Manual for `egameps.sty`

by
Martin J. Osborne
`martin.osborne@utoronto.ca`

Version 1.1, June 2004
(based on `egame.sty`, November 1997)

1. Introduction

`egameps.sty` is a LaTeX2e style file for drawing extensive games. It is intended to have the capability of drawing any extensive game. It is not very fast (probably because it is not written very efficiently, but possibly simply because it needs to make a lot of computations). The latest version is available at `http://www.economics.utoronto.ca/osborne/latex/`.

The style requires the PSTricks package (available on CTAN; documented in *The LaTeX Graphics Companion*) and a means of printing a Postscript file (e.g. a Postscript printer, or a non-Postscript printer and the program Ghostscript). Because the style uses the PSTricks macros, it is incompatible with `pdftex`. You may produce a pdf file by first creating a dvi file normally, then using `dvips` to create a Postscript file, and finally using Ghostscript to create a pdf file from the Postscript file. (If your operating system is DOS or Windows, you can automate this process in a batch file. If your text editor is sufficiently configurable, you will be able to initiate the process with a single keystroke.) Alternatively, you use the `pdftricks` package to finesse the limitation of `pdftex`; see Section 12.

2. Installation

- Put `egameps.sty` in a directory from which TeX reads input files. (In MiKTeX the directory might be something like `\miktex\localtexmf\tex\latex\` or a subdirectory thereof.)
- Let TeX know that `egameps` has arrived. (In MiKTeX, "refresh the filename database".)
- Check that you have PSTricks. If you do not, get it from CTAN.
- To use the package in a document, put the lines
  ```
  \usepackage{pstricks}
  \usepackage{pstcol}
  \usepackage{egameps}
  ```

in the preamble. To use the macro `\ctmbarc` you need to include also the line
`\usepackage{pst-3d}`

3. Using the style

To draw a game using the style, first break the game into components, each consisting of a node together with the branches that emanate from it, the names with which the actions are labeled, and the players' payoffs if the nodes at the end of the branches are terminal. To draw each component, calls to two macros are needed. First `\putbranch` is called, which sets the position of the node and the slope and length of the branch(es), then either `\ib`, `\iib`, or `\iiib`. The macro `\ib` draws a single branch, while `\iib` and `\iiib` draw two and three branches respectively. If the node has more than three branches, calls to a combination of these macros are needed.

A game is begun by a call of the type

`\begin{egame}400,500)`

which starts a `pspicture`, with dimensions $(400, 500)$ and the unitlength equal to the default of 0.1mm, or

`\begin{egame}(400,500)[1mm]`

which starts a `pspicture`, with dimensions $(400, 500)$ and sets the unitlength at 1mm.

The next section gives examples that illustrate many of the features of the package. Precise descriptions of the macros are given in Section 5.

4. Examples

The game in Figure 1 is produced by the following code.

```
\begin{figure}[htb]
\hspace*{\fill}
\begin{egame}(600,280)
%
% put the initial branch at (300,240), with (x,y) direction
% (2,1), and horizontal length 200
\putbranch(300,240)(2,1){200}
%
% give the branch two actions, label it for player 1,
% and label the actions $L$ and $R$
\iib{1}{$L$}{$R$}
```

```
%
% put a branch at (100,140), with (x,y) direction
% (1,1) and horizontal length 100
\putbranch(100,140)(1,1){100}
%
% give the branch two actions, omit a player label,
% label the actions $a$ and $b$, and assign the payoffs
% $1,0$ and $2,3$ to these actions
\iib{}{$a$}{$b$}[$1,0$][$2,3$]
%
% put a branch at (500,140), with (x,y) direction (1,1)
% and horizontal length 100
\putbranch(500,140)(1,1){100}
%
% give the branch two actions, omit a player label,
% label the actions $c$ and $d$, and assign the payoffs
% $0,1$ and $-1,0$ to these actions
\iib{}{$c$}{$d$}[$0,1$][$-1,0$]
%
% draw an information set between the nodes at (100,140)
% and (500,140)
\infoset(100,140){400}{2}
%
\end{egame}
\hspace*{\fill}
\caption[]{An extensive game}\label{f:one}
\end{figure}
```

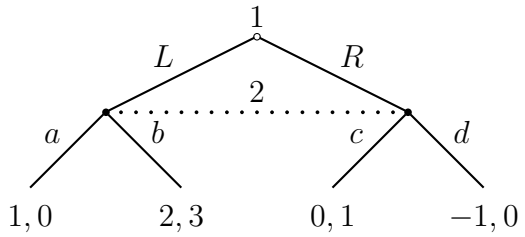


Figure 1. An extensive game

Another example, illustrating more features, is produced by the following code, and is shown in Figure 2.

```
\begin{figure}[htb]
```

```
\hspace*{\fill}
%
% fill boxes containing payoffs with a solid red color
\renewcommand{\egpayboxfillstyle}{solid}
\renewcommand{\egpayboxfillcolor}{red}
% add a bit of separation around the payoffs
\egpayoffboxsep=1mm
%
% fill boxes containing action labels with a solid blue color
\renewcommand{\egalboxfillstyle}{solid}
\renewcommand{\egalboxfillcolor}{blue}
\egactionboxsep=1mm
%
% put player labels in green circles
\renewcommand{\egplbox}{c}
\renewcommand{\egplboxlinestyle}{solid}
\renewcommand{\egplboxlinecolor}{green}
\egplayerboxsep=1mm
%
\begin{egame}(500,280)
%
% put the initial branch at (300,240), with (x,y) direction
% (2,1), and horizontal length 200
\putbranch(300,240)(2,1){200}
%
% give the branch three actions, label it for player 1,
% label the actions $L$, $M$, and $R$, and make the middle
% and right actions terminal, with payoffs $1,2$ and $0,1$
\iiib{1}{$L$}{$M$}{$R$}[][$1,2$][$0,1$]
%
% put a branch at (100,140), with (x,y) direction
% (1,1) and horizontal length 100
\putbranch(100,140)(1,1){100}
%
% give the branch two actions, make the left-hand one dashed,
% label the branch for player 2, putting
% the player label to the top left of the node,
% label the actions $a$ and $b$, and assign the payoffs
% $1,0$ and $2,3$ to these actions
\iib[linestyle=dashed][]{2}[l]{$a$}{$b$}[$1,0$][$2,3$]
```

```
%
\end{egame}
\hspace*{\fill}
\caption[]{Another extensive game}\label{f:two}
\end{figure}
```

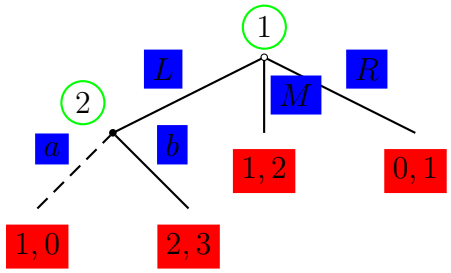


Figure 2. Another extensive game

Yet another example, illustrating still more features, is produced by the following code, and is shown in Figure 3.

```
\begin{figure}[htb]
\hspace*{\fill}
%
\begin{egame}(600,480)
%
% put the initial branch at (300,240), with (x,y) direction
% (1,0), and horizontal length 200
\putbranch(300,240)(1,0){200}
%
% give the branch two actions, label it for player $c$,
% and label the actions ${1\over 2}$ and ${1\over 2}$
\iib{$c$}{${1\over 2}$}{${1\over 2}$}
%
% put a branch at (100,240), with (x,y) direction
% (0,1), going right, and vertical length 100.  (Notes: If the
% branch were specified as going left, it would look the same,
% but the player label would be in the wrong place.  The
% third mandatory argument of \putbranch is the horizontal
% distance unless the branch is vertical, in which case it is
% the vertical distance.)
\putbranch(100,240)(0,1)[r]{100}
%
```

```
% give the branch two actions, label it for player 1,
% and label the actions $a$ and $b$
\iib{1}{$a$}{$b$}
%
% put a branch at (500,240), with (x,y) direction
% (0,1), going left, and vertical length 100.
\putbranch(500,240)(0,1)[l]{100}
%
% give the branch two actions, label it for player 1,
% and label the actions $b$ and $a$
\iib{1}{$b$}{$a$}
%
% put an information set at (100,340), of length 400,
% assigned to player 2
\infoset(100,340){400}{2}
%
% put an information set at (100,140), of length 400,
% assigned to player 2
\infoset(100,140){400}{2}
%
% put a branch at (100,340), with (x,y) direction
% (1,1), going up, and horizontal length 100.
\putbranch(100,340)(1,1)[u]{100}
%
% specify a positive horizontal shift for the action labels,
% which slides the ones on the right-hand branches to the
% right along the branches and the ones on the left to the
% left, improving the label positions
\egalhshift=20
%
% give the branch two actions, give it no player label,
% label the actions $L$ and $R$, and put payoffs of
% $-1,0$ and $0,-1$.
\iib{}{$L$}{$R$}[$-1,0$][$0,-1$]
%
% put a branch at (500,340), with (x,y) direction
% (1,1), going up, and horizontal length 100.
\putbranch(500,340)(1,1)[u]{100}
%
\egalhshift=20
```

```
%
% give the branch two actions, give it no player label,
% label the actions $L$ and $R$, and put payoffs of
% $1,0$ and $0,1$.
\iib{}{$L$}{$R$}[$1,0$][$0,1$]
%
% put a branch at (100,140), with (x,y) direction
% (1,1), going down, and horizontal length 100.
\putbranch(100,140)(1,1)[d]{100}
%
\egalhshift=20
%
% give the branch two actions, give it no player label,
% label the actions $L$ and $R$, and put payoffs of
% $2,0$ and $0,2$.
\iib{}{$L$}{$R$}[$1,0$][$0,1$]
%
% put a branch at (500,140), with (x,y) direction
% (1,1), going down, and horizontal length 100.
\putbranch(500,140)(1,1)[d]{100}
%
\egalhshift=20
%
% give the branch two actions, give it no player label,
% label the actions $L$ and $R$, and put payoffs of
% $3,0$ and $0,3$.
\iib{}{$L$}{$R$}[$1,0$][$0,1$]
%
\end{egame}
\hspace*{\fill}
\caption[]{Yet another extensive game}\label{f:three}
\end{figure}
```

The next example, shown in Figure 4, shows how to combine calls to `\iib` to draw nodes followed by four branches.

```
\begin{figure}[htb]
\hspace*{\fill}
\begin{egame}(1200,380)
%
```

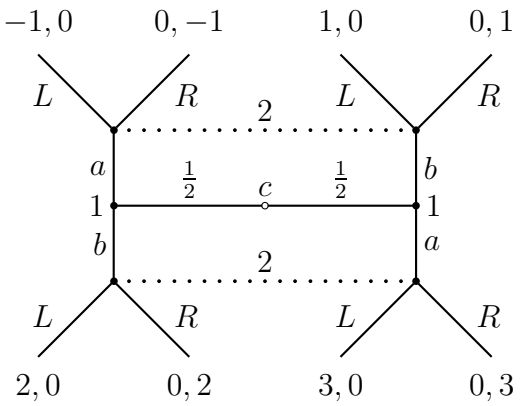


Figure 3. Yet another extensive game

```
% put an initial node at (700,340), with (x,y) direction
% (3,1), and horizontal length 600
\putbranch(700,340)(3,1){600}
%
% give the branch two actions, label it for player $1$,
% label the actions $A$ and $D$, and make the right-
% hand node terminal, with payoffs $0,1$.
\iib[linecolor=red][]{$1$}{$A$}{$D$}[][$0,1$]
%
% to add two more branches to the initial node, force
% the branch to be initial
\initialtrue
% and specify the direction (1,1) and the horizontal length 200
\putbranch(700,340)(1,1){200}
%
% tighten the spacing between labels and branches, to improve
% appearance (given the other branches)
\egactionlabelsep=0.5mm
% give the branch two actions and label the actions $B$ and $C$
\iib{}{$B$}{$C$}
%
% reset default spacing
\egactionlabelsep=1mm
% put a branch at (100,140), with direction (1,1)
% and horizontal length 100.
\putbranch(100,140)(1,1){100}
%
```

```
% give the branch two actions, omit a player label,
% label the actions $a$ and $b$, and put payoffs
\iib[][linecolor=red]{}{$a$}{$b$}[$2,1$][$4,0$]
%
% put a branch at (500,140), with direction (1,1)
% and horizontal length 100.
\putbranch(500,140)(1,1){100}
%
% give the branch two actions, omit a player label,
% label the actions $a$ and $b$, and put payoffs
\iib[][linecolor=red]{}{$a$}{$b$}[$1,3$][$-1,0$]
%
% put a branch at (900,140), with direction (1,1)
% and horizontal length 100.
\putbranch(900,140)(1,1){100}
%
% give the branch two actions, omit a player label,
% label the actions $a$ and $b$, and put payoffs
\iib[][linecolor=red]{}{$a$}{$b$}[$-1,0$][$2,1$]
%
% put an information set at (100,140), of length 800,
% assigned to player 2
\infoset(100,140){800}{2}
%
\end{egame}
\hspace*{\fill}
\caption[]{Yet another extensive game}\label{f:four}
\end{figure}
```

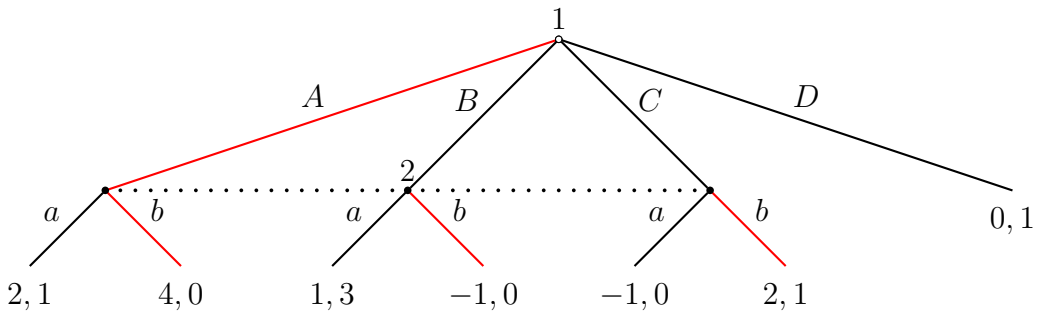


Figure 4. Yet another extensive game

5. Description of macros

`\begin{egame}(`*width*`,`*height*`)[`*unitlength*`]`
begins an extensive game of width *width* and height *height* and optionally sets the unitlength to be *unitlength* (default 0.1mm). In the game, all distances are given as integers, which are interpreted as multiples of the unitlength. These integers should (probably) be divisible by two, so that the integer arithmetic employed by TEX doesn't lose accuracy when numbers are divided by two. I have tested the macros thoroughly only with the default unitlength of 0.1mm; unless there is a compelling reason to use some different unitlength, I suggest sticking to 0.1mm. (`\begin{egame}(w,h)` starts a `pspicture` environment (defined by PSTricks) of width `w` and height `h`.)

Permissible values:

width, *height* Any pair of integers. (You need to calculate these numbers.)

unitlength Any dimension. Default: 0.1mm.

`\end{egame}`
ends an extensive game.

`\putbranch(`*x-coord*`,`*y-coord*`)(`*h-incr*`,`*v-incr*`)[`*direction*`]{`*length*`}`
sets up the parameters for a branch at the point (*x-coord*,*y-coord*), with direction parameter (*h-incr*,*v-incr*), optional direction *direction*, and length *length*. Note that this macro merely sets up the parameters for a call to `\ib` (one branch), `\iib` (two branches), `\iiib` (three branches), or `\ctmb` (continuum of branches); it does not draw anything. The way in which the direction parameter (*h-incr*,*v-incr*) is interpreted depends on whether `\ib`, `\iib`, or `\iiib` is used to draw the branches. The *length* is the horizontal distance between the ends of one of the branches to be drawn, unless *h-incr* is 0, in which case *length* is the vertical distance between the ends of one of the branches.

Permissible values:

(*x-coord*,*y-coord*) Any pair of integers.

(*h-incr*,*v-incr*) Any pair of integers except `(0,0)`.

direction `d` (down), `u` (up), `r` (right), or `l` (left); default `d`. (The direction can be changed also globally, by specifying `\egdirection{`*direction*`}` before the call to `\putbranch`.)

length Any positive integer.

`\ib[`*branchstyle*`]{`*player-name*`}[`*player-label-position*`]{`*action-label*`}[`*action-label-position*`][`*payoffs*`]`
puts a single branch with the parameters of the preceding call to `\putbranch`, optionally using the PSTricks style *branchstyle*, assigns it the player name *player-name*, optionally positions the player label relative to the node according to *player-label-position*, labels the action *action-label*, optionally positions the label according to *player-label-position*, and optionally adds the payoffs *payoffs*. A single optional argument at the end is interpreted as follows: if its value is `o` (outside), `i` (inside), or `c` (centered), it determines the position of the action label; otherwise, it is a payoff label. If there are two optional arguments, the first must be `o` (outside), `i` (inside), or `c` (centered), determining the position of the action label, and the second is a payoff label. (If you do not want an optional positioning argument and you want your payoff label to be "o", "i", or "c", here's a workaround: put the label in an `hbox`, as in `\hbox{o}`. (Note that you do not need to do this if your payoff label is `$o$`, etc.—only if it is just "o".)

The first branch in any `egame` is taken to be the initial branch of the game; its beginning node is indicated by `\eginode`, the default of which is a small circle. Subsequent branches are taken to be noninitial, and are indicated by `\egnode`, the default value of which is a small disk. To force a node to be initial, specify `\initialtrue` before it; to force a node to be noninitial, specify `\initialfalse` before it. To change the appearance of nodes, see Section 8.

Permissible values:

branchstyle Any PSTricks linestyle (e.g. `linecolor=red`, `linestyle=dashed`, `linewidth=2pt`, `doubleline=true`).

Default: `linecolor=black`, `linestyle=solid`, `linewidth=0.8pt`.

player-name Any character string.

player-label-position Either `o` (centered over the node), or

- If the direction of the branch (as specified either by `\egdirection` or by an optional argument of `\putbranch`) is `d` or `u`, either `l` (above/below and to the left) and `r` (above/below and to the right).
- If the direction of the branch is `r` or `l`, either `u` (to the left/right and up) and `d` (to the left/right and down).

Default: is to center the label above for `\egdirection=d`, below for `\egdirection=u`, to the left of the node for `\egdirection=r`, or to the right of the node for `\egdirection=l`.

action-label Any character string.

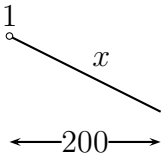
action-label-position `o` (outside), `i` (inside), or `c` (centered on branch).

Default: `o`. (For more on action label positioning, see Section 6.)

payoffs Any character string.

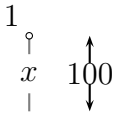
Examples:

```
\putbranch(0,100)(2,-1){200}
\ib{1}{$x$}
```

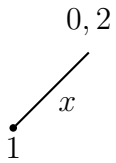


Note that in the following example the `1` in the direction pair `(0,1)` is interpreted as sending the branch down, given that the `\egdirection` is `d`. (That is, once you specify the `\egdirection` (or leave it at the default `d`) you do not need to worry about getting the sign of the *v-incr* correct (if the direction is `d` or `u`) or the sign of the *h-incr* correct (if the direction is `l` or `r`).

```
\putbranch(0,100)(0,1){100}
\ib[linecolor=gray]{1}[l]{$x$}[c]
```

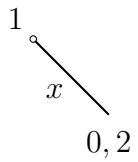



```
\initialfalse
\putbranch(0,40)(1,1)[u]{100}
\ib{1}{$x$}[$0,2$]
```



Notice that the optional direction specifier `u` in `\putbranch` affects not only the direction of the branch (which is determined by the argument `(1,1)`), but also the placement of the player label relative to the starting node.

```
\initialfalse
\putbranch(0,100)(1,1){100}
\ib{1}[l]{$x$}[i][$0,2$]
```



`\iib[`*style1*`][`*style2*`]{`*player-name*`}[`*player-label-position*`]{`*action-label1*`}{`*action-label2*`}[`*action-label-position*`][`*payoffs1*`][`*payoffs2*`]`
puts two branches with the parameters of the preceding call to `\putbranch`, optionally using the PSTricks style *branchstyle1* for the left or upper branch and the style *branchstyle2* for the other branch (or *branchstyle1* for both branches, if there is only one optional argument), assigns it the player name *player-name*, optionally positions the player label relative to the node according to *player-label-position*, labels the left/upper action with *action-label1* and the right/lower action with *action-label2*, optionally using the position *action-label-position*, and optionally adds payoffs *payoffs1* and *payoffs2*. The signs of the direction parameters (*h-incr*,*v-incr*) in the preceding `\putbranch` call are ignored; the directions of the branches is determined by the direction of the branch (as specified either by `\egdirection` or by the optional argument of `\putbranch`). If, for example, the direction is `d`, then one branch goes down and to the left and the other goes down and to the right.

The first branch in any `egame` is taken to be the initial branch of the game; its beginning node is indicated by `\eginode`. Subsequent branches are taken to be non-initial, and are indicated by `\egnode`. To force a node to be initial, specify `\initialtrue` before it; to force a node to be noninitial, specify `\initialfalse` before it.

Permissible values:

style1, *style2* Any PSTricks line styles; see `\ib`. If there is only one style present, it is applied to both branches. If there are two styles, the first is applied to the right hand branch and the second to the left hand branch if the game direction is up or down, and the first is applied to the lower branch and the second to the upper branch if the game direction is left or right.

player-name Any character string.

player-label-position Either `o` (centered over the node), or

- If the direction of the branch (as specified either by `\egdirection` or by an optional argument of `\putbranch`) is `d` or `u`, either `l` (above/below and to the left) and `r` (above/below and to the right).
- If the direction of the branch is `r` or `l`, either `u` (to the left/right and up) and `d` (to the left/right and down).

Default: is to center the label above for `\egdirection=d`, below for `\egdirection=u`, to the left of the node for `\egdirection=r`, or to the right of the node for `\egdirection=l`.

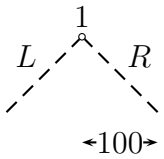
action-label1 and *action-label2* Any character strings.

action-label-position `o` (outside), `i` (inside), or `c` (centered). Default: `o`.

payoffs1 and *payoffs2* Any character strings.

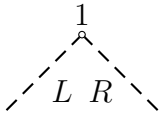
Examples:
Action labels "outside" (default):

```
\putbranch(100,140)(1,1){100}
\iib[linestyle=dashed]{1}{$L$}{$R$}
```



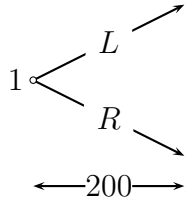
Action labels "inside":

```
\putbranch(100,140)(1,1){100}
\iib[linestyle=dashed]{1}{$L$}{$R$}[i]
```

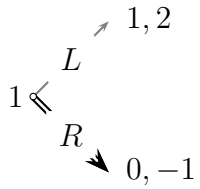


Action labels "centered":

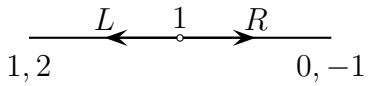
```
\renewcommand{\egarrowstyle}{e}
\putbranch(0,140)(2,1)[r]{200}
\iib{1}{$L$}{$R$}[c]
```




```
\renewcommand{\egarrowstyle}{e}
\putbranch(0,100)(1,1)[r]{100}
\iib[linecolor=gray][doubleline=true]{1}{$L$}{$R$}[$1,2$][$0,-1$]
```




```
\renewcommand{\egarrowstyle}{m}
\psset{arrowscale=2}
\putbranch(200,40)(1,0)[d]{200}
\iib{1}{$L$}{$R$}[$1,2$][$0,-1$]
```



`\iiib[`*style1*`]{`*player-name*`}[`*player-label-position*`]{`*action-label1*`}{`*action-label2*`}{`*action-label3*`}[`*action-label-position*`][`*payoffs1*`][`*payoffs2*`][`*payoffs3*`]`
puts three branches with the parameters of the preceding call to `\putbranch`, optionally all in the PSTricks style *style*, assigns it the player name *player-name*, optionally positions the player label relative to the node according to *player-label-position*, and labels the left/top action with *action-label1*, the middle action with *action-label3*, and the right/bottom action with *action-label2*, optionally positioning the labels according to *action-label-position* and putting the *payoffs1*, *payoffs2*, and *payoffs3*. The signs of the direction parameters (*h-incr*,*v-incr*) in the preceding `\putbranch` call are ignored; the directions of the branches are determined by the direction of the branch (as specified either by `\egdirection` or by the optional argument of `\putbranch`). If, for example, the direction is `d`, then one branch goes down and to the left, one goes straight down, and one goes down and to the right.

The first branch in any `egame` is taken to be the initial branch of the game; its beginning node is indicated by a small circle. Subsequent branches are taken to be non-initial, and are indicated by `\egnode`, the default value of which is a small disk. To force a node to be initial, specify `\initialtrue` before it; to force a node to be noninitial, specify `\initialfalse` before it.

Permissible values:

style Any PSTricks line style. [Note that only one style is allowed, which applies to all three branches. If you want the branches to have different styles, you need to use a combination of `\ib` and `\iib`.]

player-name Any character string.

player-label-position Either `o` (centered over the node), or

- If the direction of the branch (as specified either by `\egdirection` or by an optional argument of `\putbranch`) is `d` or `u`, either `l` (above/below and to the left) and `r` (above/below and to the right).
- If the direction of the branch is `r` or `l`, either `u` (to the left/right and up) and `d` (to the left/right and down).

Default: is to center the label above for `\egdirection=d`, below for `\egdirection=u`, to the left of the node for `\egdirection=r`, or to the right of the node for `\egdirection=l`.

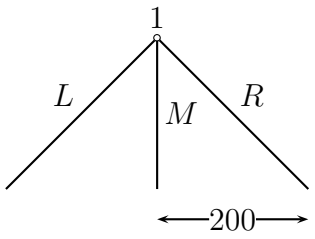
action-label1, *action-label2*, and *action-label3* Any character strings.

action-label-position `o` (outside), `i` (inside), or `c` (centered). Default: `o`.

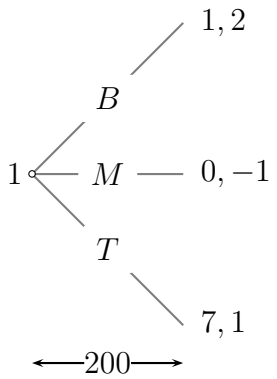
payoffs1, *payoffs1*, and *payoffs3* Any character strings.

Examples:

```
\putbranch(200,240)(1,1){200}
\iiib{1}{$L$}{$M$}{$R$}
```



```
\putbranch(30,210)(1,1)[r]{200}
\iiib[linecolor=gray]{1}{$B$}{$M$}{$T$}[c][$1,2$][$0,-1$][$7,1$]
```



`\ctmb[`*style*`]{`*player-name*`}[`*player-label-position*`](`*h-incr*`,`*v-incr*`)`
`{`*action-label*`}[`*action-label-position*`][`*payoffs*`]`
draws a continuum of branches starting with the parameters of the previous call to `\putbranch`, with a single branch drawn as a line with PSTricks style *style*, player *player*, with the label optionally positioned according to *player-label-position*, and slope *h-incr,v-incr*, labeled with *action-label*, which is optionally positioned according to *action-label-position*, and optionally with payoffs *payoffs*.

The color of the triangle representing the continuum of branches is `\ctmfillcolor` (default `verylightgray` (defined in the style)).

Permissible values:

style Any PSTricks line style. [Note that this is the style of the single branch, **not** of the triangle.]

player-name Any character string.

player-label-position Either `o` (centered over the node), or

- If the direction of the branch (as specified either by `\egdirection` or by an optional argument of `\putbranch`) is `d` or `u`, either `l` (above/below and to the left) and `r` (above/below and to the right).
- If the direction of the branch is `r` or `l`, either `u` (to the left/right and up) and `d` (to the left/right and down).

Default: is to center the label above for `\egdirection=d`, below for `\egdirection=u`, to the left of the node for `\egdirection=r`, or to the right of the node for `\egdirection=l`.

(*h-incr*,*v-incr*) Any pair of integers.

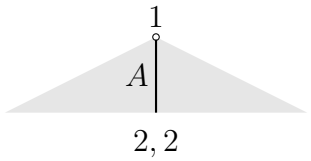
action-label Any character string.

action-label-position `o` (outside), `i` (inside), or `c` (centered). Default: `o`.

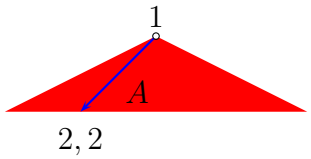
payoffs Any character strings.

Examples:

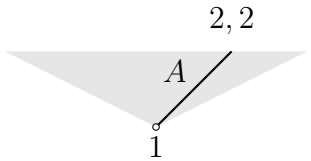
```
\putbranch(50,240)(2,-1)[d]{200}
\ctmb{1}(0,1){$A$}[i][$2,2$]:
```




```
\renewcommand{\ctmfillcolor}{red}
\renewcommand{\egarrowstyle}{e}
\putbranch(50,240)(2,-1)[d]{200}
\ctmb[linecolor=blue]{1}(-1,1){$A$}[i][$2,2$]:
```




```
\putbranch(50,0)(2,1)[u]{200}
\ctmb{1}(1,1){$A$}[i][$2,2$]:
```



`\ctmbarc[`*style*`]{`*player-name*`}[`*player-label-position*`](`*h-incr*`,`*v-incr*`)`
`{`*action-label*`}[`*action-label-position*`][`*payoffs*`]`
represents a continuum of branches by an arc. The branches start with the

parameters of the previous call to `\putbranch`, with a single branch and the arc drawn using the PSTricks style *style*, player *player*, the label optionally positioned according to *player-label-position*, the line with slope *h-incr,v-incr*, labeled with *action-label* optionally positioned according to *action-label-position*, and optionally with payoffs *payoffs*.

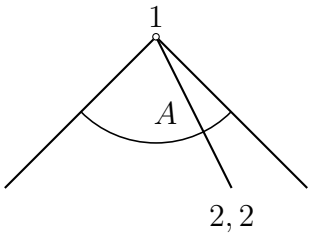
This macro requires the package `pst-3d` *to be loaded after* `pstricks`. *In the preamble to your document, include the line* `\include{pst-3d}`. (`\pstricks` itself includes the needed macro, but its definition appears to differ from the one in `pst-3d`, which is the definition `\ctmbarc` needs.)

The line drawn by the macro is the intersection of (*i*) a circle centered at the start of the branch with radius equal to a half of the distance to the following node and (*ii*) the triangle defined by the starting node and the two following nodes. For some parameter values, this line is not a full arc—see the last example.

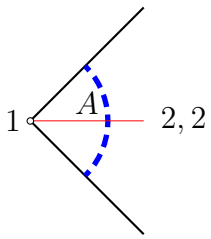
The options are the same as those for `\ctmb`. The style of the arc is controlled by the parameters `\egarclinestyle` (possible values `solid`, `dashed`, `dotted`, and `none`; default `solid`), `\egarclinewidth` (any dimension; default `0.6pt`), `\egarclinecolor` (any color; default `black`).

Examples:

```
\putbranch(50,240)(2,2)[d]{200}
\ctmbarc{1}(1,-2){$A$}[i][$2,2$]:
```

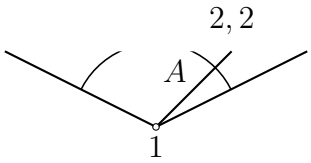



```
\def\egarclinestyle{dashed}
\def\egarclinewidth{2pt}
\def\egarclinecolor{blue}
\putbranch(0,150)(2,2)[r]{150}
\ctmbarc[linewidth=0.3pt,linecolor=red]{1}(1,0){$A$}[$2,2$]:
```



In the next example part of the arc extends beyond the triangle defined by the three nodes, and consequently does not appear.

```
\putbranch(50,0)(2,1)[u]{200}
\ctmbarc{1}(1,1){$A$}[i][$2,2$]:
```



`\infoset(`*x-coord*`,`*y-coord*`)[`*direction*`]{`*length*`}{`*player-name*`}`
`[`*player-label-position*`]`
draws an information set starting at (*x-coord*,*y-coord*), optionally with direction *direction*, of length *length*, with player label *player-name*. If the direction of the game is either down or up, the information set is horizontal; if the direction of the game is either right or left, the information set is vertical. The player label is positioned at the middle of the information set, either above, below, to the left, or to the right of it, depending on the direction of the game. (Its position can be adjusted by adding some space before or after the player name. For example,
`\infoset(100,200){400}{1\hspace{10mm}}`
centers the box containing `1\hspace{10mm}` relative to the information set, thus moving the label by about 5mm.

The dot character used in the information set is given by `\infosetdot`, the default value of which is `\pscircle*{2.5}`; the spacing between the dots is set by `\infosetdotsep`, the default value of which is 20. (See Section 8.)

Permissible values:

(*x-coord*,*y-coord*) Any pair of integers.

direction `h` (horizontal) or `v` (vertical). If none is specified, it is assumed to be `h` if the game direction is `d` or `u`, and `v` if the game direction is `r` or `l`.

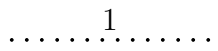
length Any nonnegative integer.

player-name Any character string.

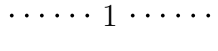
player-label-position `o` (over), `u` (up), or `d` (down) if direction is `h`, `o` (over), `l` (left), or `r` (right) if direction is `v`.

Examples:

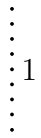
```
\egdirection{d}
\infoset(0,0){300}{1}
```




```
\egdirection{d}
\infoset(0,0){300}{1}[o]
```




```
\egdirection{d}
\infoset(0,0)[u]{200}{1}
```



6. Positioning action labels

The following algorithm is used to position an action label. The "reference point" for a branch is its midpoint. An action label is surrounded by a box (which may be colored and bordered by a line), from which it is separated by `\egactionboxsep`, then this box is surrounded by another box, from which it is separated by `\egactionlabelsep`. Then, for example, for a label positioned above a downward-sloping branch, the bottom left-hand corner of the outer box is placed at the reference point, as in Figure 5. For a label positioned above a horizontal branch, the bottom center of the outer box is placed at the reference point.

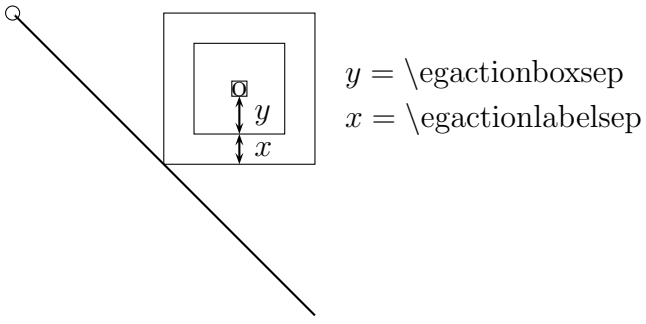


Figure 5. The position of an action label above a downward-sloping branch.

This algorithm is not perfect. (Note that it is not continuous in the slope of the branch.) The positions of labels may need fine tuning, which may be achieved by setting either `\egalhshift` or `\egalvshift` or both to be nonzero (each is an integer, interpreted as a multiple of the unitlength).

`\egalhshift` and `\egalvshift` must be set between `\putbranch` and `\ib`, `\iib`, `\iiib`, or `\ctmb`. (They are set to zero by `\putbranch`.)

If `\egalhshift` is nonzero and `\egalvshift` is not, the reference point for the action label on the single branch in `\ib`, of the *right-hand* branch for directions `d` and `u` and of both branches for directions `r` and `l` in `\iib`, and for the outer branches of `\iiib`, is moved horizontally by this amount and is moved vertically to maintain the same separation from the branch. (That is, the action label slides parallel to the branch; it moves horizontally by the amount `\egalhshift`.) The reference point of the label on the symmetric branch is moved symmetrically. (Note that the label on the middle branch of `\iiib` is not moved. If you want to move it, you need to write separate calls to `\iib` and `\ib`.)

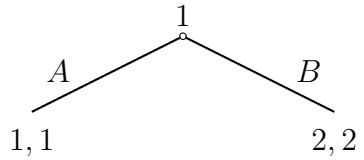
Similarly, if `\egalvshift` is nonzero and `\egalhshift` is not, the reference point for the action label is moved vertically by this amount and is moved horizontally to maintain the same separation from the branch. After a branch is drawn, `\egalhshift` and `\egalvshift` are reset to zero.

Note that the same effect can be acheived with either `\egalhshift` or `\egalvshift`. The two methods are provided simply because in some cases a vertical shift might be more natural to specify, whereas in other cases a horizontal shift might be more natural to specify.

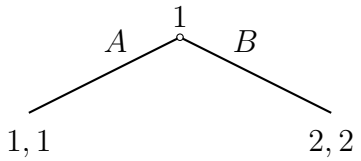
If *both* `\egalhshift` and `\egalvshift` are nonzero, the reference point is moved horizontally by `\egalhshift` and vertically by `\egalvshift`.

Examples:

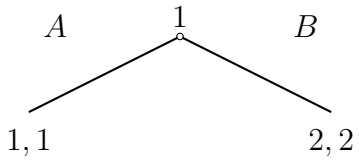
```
\egdirection{d} \egalhshift=40
```



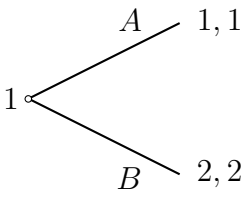
```
\egalhshift=-40
```




```
\egalhshift=40\egalvshift=40
```




```
\egdirection{u}\egalhshift=60
```



7. Arrows

Branches can have arrows either at the end or in the middle, by setting

```
\renewcommand{\egarrowstyle}{e}
```

(for arrows at the end), or

```
\renewcommand{\egarrowstyle}{m}
```

(for arrows in the middle). The positioning of arrows in the middle is not ideal: the *tip* of the arrow is placed in the middle of the branch, while ideally the *middle* of the arrow should be there. I can't see any easy way to improve the placement.

The style of the arrows can be controlled with PSTricks' various parameters (as described, for example, in the LaTeX Graphics Companion).

8. Parameters

`\egdirection`
Direction of game. Possible values: `d` (down), `u` (up), `r` (right), or `l` (left).

Default: `d`. Example: `\egdirection{u}`.

`\initialtrue`, `\initialfalse`
Force a node other than the first one in an `egame` to be initial, or force the first node to be noninitial.

`\eginode`
Object used for initial node. Possible values: any object. Default: `\pscircle [linewidth=0.4pt]{5}`. Example: `\renewcommand{\eginode} {\makebox(0,0){\rule{0.5mm}{0.5mm}}}` (the `\makebox` causes the object to be positioned correctly).

`\egnode`
Object used for nodes. Possible values: any object. Default: `\pscircle*{5}`. Example: `\renewcommand{\egnode}{\makebox(0,0){\rule{0.5mm}{0.5mm}}}` (the `\makebox` causes the object to be positioned correctly).

`\infosetdot`
Object used for "dots" in information sets. Possible values: any object. Default: `\pscircle*{2.5}`. Example: `\renewcommand{\infosetdot} {\pscircle*{5}}`.

`\infosetdotsep`
Spacing between dots in information set, as a multiple of the unit length. Possible values: any positive integer. Default: 20. Example: `\renewcommand {\infosetdotsep}{40}`.

`\egplayerlabelsep`
Spacing used to position box containing player label relative to center of initial node, and relative to center of information set. Possible values: any dimension. Default: 1mm. Example: `\egplayerlabelsep=2mm`.

`\egplayerboxsep`
Distance between edge of box around player label and player label. Possible values: any dimension. Default: 0mm. Example: `\egplayerboxsep=1mm`.

`\egactionlabelsep`
Spacing used to position box containing action label relative to branch. Possible values: any dimension. Default: 0.7mm. Example: `\egactionlabelsep= 1mm`.

`\egactionboxsep`
Distance between edge of box around action label and action label. Possible values: any dimension. Default: 0mm. Example: `\egactionboxsep=1mm`.

`\egpayofflabelsep`
Spacing used to position box containing action label relative to end of branch. Possible values: any dimension. Default: 2mm. Example: `\egpayofflabelsep=1mm`.

`\egpayoffboxsep`
Distance between edge of box around payoffs and payoffs. Possible values: any dimension. Default: 0mm. Example: `\egpayoffboxsep=1mm`.

`\egalpos`
Position of action labels. Possible values: `o` (outside), `i` (inside), `c` (center). Default: `o`. Example: `\renewcommand{\egalpos}{c}`.

`\egalbox`
Type of box for action labels. Possible values: `f` (frame), `c` (circle). Default: `f`. Example: `\renewcommand{\egalbox}{c}`.

`\egalboxlinestyle`
Style of lines around boxes containing action labels. Possible values: `none`, `solid`, `dashed`, `dotted`. Default: `none`. Example: `\renewcommand {\egalboxlinestyle}{solid}`.

`\egalboxlinecolor`
Color of lines around boxes containing action labels. Possible values: any defined color. Default: `black`. Example: `\renewcommand{\egalboxlinecolor} {red}`.

`\egalboxfillstyle`
Style of fill of boxes containing action labels. Possible values: `none`, `solid`, `vlines`, `vlines*`, `hlines`, `hlines*`, `crosshatch`, `crosshatch*`. Default: `none`. Example: `\renewcommand{\egalboxfillstyle}{solid}`.

`\egalboxfillcolor`
Color of fill of boxes containing action labels. Possible values: any defined color. Default: `white`. Example: `\renewcommand{\egalboxfillcolor} {blue}`.

\egpaybox
Type of box for payoffs. Possible values: `f` (frame), `c` (circle). Default: `f`. Example: `\renewcommand{\egpaybox}{c}`.

\egpayboxlinestyle
Style of lines around boxes containing payoffs. Possible values: `none`, `solid`, `dashed`, `dotted`. Default: `none`. Example: `\renewcommand {\egpayboxlinestyle}{solid}`.

\egpayboxlinecolor
Color of lines around boxes containing payoffs. Possible values: any defined color. Default: `black`. Example: `\renewcommand{\egpayboxlinecolor} {gray}`.

\egpayboxfillstyle
Style of fill of boxes containing action payoffs. Possible values: `none`, `solid`, `vlines`, `vlines*`, `hlines`, `hlines*`, `crosshatch`, `crosshatch*`. Default: `none`. Example: `\renewcommand{\egpayboxfillstyle}{solid}`.

\egpayboxfillcolor
Color of fill of boxes containing payoffs. Possible values: any defined color. Default: `white`. Example: `\renewcommand{\egpayboxfillcolor}{gray}`.

\egplbox
Type of box for player labels. Possible values: `f` (frame), `c` (circle). Default: `f`. Example: `\renewcommand{\egplbox}{c}`.

\egplboxlinestyle
Style of lines around boxes containing player labels. Possible values: `none`, `solid`, `dashed`, `dotted`. Default: `none`. Example: `\renewcommand {\egplboxlinestyle}{solid}`.

\egplboxlinecolor
Color of lines around boxes containing player labels. Possible values: any defined color. Default: `black`. Example: `\renewcommand{\egplboxlinecolor} {gray}`.

\egplboxfillstyle
Style of fill of boxes containing player labels. Possible values: `none`, `solid`, `vlines`, `vlines*`, `hlines`, `hlines*`, `crosshatch`, `crosshatch*`. Default: `none`. Example: `\renewcommand{\egplboxfillstyle}{solid}`.

`\egplboxfillcolor`
Color of fill of boxes containing player labels. Possible values: any defined color. Default: `white`. Example: `\renewcommand{\egplboxfillcolor}{gray}`.

`\egarrowstyle`
Style of arrows on branches. Possible values: `n` (none), `e` (end), `m` (mid). Default: `n` (none). Example: `\renewcommand{\egarrowstyle}{m}` (mid). The size of the arrows can be adjusted by using any of the PSTricks parameters—e.g. `\psset{arrowscale=1.5}`.

`\ctmfillcolor`
Color of triangle used to indicate continuum of branches. Possible values: any defined color. Default: `verylightgray`. Example: `\renewcommand{\ctmfillcolor}{red}`.

9. Suggestions

It seems hard to proceed without first drawing the game on a piece of paper, at least roughly. Be sure to allow enough space under (in the case of a downward-pointing game) the terminal nodes—put the nodes that precede the terminal nodes high enough that the bottom of the payoffs will be at height 0. Similarly, specify the height of the game so that the label for the initial player does not poke out the top. If parts of your game poke out of the frame defined by the size you specify for your `egame`, TEX will not warn you, but the spacing above and/or below your game will not be right. (If you specify the height to be too small, for example, the top of your game may overlap the text above it.)

You can float your game in a figure (as I have done in the examples above), or you can put it in the text. (For example, you may use `$$\begin{egame}...\end{egame}$$`). Putting it in the text has the disadvantage that if it's big and happens to start at the bottom of a page then you may get a lot of white space if it doesn't fit on the page.

10. Enhancements

It would be nice to have a graphical interface, like that in LATEXCAD. I'm not capable of writing one.

It would be nice also if the macros could calculate the dimensions of the whole game, so the the user does not have to specify them in the `\begin{game}` call. To make this change looks like a tough project to me.

The label-positioning algorithm could be enhanced, as discussed in Section 6.

11. History

Version 1.0 2001-4-9 (based on EGAME.STY, 1997-10-16)

Version 1.1 2004-6-19 (`\egctmarc` added, following suggestion of Paul Schweinzer).

12. Appendix: using pdftricks with pdftex to create a pdf file

If you want to create a pdf file from your LaTeX file and do not want to take the dvi → ps → pdf route, you may use `pdftricks`. Eric Gartzke reports that the following input produces the expected output. (See his comments for the limitations of this method.) If you find a better solution, please let me know.

```
\documentclass[12pt]{article}
\usepackage[shell]{pdftricks}

\begin{psinputs}
\usepackage{pstricks}
\usepackage{color}
\usepackage{pstcol}
\usepackage{pst-plot}
\usepackage{pst-tree}
\usepackage{pst-eps}
\usepackage{multido}
\usepackage{pst-node}
\usepackage{pst-eps}
\usepackage{egameps}
\end{psinputs}

\usepackage{amsfonts}
\usepackage{graphicx}
\usepackage{amsmath}
\usepackage{setspace}
\usepackage{fullpage}
\usepackage{harvard}
\usepackage{threeparttable}
\usepackage{amssymb}
```

```
\newtheorem{hypothesis}{Hypothesis}

Postscript commands need to be separated from other commands.
pdftricks literally stops the compiling of the TeX file to run
postscript/dvi in the background, generating separate small
TeX files and pdf files of each figure.  Notice that egameps
is called last.  This seems to make a considerable
difference.  Here is an example of a simple egame tree.

\begin{pdfpic}
\begin{figure}[htb]
\hspace*{\fill}
\begin{egame}(600,1250)
\renewcommand{\egarrowstyle}{e}
\renewcommand{\egnode}{\pscircle*{8}}
\putbranch(200,1200)(0,1){150} \ib{$N$}[r]{$~k_{i},~k_{j}$}
\renewcommand{\egarrowstyle}{}
\putbranch(200,1050)(1,1){200} \egdirection{d}
\iib{$i$}[l]{$\begin{array}{c}Status\\Quo\end{array}$}{$Challenge$}[$b_{
i},~1+b_{j}$][]
\putbranch(400,850)(1,1){200}
\iib{$~j$}[r]{$Reject$}{$Accept$}[][$\begin{array}{c}1-
d+b_{i},\\d+b_{j}\end{array}$]
\putbranch(200,650)(1,1){200}
\iib{$i$}[l]{$\begin{array}{c}Back\\Down\end{array}$}{$Persist$}[$\begin
{array}{c}-a_{i},\\1\end{array}$][]
\putbranch(400,450)(1,1){200}
\iib{$j$}[r]{$Fight$}{$Acquiesce$}[$\begin{array}{c}p-k_{i},\\1-p-
k_{j}\end{array}$][$\begin{array}{c}1-d,\\d\end{array}$]
\end{egame}
\setcounter{figure}{1}
\hspace*{\fill} \caption[]{Selection Game
\label{figure:selection}}
\end{figure}
\end{pdfpic}

Again, the figure is isolated from the rest of the TeX file by
pdftricks commands.  Also, the regular LaTeX numbering of
figures is interfered with, so I adjust with a \setcounter
```

command. Finally, the \label and \reference commands do not work, so one must simply number the figures in the text.