Manual for `egame.sty`

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

Version 1, October 1997

1. Introduction

`egame.sty` is a LaTeX 2_ε style file for drawing extensive games. It uses LaTeX's picture environment, which entails limitations. I now use the style `egameps.sty` (available on my website), which uses the PSTricks package and contains many features unavailable in `egame.sty`.

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, 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.

To use the macros you need to include `\usepackage{egame}` in the preamble of your document. A game is begun by a call of the type

`\begin{egame}(400,500)`

which starts a LaTeX picture, with dimensions $(400, 500)$. (The default unit length, which can be changed by an optional argument in `\begin{egame}`, is 0.1mm.)

The next section gives some examples that illustrate many of the features of the package. Precise descriptions of the macros are given in Section 3.

2. 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}
```

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

```
\begin{figure}[htb]
\hspace*{\fill}
\begin{egame}(500,280)
%
% put the initial branch at (300,240), with (x,y) direction
```

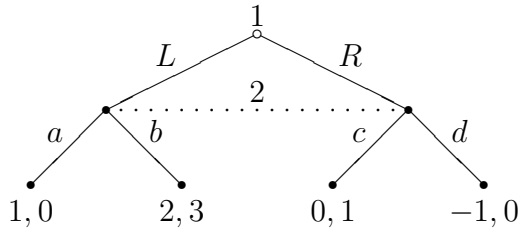


Figure 1. An extensive game

```
% (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, label it 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{2}[l]{$a$}{$b$}[$1,0$][$2,3$]
%
\end{egame}
\hspace*{\fill}
\caption[]{Another extensive game}\label{f:two}
\end{figure}
```

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
```

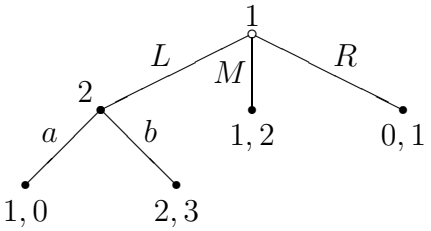


Figure 2. Another extensive game

```
\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}
%
% 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}
%
% 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}
%
% 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}
%
% 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}
```

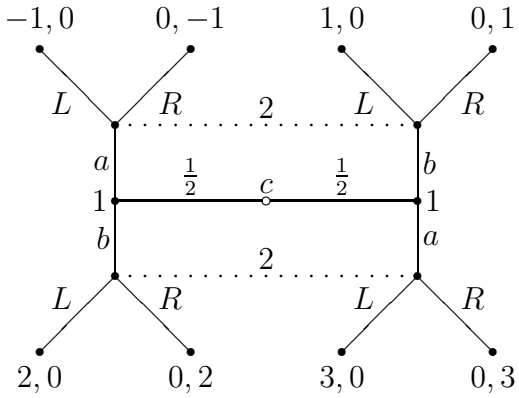


Figure 3. Yet another extensive game

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)
%
% 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{$1$}{$A$}{$D$}[][$0,1$]
%
% 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{}{$a$}{$b$}[$2,1$][$4,0$]
%
% 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{}{$a$}{$b$}[$1,3$][$-1,0$]
%
% 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{}{$a$}{$b$}[$-1,0$][$2,1$]
%
% 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}
```

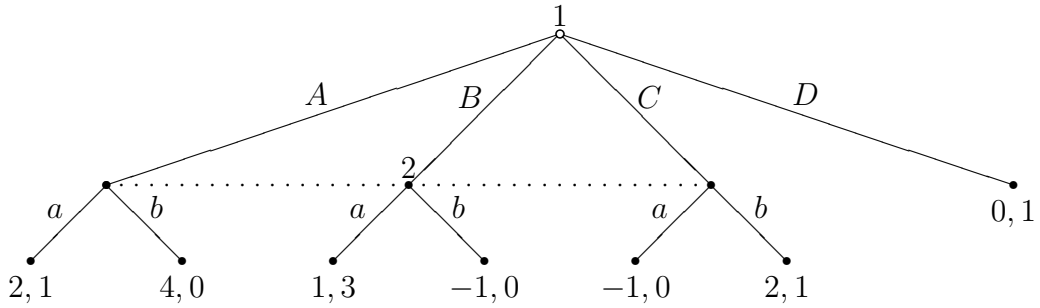


Figure 4. Yet another extensive game

3. 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 LATEX `picture` environment of width `w` and height `h`.)

`\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), or `\iiib` (three 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.

Permissible values:

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

(*h-incr*,*v-incr*) Each element must be an integer from −6 to 6; the pair must have no common divisor other than 1 and −1; both elements cannot be 0.

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

length Any positive integer.

`\ib{`*player-name*`}[`*player-label-position*`]{`*action-label*`}[`*payoffs*`]`
puts a single branch with the parameters of the preceding call to `\putbranch`, assigns it the player name *player-name*, positions the player label relative to the node according to *player-label-position*, and labels the action *action-label*; if the *payoffs* argument is present, the ending node is treated as terminal, and the payoffs *payoffs* are printed. 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. The default diameter of the initial node is 10 units. To change it, put `\initnodediam=n`, where `n` is the desired diameter, before the call to `\ib`. To change the appearance of the noninitial node, put `\renewcommand{\egnode}{`*code*`}`, where *code* defines the noninitial node. (For example, you might use `\makebox(0,0){\rule{0.5mm}{0.5mm}}` (the `\makebox` causes the rule to be positioned properly).)

Permissible values:

player-name Any character string.

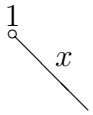
player-label-position If the direction of the branch (as specified either by `\egdirection` or by an optional argument of `\putbranch`) is `d` or `u`, either `l` (above/below and to the left) and `r` (above/below and to the right). If the direction of the branch is `r` or `l`, either `u` (to the left/right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

action-label Any character string.

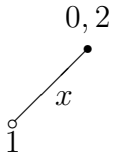
payoffs Any character string.

Examples:

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

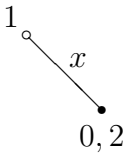



```
\initialfalse
\putbranch(0,40)(1,1)[u]{100}
\ib{1}{$x$}[$0,2$]
```



Notice that the optional direction specifier `u` in `\putbranch` has no effect on the direction of the branch (which is determined by the argument `(1,1)`, but affects the placement of the player label relative to the starting node. (I assume that most nodes with have two or more branches emanating from them, given by `\iib` or `\iiib`; in both of these cases the direction argument of `\putbranch` affects how the vector argument is interpreted—see below.)

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



`\iib{`*player-name*`}[`*player-label-position*`]{`*action-label1*`}{`*action-label2*`}[`*payoffs1*`][`*payoffs2*`]`
puts two branches with the parameters of the preceding call to `\putbranch`, assigns it the player name *player-name*, positions the player label relative to the node according to *player-label-position*, and labels the left/top action with *action-label1* and the right/bottom action with *action-label2*; if the *payoffs1* and *payoffs2* arguments are present, the ending node is treated as terminal, and the payoffs *payoffs1* and *payoffs2* are printed on the left/top and right/bottom nodes respectively. 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. There is one anomalous case: if the direction parameter in `\putbranch` is $(0, 0)$ then one branch is horizontal and the other is vertical; see the examples below.

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. The default diameter of the initial node is 10 units. To change it, put `\initnodediam=n`, where `n` is the desired diameter, before the call to `\ib`. To change the appearance of the noninitial node, put `\renewcommand{\egnode}{`*code*`}`, where *code* defines the noninitial node. (For example, you might use

```
\makebox(0,0){\rule{0.5mm}{0.5mm}}
```

(the `\makebox` causes the rule to be positioned properly).)

Permissible values:

player-name Any character string.

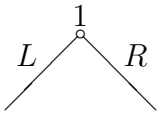
player-label-position If the direction of the branch (as specified either by `\egdirection` or by an optional argument of `\putbranch`) is `d` or `u`, either `l` (above/below and to the left) and `r` (above/below and to the right). If the direction of the branch is `r` or `l`, either `u` (to the left/right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

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

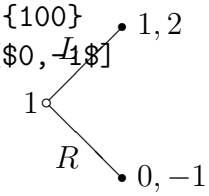
payoffs1 and *payoffs2* Any character strings.

Examples:

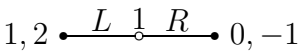
```
\putbranch(0,100)(1,1){100}
\iib{1}{$L$}{$R$}
```




```
\putbranch(0,100)(1,1)[r]{100}
\iib{1}{$L$}{$R$}[$1,2$][$0,-1$]
```

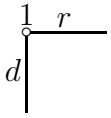


```
\putbranch(100,0)(1,0)[d]{100}
\iib{1}{$L$}{$R$}[$1,2$][$0,-1$]
```

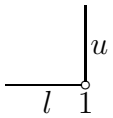


Anomalous cases (with direction parameter $(0, 0)$):

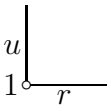
```
\putbranch(0,100)(0,0)[d]{100}
\iib{1}{$d$}{$r$}
```



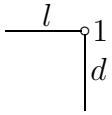

```
\putbranch(0,40)(0,0)[u]{100}
\iib{1}{$l$}{$u$}
```




```
\putbranch(0,40)(0,0)[r]{100}
\iib{1}{$u$}{$r$}
```




```
\putbranch(100,100)(0,0)[l]{100}
\iib{1}{$l$}{$d$}
```



`\iiib{`*player-name*`}[`*player-label-position*`]{`*action-label1*`}{`*action-label2*`}{`*action-label3*`}[`*payoffs1*`][`*payoffs2*`][`*payoffs3*`]`
puts three branches with the parameters of the preceding call to `\putbranch`, assigns it the player name *player-name*, 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*; if the *payoffs1*, *payoffs2*, and *payoffs3* arguments

are present, the ending node is treated as terminal, and the payoffs *payoffs1*, *payoffs2*,and *payoffs3* are printed on the left/top, middle, and right/bottom nodes respectively. 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 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. The default diameter of the initial node is 10 units. To change it, put `\initnodediam=n`, where `n` is the desired diameter, before the call to `\ib`. To change the appearance of the noninitial node, put `\renewcommand{\egnode}{`*code*`}`, where *code* defines the noninitial node. (For example, you might use

```
\makebox(0,0){\rule{0.5mm}{0.5mm}}
```

(the `\makebox` causes the rule to be positioned properly).)

Permissible values:

player-name Any character string.

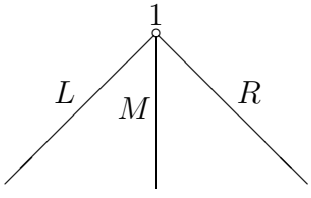
player-label-position If the direction of the branch (as specified either by `\egdirection` or by an optional argument of `\putbranch`) is `d` or `u`, either `l` (above/below and to the left) and `r` (above/below and to the right). If the direction of the branch is `r` or `l`, either `u` (to the left/right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

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

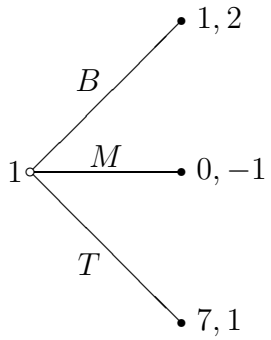
payoffs1, *payoffs1*, and *payoffs3* Any character strings.

Examples:

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



```
\putbranch(30,210)(1,1)[r]{200}
\iib{1}{$B$}{$M$}{$T$}[$1,2$][$0,-1$][$7,1$]
```



`\infoset(`*x-coord*`,`*y-coord*`)[`*direction*`]{`*length*`}{`*player-name*`}`
draws an information set starting at (*x-coord*,*y-coord*), 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 `\circle*{5}`; the spacing between the dots is set by `\infosetds`, the default value of which is 20. (Change by using `\renewcommand{\infosetds}{`*n*`}`, where *n* is the dot spacing you want.)

Permissible values:

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

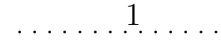
direction `h` (horizontal) or `v` (vertical).

length Any nonnegative integer.

player-name Any character string.

Example:

```
\egdirection{d}
\infoset(0,0){100}{1}
```



4. Parameters

`\egdirection`
Direction of game. Can be any `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.

`\initnodediam`
Diameter of initial node (which is restricted to be a circle). Can be any integer. Default: 10. Example: `\initnodediam=20`.

`\egnode`
Object used for nodes. Can be any object. Default: `\circle*{10}`. 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. Can be any object. Default: `\circle*{5}`. Example: `\renewcommand{\infosetdot}{\circle*{10}}`.

`\infosetds`
Spacing between dots in information set. Can be any integer. Default: 20. Example: `\renewcommand{\infodotds}{40}`.

`\egplayerlabelsep`
Spacing between player label and center of initial node and between player label and center of information set. Can be any dimension. Default: 1mm. Example: `\egplayerlabelsep=2mm`.

`\egactionlabelsep`
Spacing used to position action labels relative to branches. Can be any dimension. Default: 0.7mm. Example: `\egactionlabelsep=1mm`.

`\egpayoffsep`
Spacing between payoffs and center of terminal nodes. Can be any dimension. Default: 2mm. Example: `\egpayoffsep=1mm`.

5. 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.

You will probably need to fine-tune the placement of the action labels in some games. You can do so by adding space to the text of the labels.

6. Enhancements

It would be nice to have a graphical interface, like that in LATEXCAD. I doubt that I will write one.

It would be nice also if the macro 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. Consider a positively- and finitely-sloped branch for a downward-pointing game. Currently a label is attached to such a branch by surrounding the label with a border of width `\egactionlabelsep`, and then putting the bottom right-hand corner of the resulting box at the midpoint of the branch. If the slope is infinite then the middle of the right-hand side of the box is put at the midpoint of the branch, and if the slope is zero then the middle of the bottom side of the box is put at the midpoint of the branch. It would nice to have a continuous change in the position as the slope goes from 0 to infinity, but I have not yet developed a satisfactory algorithm. The current algorithm works best for slopes of 0, 1, and infinity; it is not great for short branches with high (but not infinite) or low (but not zero) slopes.