontal trees). For example, sometimes it is esthetically pleasing to smooth over small variations in the sizes of the nodes. Compare



```
\pstree[nodesepB=-8pt]{\Tc{3pt}}{%
  \TR{$j$}%
  \TR{$K_4$}%
  \TR{$x>y$}}
```

with



```
\pstree[treenodesize=.4cm,treesep=.3cm,nodesepB=-8pt]{\Tc{3pt}}{%
  \TR{$j$}%
  \TR{$K_4$}%
  \TR{$x>y$}}
```

A subtree's profile varies from level to level. **\pstree** has two modes for fitting subtrees together:

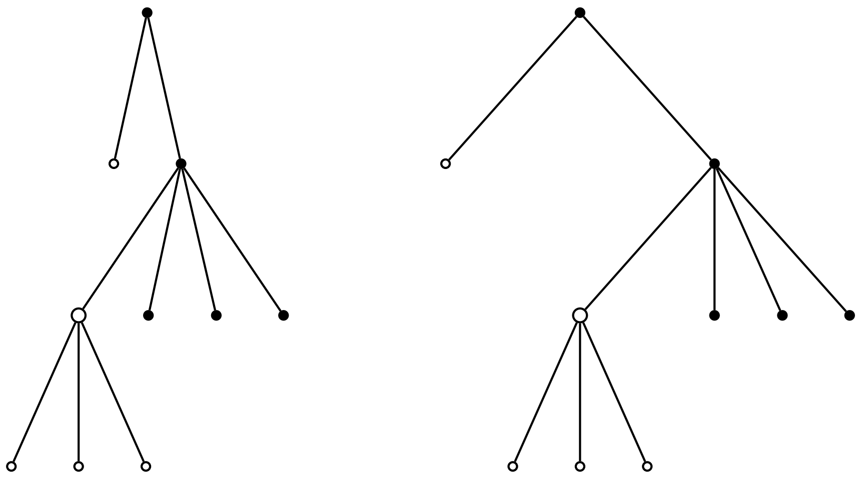
tight With tight fit, the subtrees are fit together so that the minimum distance on any level is **treesep**. This is the default.

loose With loose fit, the distance between the subtrees' bounding boxes is **treesep**. Except when you have exceptionally large intermediate nodes, the effect is that the horizontal distance (or vertical distance, for horizontal trees) between all the terminal nodes is the same.

You select the mode with the

treefit=*tight/loose* **Default: tight**

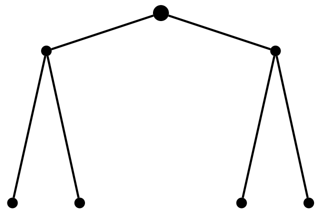
parameter.



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:

thistreesep=*dim* **Default:**
thistreenodesize=*dim* **Default:**
thistreefit=*tight/loose* **Default:**
thislevelsep=dim*** **Default:**

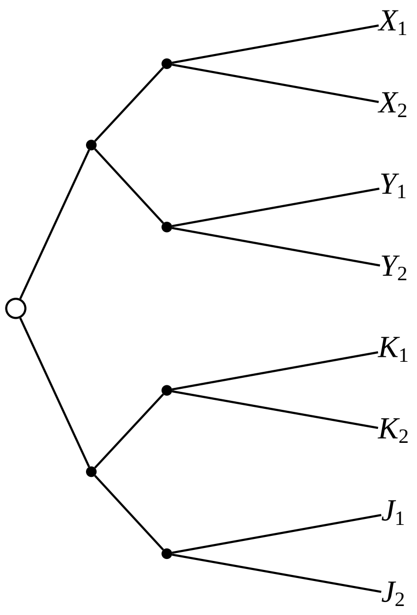
For example:



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

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



16 Edges

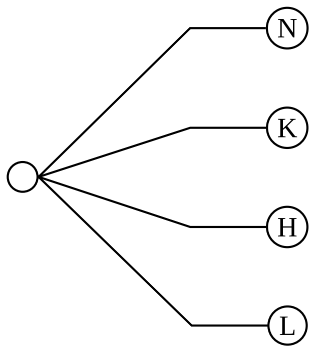
A tree node is really a composite object. In addition to creating a new node, it also draws a node connection between itself and its predecessor, if there is one.

When a tree node has made the new node, the command **\pssucc** is equal to the name of this node, and **\pspred** is equal to the name of its predecessor. Then the tree node executes

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

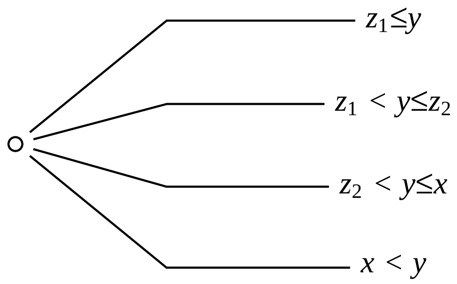
You can define **\psedge** to make whatever node connection you want (see Section **??**). For example, here I use **\ncdiag**, with **armA=0**, to get all the node connections to emanate from the same point in the predecessor:

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



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

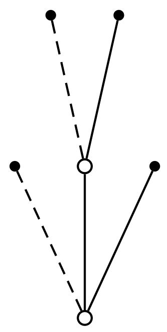
```
$
\def\psedge#1#2{\ncdiagg[angleA=180, armA=-3cm,
   nodesep=4pt]{#2}{#1}}
% Or: \renewcommand{\psedge}[2]{ ... }
\pstree[treemode=R, levelsep=5cm]{\Tc{3pt}}{%
   \Tr{z_1\leq y}
   \Tr{z_1<y\leq z_2}
   \Tr{z_2<y\leq x}
   \Tr{x<y}}
$
```



Another way to define **\psedge** is with the

edge=*command* **Default: \ncline**

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.

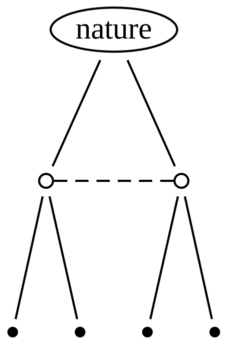


```
\def\dedge{\ncline[linestyle=dashed]}
\pstree[treemode=U,radius=2pt]{\Tc{3pt}}{%
   \TC*[edge=\dedge]
   \pstree{\Tc{3pt}}{\TC*[edge=\dedge]  \TC*}
   \TC*}
```

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

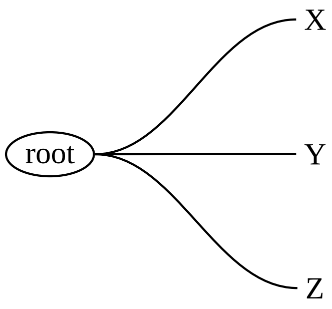
edge is the only parameter which, when set in a tree node's parameter argument, affects the drawing of the node connection (e.g., if you want to change the **nodesep**, your edge has to include the parameter change, or you have to set it before the node).

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:

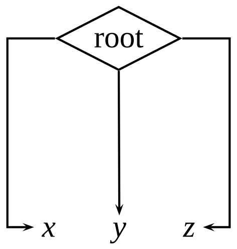


```
\pstree[nodesep=3pt,radius=2pt]{\Toval{nature}}{%
   \pstree{\Tc[name=top]{3pt}}{\TC*  \TC*}
   \pstree{\Tc[name=bot]{3pt}}{\TC*  \TC*}}
 \ncline[linestyle=dashed]{top}{bot}
```

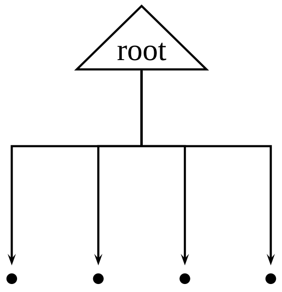
We conclude with the more examples.



```
\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}
   $}
```

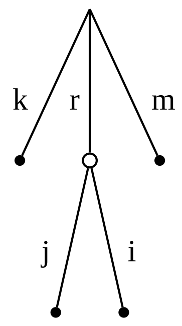


```
\psset{armB=1cm,  levelsep=3cm,  treesep=1cm,
   angleB=-90,  angleA=90,  arrows=<-,  nodesepA=3pt}
\def\psedge#1#2{\ncangle{#2}{#1}}
\pstree[radius=2pt]{\Ttri{root}}{\TC*  \TC*  \TC*  \TC*}
```

17 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.



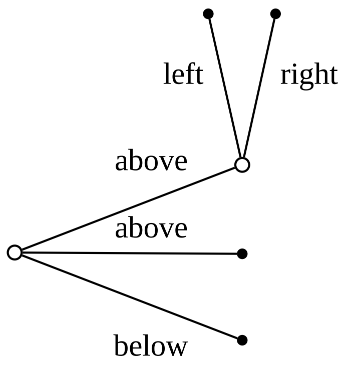
```
\pstree[radius=2pt]{\Tp}{%
   \psset{tpos=.6}
   \TC*  \tlput{k}
   \pstree{\Tc{3pt}  \tlput[labelsep=3pt]{r}}{%
      \TC*  \tlput{j}
      \TC*  \trput{i}}
   \TC*  \trput{m}}
```

Within trees, the **tpos** parameter measures this distance from the predecessor to the successor, whatever the orientation of the true. (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 true:

	Short for:	
Char.	*Vert.*	*Horiz.*
^	**\tlput**	**\taput**
_	**\trput**	**\tbput**

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

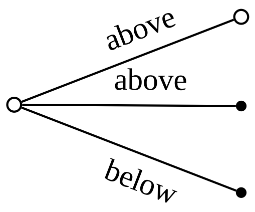


```
\psset{tpos=.6}
\pstree[treemode=R,  thistreesep=1cm,
   thislevelsep=3cm,radius=2pt]{\Tc{3pt}}{%
   \pstree[treemode=U,  xbbr=20pt]{\Tc{3pt}^{above}}{%
      \TC*^{left}
      \TC*_{right}}
   \TC*^{above}
   \TC*_{below}}
```

You can change the character abbreviations with

\MakeShortTab{*char1*}{*char2*}

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



```
\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}}
```

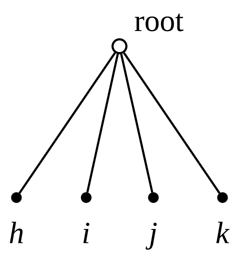
You can put labels on the nodes using **\nput**. However, **\pstree** won't take these labels into account when calculating the bounding boxes.

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

~*[*par*]{*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:



```
\pstree[radius=2pt]{\Tc{3pt}\nput{45}{\pssucc}{root}}{
   \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.,

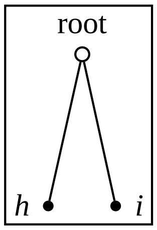
```
\MakeShortTnput{\tnput}
```

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 be setting several parameters.

To position the label on any side of the node (left, right, above or below), set:

tnpos=*l/r/a/b* **Default:**



```
\psframebox{%
  \pstree{\Tc{3pt}~[tnpos=a,tndepth=0pt]{root}}{
    \TC*~[tnpos=l]{$h$}
    \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=*dim* **Default:**

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=*dim* **Default: \ht\strutbox**

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 TEX 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=*dim* **Default: \dp\strutbox**

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=*num* **Default:**

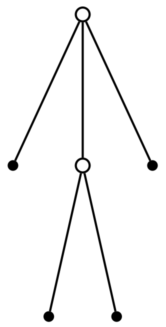
When you leave this empty, **vref** is used instead. Recall that **vref** gives the vertical *distance* from the baseline. Otherwise, the **tnyref** parameter works like the **yref** parameter, giving the fraction of the distance from the bottom to the top of the label.

18 Details

Both **\pstree**'s root node argument and successors argument are processed as LR-boxes, and so everything in Appendix **??**, including the treatment of math and verbatim text, applies, except the following. Because **\pstree** has two arguments, you cannot use **\pslongbox** to define a "long" version of **\pstree**. However, there is a variant **\psTree** of **\pstree** whose syntax is:

\psTree{*root node*} *successors* \endpsTree

For example:



```
\psTree{\Tc{3pt}}
  \TC*
  \psTree{\Tc{3pt}}
    \TC*
    \TC*
  \endpsTree
  \TC*
\endpsTree
```

LaTeX purists can write \begin{psTree} and \end{psTree} instead.

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:

bbl=*dim*	**Default:**
bbr=*dim*	**Default:**
bbh=*dim*	**Default:**
bbd=*dim*	**Default:**
xbbl=*dim*	**Default:**
xbbr=*dim*	**Default:**
xbbh=*dim*	**Default:**
xbbd=*dim*	**Default:**

The x versions increase the bounding box by *dim*, and the others set the bounding box to *dim*. There is one parameter for each direction from the center of the node, **l**eft, **r**ight, **h**eight, and **d**epth.

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).

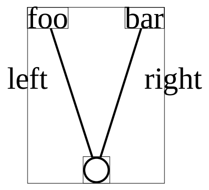
Save any fiddling with the bounding box until you are otherwise finished with the tree.

You can see the bounding boxes by setting the

showbbox=*true/false* **Default:** **false**

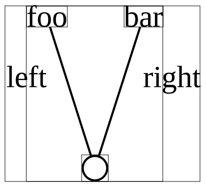
parameter to true. To see the bounding boxes of all the nodes in a tree, you have to set this parameter before the tree.

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



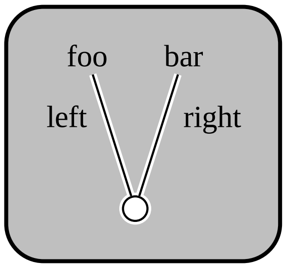
```
\psset{tpos=.6,showbbox=true}
\pstree[treemode=U]{\Tc{5pt}}{%
   \TR{foo}^{left}
   \TR{bar}_{right}}
```

Here is how we fix it:



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

Now we can frame the tree:



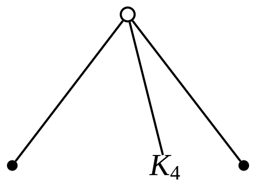
```
\psframebox[fillstyle=solid,fillcolor=lightgray,framesep=14pt,
    linearc=14pt,cornersize=absolute,linewidth=1.5pt]{%
    \psset{tpos=.6,border=1pt,nodesepB=3pt}
    \pstree[treemode=U,xbbl=8pt,xbbr=14pt]{%
        \Tc[fillcolor=white,fillstyle=solid]{5pt}}{%
        \TR*{foo}^{left}
        \TR*{bar}_{right}}}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

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

You can also adjust the distance between successors with the

\tspace{*dim*}

command.



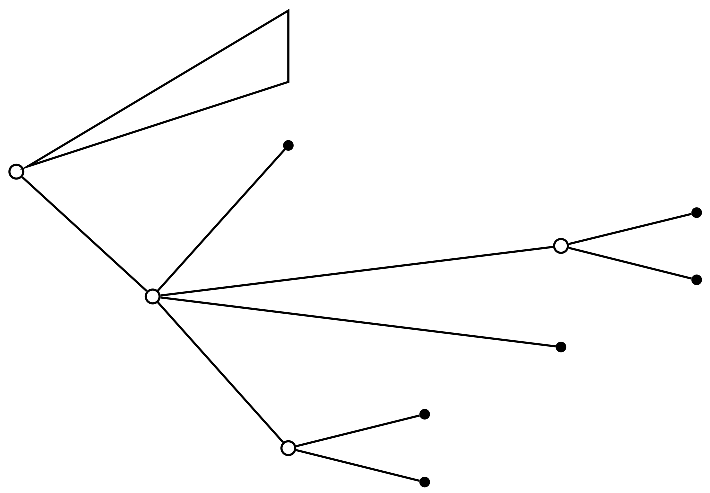
```
\pstree{\Tc{3pt}}{%
    \Tc*{2pt}%
    \tspace{1cm}
    \TR*{$K_4$}%
    \Tc*{2pt}}
```

To skip levels, use

\skiplevel*[*par*]**{*nodes or subtrees*}**
\skiplevels*[*par*]**{*int*}** ***nodes or subtrees*** **\endskiplevels**

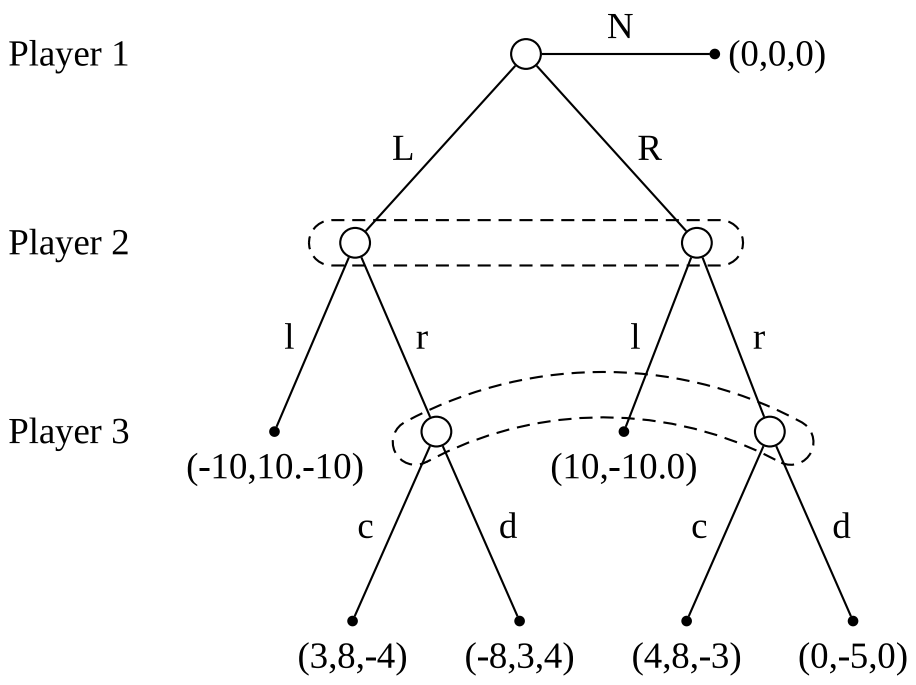
These are kind of like subtrees, but with no root node.

```
\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*}}}
```



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**.

```
\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}
```


Player 1
Player 2
Player 3
N
(0,0,0)
L
R
l
r
l
r
(-10,10.-10)
(10,-10.0)
c
d
c
d
(3,8,-4)
(-8,3,4)
(4,8,-3)
(0,-5,0)