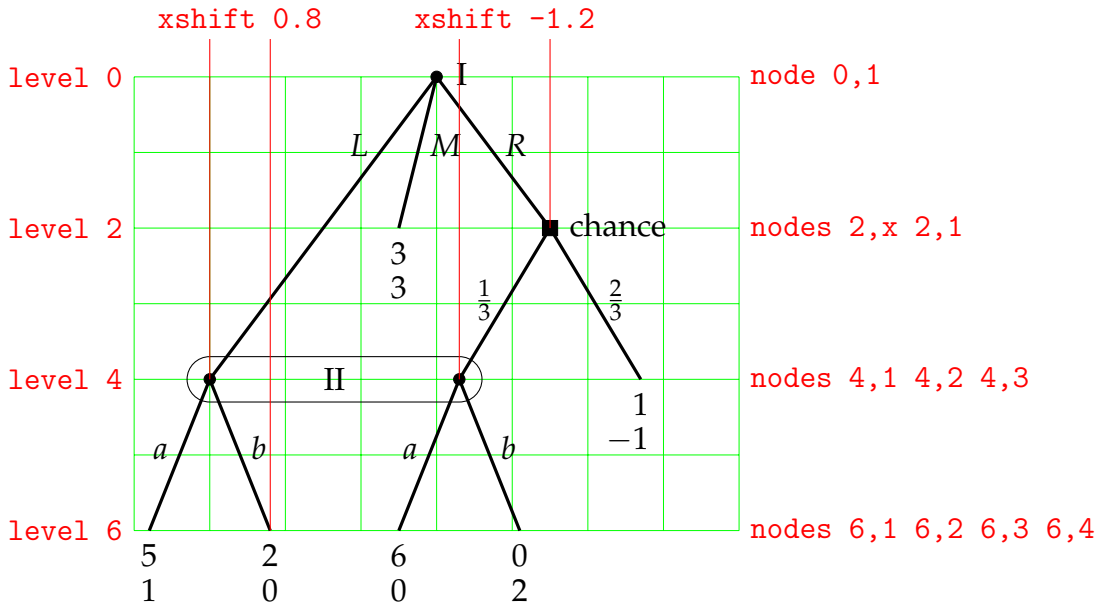
Documentation of drawtree.py



This extensive game, apart from the red annotations, has been generated from the following input in the file with the (default) name `example.ef` :

```
player 1 name I
player 2 name II
level 0 node 1 player 1
level 2 node 1 player 0 xshift a=1.5 from 0,1 move R
level 4 node 1 xshift -2a from 0,1 move::0.25 L
level 2 node x xshift -.5 from 0,1 move:r M payoffs 3 3
level 4 node 2 xshift -b=1.2 from 2,1 move \frac{1}{3}
level 4 node 3 xshift b from 2,1 move \frac{2}{3} payoffs 1 -1
level 6 node 1 xshift -c=.8 from 4,1 move a payoffs 5 1
level 6 node 2 xshift c from 4,1 move b payoffs 2 0
level 6 node 3 xshift -c from 4,2 move a payoffs 6 0
level 6 node 4 xshift c from 4,2 move b payoffs 0 2
iset 4,1 4,2 player 2
```

The separate lines in this file start with one of the keywords `player`, `level`, or `iset`, followed by additional information as in the example, separated by whitespace (at least one blank or tab; above only single blanks have been used).

`player` sets the player name (defaults if omitted: player number, "chance" for player 0). If the chance player is meant to have no name at all, write `player 0 name ~` .

`level` encodes the game tree nodes. Start with level 0 for the root. Each level should be the standard level distance of 2 centimetres, i.e. even numbers. Adhere to this standard, even for large trees; the actual distance can be changed globally. However, levels can be intermediate numbers such as odd integers or floating point numbers. Any floating point number will be rounded to three digits after the decimal point and used

as such. That is, `level 0.333333333` will automatically be used like `level 0.333`. It is better to use integers, or to round levels to multiples of 0.1 to keep the level identifier small.

Each node gets an identifier, typically just numbered per level (see annotation on the right above). The combination **level,nodeid** (no blanks) identifies the node, e.g. `0,1` for the root node.

For each node, omit `player` if it is a terminal node with payoffs, or if the player will be specified with the information set via `iset` as below. A chance node needs `player 0` to get a square node symbol.

The parent of a node is specified with `from`, e.g. `from 0,1` if the parent is the root node. The parent node must already have been defined.

`xshift` specifies the horizontal offset in centimetres **relative to the parent node**. For example, `xshift -1.2` means 1.2 centimetres to the left of the parent node. One can assign **variables** for later use, e.g. `xshift -b=1.2` means 1.2 centimetres to the left, and 1.2 is stored in `b` which is then used again. In the above code, the variable `c` is defined once and used three more times. Variables can be any identifiers that do not start with a digit which would otherwise be used as a multiplier as in `xshift -2a` in the example above (the use of this multiplier automatically places correctly the move label *L* even if just writing `move L` instead of `move::0.25 L`).

The move label is specified with `move`, with the options `move:l` or `move:r` to force the move label to the left or right of the line that connects to the parent. Otherwise, the default is to put the label on the right for a line that goes down to the right from the parent, and left otherwise. The further specification `move::0.25` specifies that the move label is located at 0.25 of the way from the parent to the child, where 0.25 can be any number in $[0,1]$, with default 0.5. For a chance node as parent, use `\frac{1}{3}` as a move label to specify move probabililty $\frac{1}{3}$. **No blanks are allowed in move names, player names, or payoffs**. Use `~` (non-breakable space in LaTeX) instead.

`payoffs` come last, separated by blanks. As many as there are players (maximally 4).

The keyword `iset` followed by a list of nodes defines an information set. Add `player` with the player number at the end to specify the player. The player name will be placed in between the two middle nodes for an even number of nodes, or before the middle node for an odd number of nodes. If you want it somewhere else, write it between the respective nodes in the list.

You can use comment lines by prefixing them with `%` .