`istgame.sty`
Draw Game Trees with Ti*k*Z

In-Sung Cho
`ischo <at> ktug.org`

Economics, Kongju National University

2019/01/27   version 2.0

Abstract

This is a LaTeX package that provides macros based on Ti*k*Z to draw a game tree. The main idea underlying the core macros here is the completion of a whole tree by using a sequence of simple 'parent-child' tree structures, with no longer nested relations involved like the use of grandchildren or great-grandchildren. With the istgame package, you can draw a game tree as easily as drawing a game tree with pen and paper.

KEYWORDS: game trees, nodes, branches, information sets, continuum of branches, subgames

Table of Contents

0 Changes and remarks

0.1 Changes

Some macros have been CHANGED and REMOVED. Those who have used these changed and removed macros may want to FIND and REPLACE the followings:

ver. 1.0	ver. 2.0 or later
`\istb.`	`\istbt`
`\xtInfoset'`	`\xtInfoset`
`\xtInfoset0'`	`\xtInfoset0`
`\setistgrowkey`	`\setxtgrowkey`

Changed and removed macros

- The macro name `\istb.`(terminal version) has been changed to `\istbt` (terminal version).
  - This is the opportunity cost of having a new macro `\istB`.
- The two (unsatisfactory) macros `\xtInfoset'` and `\xtInfoset0'` have been removed.
  - The macro `\xtInfoset0` is completely redesigned, so that we do not need the macro `\xtInfoset0'` any more.
  - No reasons could be found to keep (even for the backward compatibility) the swap versions `\xtInfoset'` and `\xtInfoset0'`, except for the inconvenience to do 'find and replace.'
- To keep consistency in naming macros, `\setistgrowkey` is renamed as `\setxtgrowkey`.

Redesigned macros and the environment

- `\xtInfoset0`: completely redefined to improve its function
  - Now a *sloped* information set is possible.
  - It connects two nodes like `\xtInfoset0(coor1)(coor2)`, but if the two coordinates are identical it represents a *singleton information set* by a *circle* by default.
  - This change does not seem to cause much harm, but be aware that the swap version `\xtInfoset0'` has been removed and replaced by `\xtInfoset0`.
  - Be aware also that the way to change the *height* (`1em` by new default) of an information set has been *changed*, though you might not see much difference if you have only used the default information sets.
  - With *new macros* `\xtCInfoset` and `\xtCInfoset0`, a curved (even skewed curved) information set is now possible.
- istgame environment: (internal change)
  - Now the each value of the option of `xscale` and `yscale`, if exists, is extracted and saved at `\xtxscale` and `\xtyscale`, respectively. The value of `scale` is also saved at `\xtscale` only when it is used without `xscale` nor `yscale`. These values are internally used to get the best outputs of trees in many ways.
  - If the Ti*k*Z arrow option `->` exists in the option list of an istgame environment, you can globally control the arrow-end-shorten value (by default, `shorten >=0pt`) by using a new macro `\setistgameshorten`. This is to get a better result of branches with arrows.
- Some changes that you might not notice have been made, including:
  - The core macros `\istroot`, `\istb`, and `\endist` have literally been redefined for some purposes, but this makes no difference to users.
  - The options `thin` and `solid` have been added to the definitions of basic node styles.
  - Some default values have been slightly changed.

0.2 What's new

Some new functions

- *input mode changer* (math or text) for important labels: owners, action labels, and payoffs
    - `\setistmathTF` as the input mode changer for the important labels
    - `\setistmathTF*` having additional function as the *text font style changer*
- *curved* (even *skewed*) *information sets*
    - `\xtCInfoset` for curved information sets
    - `\xtCInfosetO` for curved bubble type information sets
- enhanced *continuum of branches* (making `\istcntm` and `\istcntmarc` obsolete)
    - `\istrootcntm` for a continuum triangle
    - `\istrootcntmA` for a continuum arc (with `\istbA`)
    - `\cntmAInfoset` and `\cmtmAInfosetO` for information sets for a continuum of branches
- *arrows* and *middle arrows* on branches
    - controllable arrow option `->-` with `\setxtarrowtips` and middle arrow tip styles
    - `\xtShowMidArrows` and `\xtShowArrows`
- and some more
    - `\istB` for dual action labels
    - `\xtTimeLineH`, `\xtTimeLineV`, `\xtCommentTo`, `\xtCommentFrom`, etc.

List of new macros

- `\istbt(*)`: terminal version of `\istb(*)` (replacement of the removed macro `\istb.(*)`)
- `\istB(*)`: dual action label version of `\istb(*)`
- `\istBt(*)`: terminal version of `\istB(*)`
- `\istbA(*)`: alternative version of `\istb(*)` (intended to work with a continuum arc)
- `\istbAt(*)`: terminal version of `\istbA(*)`

- `\setistmathTF`: input mode changer (math or text) for owners, action labels, and payoffs
- `\setistmathTF*`: input mode and font style changer for owners, action labels, and payoffs

- `\xtCInfoset`: (curved version) curved information set
- `\xtCInfosetO`: (curved oval version) curved oval type information set
- `\xtCInfosetOTurnX`: turns `X` circles of `\xtCInfosetO` (to use it just in case)

- `\xtInfoset*`: prints owners according to the input mode as set by `\setistmathTF(*)`
- `\xtInfosetO*`: prints owners according to the input mode as set by `\setistmathTF(*)`
- `\xtCInfoset*`: prints owners according to the input mode as set by `\setistmathTF(*)`
- `\xtCInfosetO*`: prints owners according to the input mode as set by `\setistmathTF(*)`
- `\xtOwner*`: prints owners according to the input mode as set by `\setistmathTF(*)`
- `\xtActionLabel*`: prints action labels according to the input mode set by `\setistmathTF(*)`
- `\xtPayoff*`: prints payoffs according to the input mode as set by `\setistmathTF(*)`
- `\xtInfosetOwner*`: prints owners according to the input mode set by `\setistmathTF(*)`

- `\setxtinfosetstyle`: changes line the style (line style, fill, etc.) for all information sets
- `\setxtinfosetlayer`: changes the layer of information sets
- `\setxtsubgamelayer`: changes the layer of `\xtSubgameBox` and `\xtSubgameOval`
- `\setistgameshorten`: value of the key `shorten >` in `istgame` environment option list

- `\cntmdistance`: analogous to `\xtdistance`, for a continuum of branches
- `\cntmdistance*`: incorporates `\cntmdistance` with `\xtdistance`
- `\istrootcntm`: `\istroot + cntm`, printing a continuum of branches
- `\istrootcntm'`: swap version of `\istrootcntm`
- `\istrootocntm`: (oval version) `\istrooto + cntm`, printing a continuum triangle
- `\istrootocntm'`: swap version of `\istrootocntm`
- `\cntmpreset`: controls a continuum of branches (line style, color, size, fill color)
- `\cntmpreset*`: controls a simple triangle continuum of branches with no background color

- `\cntmistb`: similar to `\istb` for the outermost branches of a continuum triangle
- `\cntmistb*`: draws `solid node`s at the ends of the continuum outermost branches
- `\istrootcntmA`: `\istroot + cntmA`, printing a continuum arc
- `\istrootcntmA'`: swap version of `\istrootcntmA`
- `\istrootocntmA`: (oval version) `\istrooto + cntmA`, printing a continuum arc
- `\istrootocntmA'`: swap version of `\istrootocntmA`
- `\cntmApreset`: controls the features of a continuum arc
- `\cntmAlayerpreset`: sets the layer of a continuum wedge (with fill color)
- `\cntmAistb`: similar to `\istb` for the outermost branches of a continuum arc
- `\cntmAistb*`: draws `solid node`s at the ends of the continuum arc outermost branches
- `\cntmAInfoset`: prints an information set for a continuum arc
- `\cntmAInfosetO`: oval version of `\cntmAInfoset`
- `\xtShowEndPoints*`: shows additionally the outermost endpoints of a continuum of branches
- `\xtHideEnpPoints*`: turns off only the endpoints of a continuum of branches
- `\cntmAexpostShowEndPoints`: shows the two endpoints of a continuum arc

- `\setxtarrowtips`: works through `->-` to control the features of middle arrow tips
- `\xtShowMidArrows`: shows arrows in the middle of branches
- `\xtHideMidArrows`: hides middle arrows drawn by `\xtShowMidArrows`
- `\setxtshowmidarrows`: controls the middle arrows on branches in a simple tree
- `\xtShowArrows`: shows arrows at the ends of branches with endpoints
- `\xtHideArrows`: hides arrows drawn by `\xtShowArrows`
- `\xtHideArrows*`: hides arrows but with endpoints remained
- `\setxtshowarrows`: controls the features of the arrows shown by `\xtShowArrows`

- `\xtTimeLineH`: a horizontal time-line
- `\xtTimeLineH'`: a horizontal time-line with a label at the other end
- `\xtTimeLineV`: a vertical time-line
- `\xtTimeLineV'`: a vertical time-line with a label at the other end
- `\xtCommentTo`: to leave a comment `to` a node `from` a *relative* coordinate
- `\xtCommentFrom`: to leave a comment `from` an *absolute* coordinate `to` a node

- `\xtcureslopedlabelsNS`: cures the Ti*k*Z issue of sloped labels with asymmetric scales
- `\xtcureslopedlabelsEW`: same, for a tree growing eastwards or westwards
  The last two macros are for only temporary use and could be removed at any time.

List of new arrow styles

- `->-`: (controllable) middle arrow tip, taking one optional argument
- `->>-`: double middle arrow tip
- `->>>-`: triple middle arrow tip
- `-o-`: circle middle arrow tip
- `-x-`: cross middle arrow tip

0.3 How to read this document

As a Ti*k*Z user, if this is your first read of this manual, *all you need to read* is Section 1 entitled "*Getting started.*" That's only *four* pages long. (You can find a little secret at the first part of Section 1.3 entitled "*Complete examples for desperate users.*") Every example in this document is provided with its complete codes you can copy to use. Now you can go to Section 1 to get started!

If you are not urgent, you can continue to read sections on *core macros* up to Section 5. That's about *twenty* pages total. Still more time? Then read Section 6 (entitiled "*Important labels: players, action labels, and payoffs*") and Section 7 (entitled "*Input mode and text font style changer*"). That's about *thirty* pages all total.

Throughout the manual, just disregard the parts with marks including "*fine-tuning*" or "*not for most users*," if you are not an experienced user of this package. Just reading first part of each section will suffice most users to draw game trees in almost all cases, hopefully.

If you you are an experienced user of the istgame package, enjoy every detail of the package.

0.4 Remarks

Rules of thumb for the usage of delimiters Though the rules are not strictly observed, it might be useful to go over the rules of thumb for the usage of delimiters.

- `{ }`: contents, texts, important dimensions (mandatory or optional)
- `[ ]`: usual options, positions or directions, node types, fill color
- `( )`: coordinate related arguments, dimension as the last optional argument, special purposes
- `< >`: angles (or directions), special purposes (mostly used right before `{ }`)
- `+ .. +`: only for the local change of level and sibling distances with `\istroot` and its friends
- `! !`: `midpoint factor` only for curved information sets

As for the order of delimiters, the (somewhat loose) rules are as follows:

- Generally, all macros are designed to avoid leaving an empty argument like `[][blue]`, as possible as they can do. (This is *input minimalism* for lazy users including myself!)
- `< >` is used just before `{ }`, except for `\istb` and its friends.
- `( )` as the last optional argument is for dimensions, like `(1em)` or `(3pt)`.
- The order mostly looks like `<>{}[]` or `<>{}[]()`, especially after a mandatory argument.

Optional versions of macros Some macros have a starred(`*`) version or a swap(`'`) version, for example, `\istb*` or `\istroot'`.

- `* version`: just another version with different functions (in some cases, it's quite different)
- `' version`: clockwise arrangement of branches or its related version

Global macros Some macros have global effects, so you can use them outside of the istgame environment or even in the preamble of your document, but *with very great caution.*

The following macros, all prefixed by `\setist...`, can be used outside of the istgame environment, to change the default values of the options:

- `\setistgamefontsize{<text size>}` (default: `\normalsize`)
- `\setistgameshorten{<arrow end shorten dim>}` (default: `0pt`)
  - This works only when `->` exists in the option list of the istgame environment.

Though it is not recommended, you can use all the macros prefixed by `\setist...` inside or outside of the istgame environment to change the default values.

- `\setistmathTF`, `\setistmathTF*` (initially: `011`)
- `\setistdefaultnodeinnersep` (default: `1pt`)
- `\setistdefaultnodeoutersep` (default: `0pt`)
- `\setistdefaultnodedrawcolor` (default: `black`)
- `\setistdefaultnodefillcolor` (default: `white`)
- `\setist<...>NodeStyle`
- `\setistgrowdirection`, `\setistgrowdirection'` (default: `south`):
  - You may not want to use (but still can use) this outside of an istgame environment.

You can even do, in the preamble of your document, like:

```
\usepackage{istgame}
  \setistgamefontsize{\normalsize}
  \setistdefaultnodedrawcolor{black}
  \setistSolidNodeStyle{2.4pt}
  \setistgameshorten{.3pt}
  \setistmathTF011
```

The node styles such as `ellipse node` and `rectangle node` has `white` as the default background color. This means that if the paper color of your document is not white, say, `blue!16`, you might not satisfy the results because the nodes have `white` background. In this case you can resolve the conflict by adding the following two lines at the very first of your document:

```
\papercolor{blue!16}
\setistdefaultnodefillcolor{blue!16}
```

The background color of the TeXShop preview in the dark mode of `macOS Mojave` is very similar to `black!16`.

Known problem with the tikz-qtree package It seems that tikz-qtree changes node anchors. So, with the tikz-qtree package uploaded, you will get unexpected results when you draw a game tree by using the `tree` library in Ti*k*Z. Since istgame is based on the `tree` library, it is also affected by tikz-qtree, resulting in unexpected outputs.

The best way to resolve this problem is that you DO NOT LOAD tikz-qtree when you draw game trees with Ti*k*Z. If, for some reason, you need to load tikz-qtree when you draw a game tree by using the istgame package, a temporary solution to resolve the conflict is to add the Ti*k*Z option `edge from parent path` in the option list of istgame environment as follows:

```
% tikz-qtree conflict resolution (only with \usepackage{tikz-qtree})
[
  edge from parent path={(\tikzparentnode) -- (\tikzchildnode)}
]
```

0.5 Previous changes (up to version 1.0)

A considerable number of macro names have been changed in the version 0.8 (Jan. 17, 2017) of this package.[1] The following old macro names in any previously written documents using codes in istgame ver. 0.7 or before, should be replaced by the new names, accordingly.

Also, `\istroot*`, `\istcntm*`, and `\xtInfoset*` should be replaced by `\istrooto`, `\istcntmarc`, and `\xtInfoseto`, respectively, in the version 1.0 or later.

ver. 0.7 or before	**ver. 0.8 or later**	**ver. 1.0 or later**
`\xdistance`	`\xtdistance`	
`\xDot`	`\xtNode`	
`\xInfoset`	`\xtInfoset`	
`\xInfoset*`	`\xtInfoset*`	`\xtInfosetO`
`\xInfosetOwner`	`\xtInfosetOwner`	
`\xActionLabel`	`\xtActionLabel`	
`\xPayoff`	`\xtPayoff`	
`\ShowTerminalNodes`	`\xtShowTerminalNodes`	
`\HideTerminalNodes`	`\xtHideTerminalNodes`	
`\levdist`	`\xtlevdist`	
`\sibdist`	`\xtsibdist`	
`\setistactionlabelshift`		`\xtALPush`
`\setistactionlabelposition`		`\xtALShift`
`\istroot*`		`\istrooto`
`\istcntm*`		`\istcntmarc`

---

[1] The istgame package of the version older than ver. 1.0 had been distributed via the KTUG (Korean TeX Users Group) Private Repository.

1 Getting started

The package istgame provides macros built on Ti*k*Z to draw game trees. The core macros provided with this package are `\istroot`, `\istb`, and `\endist`. `\istroot` pins down the root of a tree or a subtree, `\istb` represents a branch, and `\endist` indicates the end of drawing a simple tree. Without `\endist`, the tree is NOT actually drawn, with no error messages produced. A tree drawn by the sequence of `\istroot-\istb-\endist` is a `simple tree`. You can draw a whole game tree by repeatedly connecting these *simple tree structures*.

Here, the prefix 'ist' stands for 'it's a simple tree.' You can also read it as 'insung's simple tree' if you would like.

The package istgame depends on the packages tikz, xparse, and expl3.
To use the istgame package you must load the package in the preamble of your document:

```
\usepackage{istgame}
```

The package uses the following Ti*k*Z libraries:

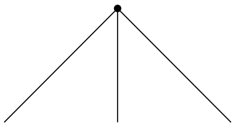
```
trees,calc,arrows,shapes,positioning,backgrounds,fit,decorations.markings,
```

and also loads `patterns` and `intersections` for additional use.

1.1 Getting-started example: a simple tree

Let us get started with a simple self-explanatory example:

```
%% \usepackage{istgame}
% Example: a simple tree
\begin{istgame}
\istroot(0)(0,0) % names the root as (0) at (0,0)
  \istb % endpoint will be (0-1), automatically
  \istb % endpoint will be (0-2), automatically
  \istb % endpoint will be (0-3), automatically
  \endist % end of simple (parent-child) structure
\end{istgame}
```



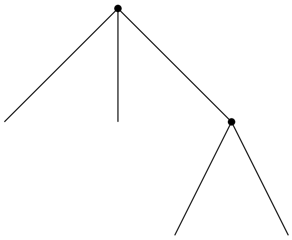
The resulting tree has the *height* of `15mm` and the *distance between two neighbor endpoints* (not shown) is also `15mm` by default. In Ti*k*Z, the height is called the level distance and the distance between two neighbor endpoints is called the sibling distance.

If the second parenthesis argument of `\istroot` is omitted, it is regarded as `(0,0)` by default, otherwise it is necessary to specify the coordinate from which a simple tree starts.

1.2 Connecting simple tree structures

Basically, in order to draw a whole game tree, we just repeat the simple `\istroot-\istb-\endist` structure.

```
% Example: connecting simple trees
\begin{istgame}
\istroot(0) % names the root (0) at (0,0)
  \istb % endpoint will be (0-1), automatically
  \istb % endpoint will be (0-2)
  \istb % endpoint will be (0-3)
  \endist % end of simple (parent-child) structure
\istroot(c)(0-3) % names the subroot (c) at (0-3)
  \istb % endpoint will be (c-1)
  \istb % endpoint will be (c-2)
  \endist
\end{istgame}
```



In the above example, the simple *subtree* is rooted at `(0-3)`, names the subroot `(c)`, and has two branches whose endpoints are automatically named `(c-1)` and `(c-2)`, respectively.

Note that the user-defined names of the (sub)roots and the names of endpoints are arranged `counterclockwise` (from left to right) by Ti*k*Z at the endpoints of branches, which can be used as coordinates in the usual Ti*k*Z way.

1.3 Complete examples for desperate users

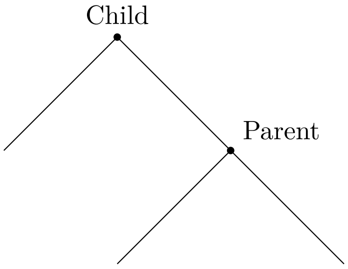
Basically, `\istroot` designates a *decision node* and its *owner* (or a *player*), `\istb` prints a *branch* coming out from the decision node with *action labels* and *payoffs*, and `\endist` actually draws the tree structures. (One secret is that I almost always use only the basic features of the istgame package, discussed in this subsection.)

1.3.1 How to put a decision node and its owner

```
\istroot(<decision node name>)(<root location>)<owner position>{<owner>}
```

The only *mandatory* argument of `\istroot` is `(<decision node name>)` and all others are *optional*. If the `(<location>)` where the *root* or a *decision node* is placed is omitted, it is regarded as `(0,0)` by default. The position of an owner (or a player) is `<above>` (or equivalently, `<90>` degree) by default. The *owner* of a node is printed in *text mode* by default.

```
% Example: first try
\begin{istgame}
\xtdistance{15mm}{30mm}
\istroot(0)(0,0){Child}
  \istb%{Good}[above left]{(0,2)}
  \istb%{Bad}[above right]
  \endist
\istroot(1)(0-2)<30>{Parent}
  \istb%{Forgive}[above left]{(1,1)}
  \istb%{Punish}[above right]{(-1,-1)}
  \endist
\end{istgame}
```



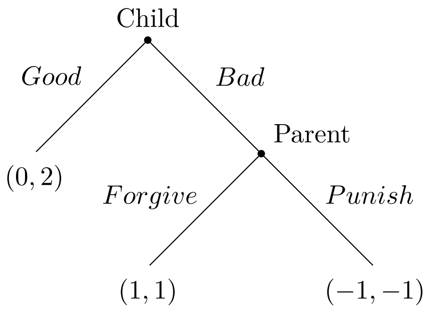
In fact, `\istroot` and its variants have much more functions than these. Later, you can look into Section 4 on page 14 for more details.

1.3.2 How to print branches with action labels and payoffs

```
\istb{<action label>}[<action label pos>]{<payoffs>}[<payoff pos>]
```

With the macro `\istb`, you can draw a *branch* and put an *action* label and *payoffs* as optional arguments.

```
% Example: first try
\begin{istgame}
\xtdistance{15mm}{30mm}
\istroot(0){Child}
  \istb{Good}[above left]{(0,2)}
  \istb{Bad}[above right]
  \endist
\istroot(1)(0-2)<30>{Parent}
  \istb{Forgive}[al]{(1,1)}
  \istb{Punish}[ar]{(-1,-1)}
  \endist
\end{istgame}
```



The positions of an action label and payoffs are specified as options right after each of the two. When you omit the position of payoffs, the package prints them naturally, but still you can change the location using an option like `[left]` or `[above left]`. For the positions of action labels and payoffs, you can use an abbreviation `[al]` for `[above left]` and similarly `[ar]`, `[bl]`, and `[br]`. The abbreviations `[a]`, `[b]`, `[l]`, and `[r]` are also available.

Both of the *action labels* and *payoffs* are printed in *math mode* by defualt. (You can change the input mode for owners, action labels, and payoffs using a very useful macro `\setistmathTF`, documented in Section 7 on page 36.)

In fact, `\istb` and its variants have much more functions than these. For more details, see Section 5, on page 18. If this is your first read of this manual, however, you don't need to bother about all the details at the moment.

Following three sections are about an information set, a continuum of branches, and changing the direction of tree growing. If you do not need to use them now, the core macros `\istroot`, `\istb`, and `\endist` are *all you need to know about* drawing game trees of any size, small or big. Just connect simple trees to complete a whole tree.

1.3.3 How to put information sets

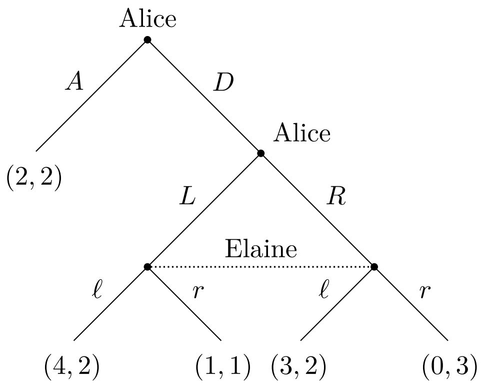
The macro `\xtInfoset` connects two nodes with a densely dotted line, by default, representing an information set.

```
\xtInfoset(<from coor>)(<end coor>){<owner>}[<owner pos>]
```

The two node coordinates are *mandatory*. You can put the owner of an information set as an optional argument. An owner is printed above (by default) the line in *text mode*, which can be changed. For more details about `\xtInfoset`, see Section 10 on page 45.

In the example below, the macro `\xtInfoset` is used to show an information set.

```
% Example: information set
\begin{istgame}
%\setistgrowdirection'{east}
\xtdistance{15mm}{30mm}
\istroot(0){Alice}
  \istb{A}[al]{(2,2)}
  \istb{D}[ar]
  \endist
\istroot(1)(0-2)<above right>{Alice}
  \istb{L}[al]
  \istb{R}[ar]
  \endist
\xtdistance{10mm}{20mm}
\istroot(2)(1-1)
  \istb{\ell}[al]{(4,2)}
  \istb{r}[ar]{(1,1)}
  \endist
\istroot(3)(1-2)
  \istb{\ell}[al]{(3,2)}
  \istb{r}[ar]{(0,3)}
  \endist
\xtInfoset(2)(3){Elaine}
\end{istgame}
```



The package also provides the macro `\xtInfosetO` to draw a bubble type information set. Just replace `\xtInfoset` with `\xtInfosetO` (see Section 10.2.1 on page 46).

You can also draw a curved information by using `\xtCInfoset` (Section 10.3) and even a curved bubble type by using `\xtCInfosetO` (Section 10.4). You can try now.

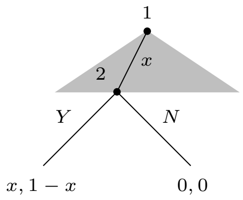
1.3.4 How to put a continuum of branches

Just use `\istrootcntm` (`\istroot + cntm`), instead of `\istroot`, to draw a continuum of branches.

```
\istrootcntm(<decision node name>)(<root location>)<owner position>{<owner>}
```

The macro `\istrootcntm` works just like `\istroot`, but it prints a background triangle in `black!25`, by default, representing a continuum of branches.

```
% Example: continuum of branches
\begin{istgame}[font=\scriptsize]
\istrootcntm(0){1}
  \istb{x}[r] \istbm \endist
\xtdistance{10mm}{20mm}
\istroot(1)(0-1)<120>{2}
  \istb{Y}[al]{x,1-x} \istb{N}[ar]{0,0} \endist
\end{istgame}
```



Here, `\istbm` represents a missing (or an invisible) branch (see page 19).

You can also change the color and size of the triangle representing a continuum of branches. For more details on `\istrootcntm` see Section 11.1, on page 56.

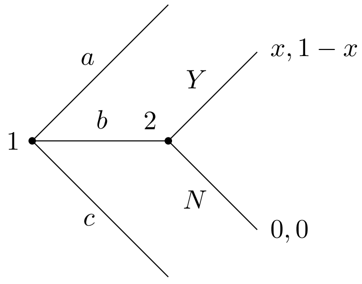
The package also provides the macro `\istrootcntmA` to draw an arc to represent a continuum of branches. You can just replace `\istrootcntm` by `\istrootcntmA` to do that, but let us not try this now. (If you really want to try this now, you should change the first `\istb` to `\istbA` and then take out `\istbm` in the above example. It is said not to try this now.) For more details on `\istrootcntmA` see Section 11.2, on page 61.

You can use every options and macros you can use for the `tikzpicture` environment with the `istgame` environment. In the above example, `font=\scriptsize` is used as an option.

1.3.5 How to change the growing direction of a tree

With `\setistgrowdirection` or `\setistgrowdirection'`, you can easily change the direction (`south` by default) to which a game tree grows, as shown in the example below.

```
% Example: \setistgrowdirection(')
\begin{istgame}[scale=1.2]
\setistgrowdirection'{east}
\istroot(0)<180>{1}
  \istb{a}[al]
  \istb{b}[a]
  \istb{c}[bl]
  \endist
\xtdistance{10mm}{20mm}
\istroot(1)(0-2)<135>{2}
  \istb{Y}[al]{x,1-x}
  \istb{N}[bl]{0,0}
  \endist
\end{istgame}
```



All you need to do is just to specify the direction you want, like `\setistgrowdirection{west}`, `\setistgrowdirection'{east}`, or `\setistgrowdirection'{-45}`. The prime version (`'`) is just to arrange branches `clockwise` (by default `coutnerclockwise`). In the above example, if you use `\setistgrowdirection` without the prime, the branches will be arranged `counterclockwise`, like a, b, and c *from bottom to top*. When changing the direction of a tree, you may want to relocate the owner and action labels. For more details, see Sections 9 (on page 41) and 6.3.2 (on page 33).

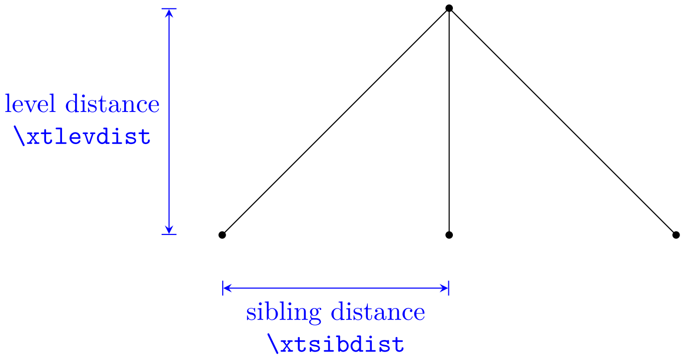
Now you are ready to draw any standard game trees such as all game trees in Osborne's book.

2 Important distances: `\xtdistance`

The length and direction of branches in a simple tree can be controlled by the macro `\xtdistance`. Here, the prefix `xt` stands for `extensive tree`.

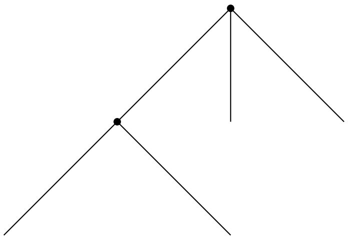
```
% syntax: \xtdistance
  \xtdistance[<level depth>]{<level dist>}{<sibling dist>}
% defaults:
  [1]{15mm}{15mm}
```

The macro `\xtdistance` sets or resets the `level distance` and the `sibling distance`, respectively. Note also that internally, for example, `\xtdistance{20mm}{30mm}` assigns `20mm` to `\xtlevdist` and `30mm` to `\xtsibdist`, which renew the default distances. It is effective until you change the distances by using another `\xtdistance`.



You can use `\xtdistance` at any time you want to change the length and the directions of branches. Since we are dealing with simple *parent-child* tree structures, `<level depth>` is `1` by default. (The level depth number other than `1` is not expected to be used.)

```
% Example: \xtdistance
\begin{istgame}
\xtdistance{15mm} % default
\istroot(0) % names the root (0) at (0,0)
  \istb % endpoint will be (0-1), automatically
  \istb % endpoint will be (0-2)
  \istb % endpoint will be (0-3)
  \endist % end of simple (parent-child) structure
\xtdistance{15mm}{30mm} % changes sibling dist
\istroot(a)(0-1) % names the subroot (a) at (0-1)
  \istb % endpoint will be (a-1)
  \istb % endpoint will be (a-2)
  \endist
\end{istgame}
```



Remark:

- Since `\xtlevdist` and `\xtsibdist` are assigned values by `\xtdistance`, you can use these values to do some calculation, for example, like `1.2*\xtlevdis` or `1.5*\xtsibdist`.
- The starred version `\cntmdistance*` is provided to deal with a continuum of branches and `\xtdistances` together, which is not documented at the moment. See Section 11.1.3, on page 58, for more details about `\cntmdistance*`.

In fact, the core macros are much more powerful. `\istroot` controls the direction to which a simple parent-child tree grows, node styles, the node owner and its position, the height and sibling distance of a current simple tree, etc. `\istb` specifies the growing direction of an individual branch, branch line styles, branch color, action labels, and payoffs and their position. Below we will see in more details on how the core macros and others work.

3 The istgame environment and node styles

3.1 The istgame environment

The package provides the istgame environment, which is basically the sum of the tikzpicture environment plus some additional functions and different initial values. So it accepts all the options and macros that can be used for the tikzpicture environment. (Note that most of the macros provided by the package work also in the tikzpicture environment, but some works only in the istgame environment.)

```
% istgame environment
% \def\istgame@default@fontsize{\normalsize}
\begin{istgame}[ <Tikz  options> ]
  < istgame contents >
  < tikzpicture contents >
\end{istgame}
% default options:
[ font=\normalsize , >=stealth ]
```

The default font size is set as `font=\normalsize`. You can globally change the default font size by using `\setistgamefontsize`, like `\setistgamefontsize{\scriptsize}`. Since the environment istgame is basically the same as tikzpicture, you can also locally change the font size by using the `font` option key, like `\begin{istgame}[font=\scriptsize]...\end{istgame}`.

Remark: (Not for most users) What the istgame environment *internally* does includes:

- In the newly designed istgame environment, it internally checks and extracts the optional values of `xscale` and `yscale` and, if exist, saves the values (`1.0` by default) at `\xtxscale` and `\xtyscale`, respectively. And If the optional value of `scale` exists, it is saved at `\xtscale` only when neither `xscale` nor `yscale` exists. You can use these values to calculate something you want, like `5*1/\xtscale`. The extracted values are internally used to get the best results of the shapes of bubble type information sets.

- The istgame environment also checks if the arrow option `[->]` exists in the option list for the environment. If it exists the istgame adds `shorten >=<dim>` (by default `0pt`) to the list as the first option together with `[->]`. You can change the default value, like `\setistgameshorten{1.3pt}`. (Though this is *not for most users*, you can see some more details, in Section 12.1.2.)

3.2 Node styles

3.2.1 Basic node styles

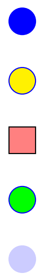
The `tikzstyle`'s of the six basic node styles are predefined.

- plain node: draws nothing (default: `inner sep=1pt`)
- null node: . (very small node) (default: `minimum size=0.2pt`)
- solid node: • (default node style) (default: `minimum size=2.4pt`)
- hollow node: ∘ (default: `minimum size=2.8pt`)
- rectangle node: □ (defaults: `inner sep=2pt`, `minimum size=4pt`)
- ellipse node: ○ (defaults: `inner sep=1.5pt`, `minimum size=4.8pt`)

For some special cases, you may want to change some node styles, including the minimum size. This can be done by `\setist<...>NodeStyle`, all of whose arguments are optional.

```
syntax:
  \setistPlainNodeStyle{<inner sep dim>}{<outer sep dim>}
  \setistNullNodeStyle[<draw color>]{<min-size dim>}[<bg color>][<opacity>]
  \setistSolidNodeStyle[<draw color>]{<min-size dim>}[<bg color>][<opacity>]
  \setistHollowNodeStyle[<draw color>]{<min-size dim>}[<bg color>][<opacity>]
  \setistRectangleNodeStyle[<draw color>]{<min-size dim>}[<bg color>][<opacity>]
  \setistEllipseNodeStyle[<draw color>]{<min-size dim>}[<bg color>][<opacity>]
```

```
% Examples:
\begin{istgame}\setistSolidNodeStyle[blue]{10pt}
    \istroot(0)[solid node]\endist\end{istgame}\\[1ex]
\begin{istgame}\setistHollowNodeStyle[blue]{10pt}[yellow]
    \istroot(0)[hollow node]\endist\end{istgame}\\[1ex]
\begin{istgame}\setistRectangleNodeStyle{10pt}[red][.5]
    \istroot(0)[rectangle node]\endist\end{istgame}\\[1ex]
\begin{istgame}\setistEllipseNodeStyle[blue]{10pt}[green]
    \istroot(0)[ellipse node]\endist\end{istgame}\\[1ex]
\begin{istgame}\setistNullNodeStyle[blue!20]{10pt}
    \istroot(0)[null node]\endist\end{istgame}
```



These basic node styles have their aliases, for convenience, for those who are familiar with game theoretic terminology.

```
% aliases for game theorists
\tikzset{decision node/.style=solid node}  % decision nodes
\tikzset{terminal node/.style=solid node}  % terminal nodes
\tikzset{initial node/.style=hollow node}
\tikzset{chance node/.style=hollow node}
```

The set of all nodes of a game tree can be partitioned into the set of decision nodes and that of terminal nodes. You can use initial node to distinguish the `root` (or the `initial node`) of a game tree from decision nodes. You can also use chance node to represent a chance node of a game tree.

Additional convenient node aliases are also provided: box node, square node, and oval node.

```
% some more aliases
\tikzset{box node/.style=rectangle node}
\tikzset{square node/.style=rectangle node}
\tikzset{oval node/.style=ellipse node}
```

For aliases, you can also change the node styles, like `\setistDecisionNodeStyle[blue]{3pt}` or `\setistBoxNodeStyle{3pt}[green][.5]`.

3.2.2 Your own node styles: `\setistNewNodeStyle`

You can create your own node style by `\setistNewNodeStyle`.

```
% \setistNewNodeStyle
% syntax:
  \setistNewNodeStyle{<style name>}[<opt>]{<minimum size>}
% defaults:
  {<m>}[-,circle,draw=black,fill=white,inner sep=1pt]{6mm}
```

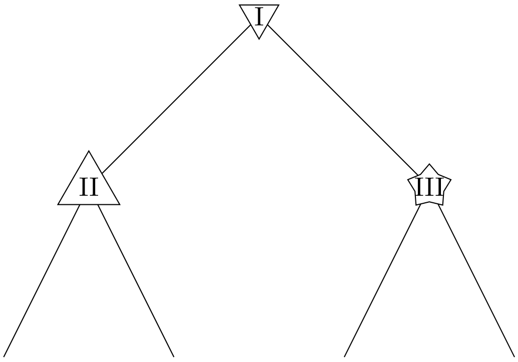
The first, mandatory denoted by `{<m>}`, argument is `{style name}` to be used. The second, optional, argument is `[<options>]` to determine the style of a new node, with defaults. The third, optional, argument should be `{<minimum size>}` only in dimension, which may be frequently used.

To define, for example, a circle (by default) node with the minimum size of `3mm` filled with `red`, you can do like this: `\setistNewNodeStyle{new node}[fill=red]{3mm}`. In fact, this is an abbreviation of the following Ti*k*Z macro:

```
\tikzset{new node/.style={
          circle , draw=black , fill=red , innser sep=1pt , minimum size=3mm }
        }
```

Here is an example of using `\setistNewNodeStyle`.

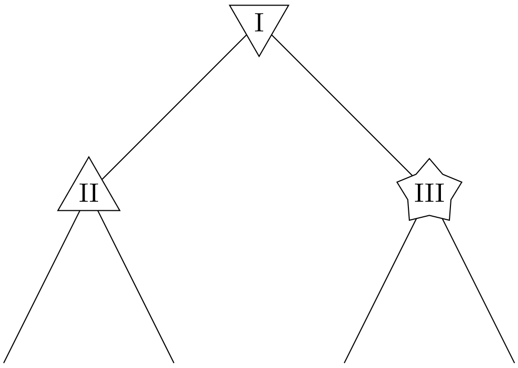
```
% Example: \setistNewNodeStyle
\begin{istgame}[scale=1.5]
\setistNewNodeStyle{new 1}
  [regular polygon,regular polygon sides=3,
   shape border rotate=180]
\setistNewNodeStyle{new 2}
  [regular polygon,regular polygon sides=3]
\setistNewNodeStyle{new 3}[star]
\xtdistance{15mm}{30mm}
\istroot(0)[new 1]<center>{I}
  \istb  \istb  \endist
\xtdistance{15mm}{15mm}
\istrooto(1)(0-1)[new 2]{II} % \istrooto
  \istb  \istb  \endist
\istroot(2)(0-2)[new 3]<center>{III}
  \istb  \istb  \endist
\end{istgame}
```



Remark: From the above example, observe that when you use `\istroot` to put the owner of a node, you need to specify `<center>` for the position of the node owner. However, if you use the oval version `\istrooto`, you don't need that. (About `\istroot` and `\istrooto`, see Section 4.)

Remark: In Ti*k*Z, the shape of a node is independent of the `scale` option in tikzpicture. If you want to make the shape scaled according to the `scale` option, you can use the Ti*k*Z option transform shape, as shown below. (In this kind of case, `\istrooto` helps, rather than `\istroot`.)

```
% Example: \setistrNewNodeStyle (scaled)
\begin{istgame}[scale=1.5]
\setistNewNodeStyle{new 1}
  [regular polygon,regular polygon sides=3,
   shape border rotate=180]
\setistNewNodeStyle{new 2}
  [regular polygon,regular polygon sides=3]
\setistNewNodeStyle{new 3}[star]
\xtdistance{15mm}{30mm}
\istroot(0)
  [new 1,transform shape]<center>{I}
  \istb  \istb  \endist
\xtdistance{15mm}{15mm}
\istrooto(1)(0-1)[new 2]{II} % \istrooto
  \istb  \istb  \endist
\istroot(2)(0-2)
  [new 3,transform shape]<center>{III}
  \istb  \istb  \endist
\end{istgame}
```



4 Core macro: `\istroot`

4.1 `\istroot`: basics

4.1.1 `\istroot` – counterclockwise: standard version

The macro `\istroot` defines the *root* of a game or a subgame at a designated location, specifies the *owner* of the root or the subroot, and does other functions. In game theoretic terminology, `\istroot` designates a decision node and its owner (or a player).

```
% \istroot
% syntax:
  \istroot[<grow keyval>,<tree opt>](<coor1>)(<coor2>)
          [<node style>,<node opt>]<[owner opt]owner label angle>{<owner>}
          +<lev-distance>..<sib-distance>+
% defaults:
  [south](<m>)(0,0)[decision node]<above>{}+15mm..15mm+
% arguments: (coor1) is mandatory, all others are optional arguments
  [grow] % the direction of growing <default: south>
  (coor1) % name of the (sub)root: mandatory
  (coor2) % the (sub)root is at (coor2) <default: (0,0)>
  [node style] % node style <default: decision node>
  <angle> % position of owner name <default: above>
  {owner} % name of the owner of the (sub)root
  +level dist..sibling dist+ % <defaults: 15mm,15mm>
```

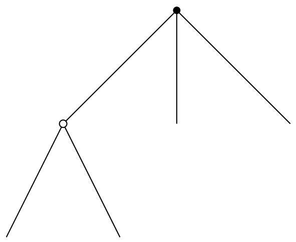
the root

The only mandatory argument, denoted by `<m>`, of `\istroot` is `(<coor1>)`, which gives the name of the root or subroot. All the other arguments are optional. The name of the (sub)root, `(<coor1>)`, can be referred as a normal coordinate. `(coor2)` specifies the location where the (sub)root is placed. If `(coor2)` is omitted, it is regarded as `(0,0)` by default.

The default node style of the root is a `decision node`, which is just a `solid node`. You can change the node style to any other node style such as `initial node`, `chance node`, `oval node`, `box node`, and so on.

Here is a simple example of drawing a tree structure.

```
% Example: \istroot
\begin{istgame}
\istroot(0)
  \istb  \istb  \istb  \endist
\istroot(a)(0-1)[chance node]
  \istb  \istb  \endist
\end{istgame}
```

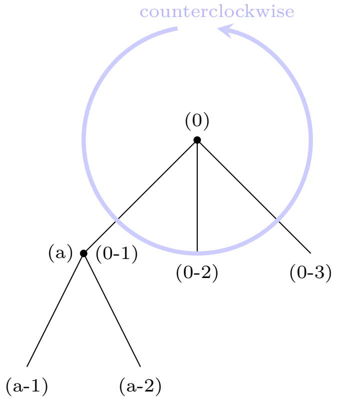


naming children: counterclockwise or clockwise

In the previous example, the game tree has the root named `(0)`, located at `(0,0)` by default, which has three branches (by three `\istb`'s). Since Ti*k*Z arranges branches of a tree `counterclockwise`, by default, the endpoints of the three branches are automatically named `(0-1)`, `(0-2)`, and `(0-3)` *from left to right* (when a tree grows down).

The root of the subtree is named `(a)`, located at `(0-1)`, and has two children. Its children are automatically named `(a-1)` and `(a-2)` `counterclockwise` (or from left to right if the tree grows down). See the following code example with explanatory labels to see what is going on.

```
% Example: \istroot (explained with labels)
\begin{istgame}[font=\scriptsize]
\istroot(0)
  \istb  \istb  \istb \endist
\istroot(a)(0-1)
  \istb  \istb  \endist
%% labels: (ignore the following lines at the moment)
\setistmathTF000
\xtOwner(0){(0)}
\xtOwner(a){(a)}[l]
\xtPayoff*(0-1){(0-1)}[r]
\xtPayoff*(0-2){(0-2)}[b]
\xtPayoff*(0-3){(0-3)}[b]
\xtPayoff*(a-1){(a-1)}[b]
\xtPayoff*(a-2){(a-2)}[b]
\draw [->,ultra thick,blue!20](-260:1.5)
  arc (-260:80:1.5cm)
  node [above,blue!30] {counterclockwise};
\end{istgame}
```



owner (or player)

The owner of a decision node (or a player) is expressed in curly braces, like `{player 1}`, and printed in *text mode*. The input mode and text font style of an owner can be changed by `\setistmathTF(*)` (see Section 7 on page 36, for more details).

The position of the owner of a decision node is specified in angle brackets, like `<90>`, `<above>`, or `<north>`. To specify the position of an owner you can use `<degrees>`, or the compass directions such as `<north>`, `<south>`, `<east>`, `<west>`, and their valid combinations. You can also use the positional words such as `<above>`, `<below>`, `<left>`, `<rigth>`, and their valid combinations.

growing direction of a simple tree

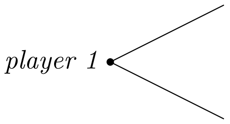
The first bracket option is mainly for the direction of a simple tree (`[south]` by default). Internally, `[<grow keyval>]` typed in as the first option of `\istroot` renews the direction of tree growing by assigning its value to `\istgrowdirection`, whose default is `south`.

Remark:

- In fact, the first option of `\istroot` controls the features of an whole simple tree (but a node style), while the second bracket option controls a node style only.
- In addition to the direction of a simple tree, you can add more options to control the whole branch styles and their labels (except a node sytle). For example, if you want, at any reason, to draw a simple tree with all *red dashes branches with red labels* growing south-eastwards, you can do like `[south east,red,dashed]`. (For examples in more detail, see Section 6.1.2 on page 27.)
- Be aware that the first entry in the option list *must* be a *directional word* for a simple tree.

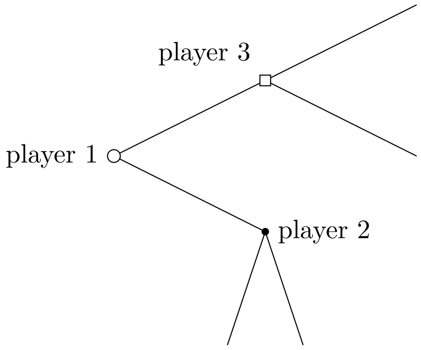
The tree growing direction can be specified by `[<degrees>]` or by using the compass directions such as `[north]`, `[south]`, `[east]`, `[west]`, `[north east]`, `[north west]`, `[south east]`, `[south west]`. You can also use positional words like `[left]`, `[rigth]`, `[down]`, and `[up]`, but you cannot use `[above]` nor `[below]`.

```
% Example 1: \istroot (one simple tree)
\begin{istgame}[font=\itshape]
\istroot[right](0)<left>{player 1}
  \istb  \istb  \endist
\end{istgame}
```



Remark: One thing you should remember about this is that `\istgrowdirection` is internally used in the definition of `\istb` to control the label position for payoffs. However, for the label position in Ti*k*Z, `[below]` and `[above]` are good, but not `[down]` nor `[up]`. So DO NOT USE `[down]` nor `[up]` to specify the tree growing direction.

```
% Example 2: \istroot (three simple trees connected)
\begin{istgame}
\xtdistance{20mm}{20mm}
\istroot[right](0)[oval node]<left>{player 1}
  \istb  \istb  \endist
\istroot(a)(0-1)<right>{player 2}+15mm..10mm+
  \istb  \istb  \endist
\istroot[right](b)(0-2)[box node]<135>{player 3}
  \istb  \istb  \endist
\end{istgame}
```



local change of distances

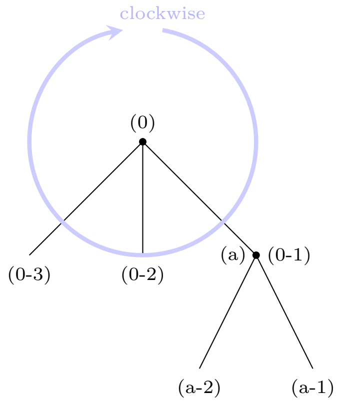
The last two options of `\istroot` specify the `level distance` and the `sibling distance`. This *local* change of distances is valid only for the corresponding simple tree, while the distances changed by `\xtdistance` are valid within the current istgame environment unless they are changed again by `\xtdistance`. Do not forget, when you use decimal distances, to delimit the decimal dimensions with curly braces, like `+{15.5mm}..{10.5mm}+`.

4.1.2 `\istroot'` – clockwise: swap version

The macro `\istroot'` is the swap version of `\istroot`. `\istroot'` works just like `\istroot` but with one exception: going *clockwise* instead of *counterclockwise*. `\istroot'` arranges its branches `clockwise` (or from right to left if the tree grows down).

Compare the following example with that of `\istroot` above on page 15. The two examples have exactly the same codes as each other except for one thing: either `\istroot` or `\istroot'`.

```
% Example: \istroot' (explained with labels)
\begin{istgame}[font=\scriptsize]
\istroot'(0)
  \istb  \istb  \istb \endist
\istroot'(a)(0-1)
  \istb  \istb  \endist
%% labels: (ignore the following lines at the moment)
\setistmathTF000
\xtOwner(0){(0)}
\xtOwner(a){(a)}[l]
\xtPayoff*(0-1){(0-1)}[r]
\xtPayoff*(0-2){(0-2)}[b]
\xtPayoff*(0-3){(0-3)}[b]
\xtPayoff*(a-1){(a-1)}[b]
\xtPayoff*(a-2){(a-2)}[b]
\draw [<-,ultra thick,blue!20](-260:1.5)
  arc (-260:80:1.5cm)
  node [above,blue!30] {clockwise};
\end{istgame}
```



We will look into this issue (of going counterclockwise or clockwise) in more detail in Section 9 (on page 41), where we discuss the tree growing direction.

If you draw a game tree growing south, you don't need to worry about the swap version `\istroot'`. Just use `\istroot`.

4.2 `\istrooto`: oval version

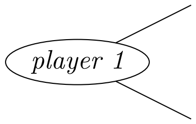
4.2.1 `\istrooto` – counterclockwise

The macro `\istrooto` is the oval version of `\istroot`. This allows us to draw a bubble (by default, `oval node`) with a node owner (or a game player) in it. Except for this difference, `\istrooto` works just like `\istroot`. Since an owner is shown in a specified node with `\istrooto`, the option `<owner label angle>` of the standard version `\istroot` is ignored.

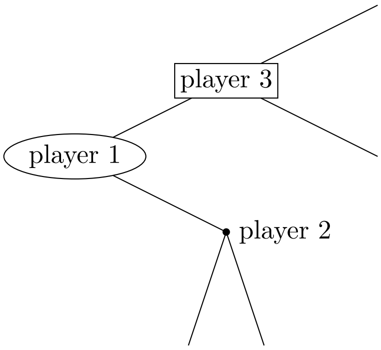
```
% \istrooto
% syntax:
  \istrooto[<grow keyval,tree opt>](<coor1>)(<coor2>)
           [<node style>,<node opt>]{<owner>}+<lev-distance>..<sib-distance>+
% default: only (coor1) is mandatory, <m>, all others optional
  [south](<m>)(0,0)[oval node]{}+15mm..15mm+
```

The following two examples are the same as above with `\istroot` on pages 15 and 16, respectively, but now with the oval version `\istrooto`.

```
% Example 1: \istrooto (one simple tree)
\begin{istgame}[font=\itshape]
\istrooto[right](0)<180>{player 1}
  \istb  \istb  \endist
\end{istgame}
```

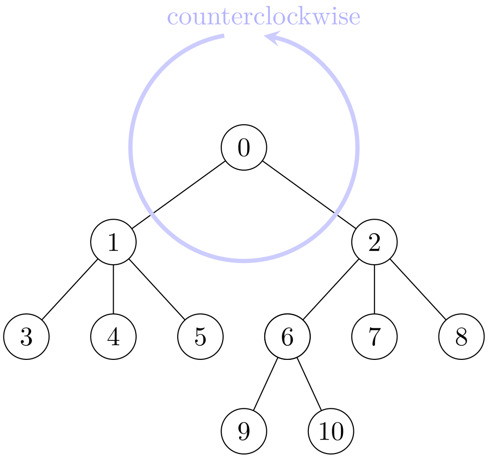


```
% Example 2: \istrooto (three simple trees connected)
\begin{istgame}
\xtdistance{20mm}{20mm}
\istrooto[right](0)[oval node]<left>{player 1}
  \istb  \istb  \endist
\istroot(a)(0-1)<right>{player 2}+15mm..10mm+
  \istb  \istb  \endist
\istrooto[right](b)(0-2)[box node]<135>{player 3}
  \istb  \istb  \endist
\end{istgame}
```



Observe that the angle `<180>`, `<left>`, `<right>`, or `<135>` specifying the position of an owner's name is redundant to `\istrooto`.

```
% Example 3: \istrooto (counterclockwise)
\begin{istgame}
\setistOvalNodeStyle{.6cm}
\istrooto(0){0}+{12.5mm}..{3.45cm}+
  \istb  \istb  \endist
\xtShowEndPoints[oval node]
\xtdistance{12.5mm}{11.5mm}
\istrooto(1)(0-1){1}
  \istb{}{3}[center]  \istb{}{4}[center]
  \istb{}{5}[center]  \endist
\istrooto(2)(0-2){2}
  \istb{}{6}[center]  \istb{}{7}[center]
  \istb{}{8}[center]  \endist
\istrooto(6)(2-1){6}
  \istb{}{9}[center]  \istb{}{10}[center]
  \endist
\draw [->,ultra thick,blue!20](-260:1.5)
  arc (-260:80:1.5cm)
  node [above,blue!30] {counterclockwise};
\end{istgame}
```

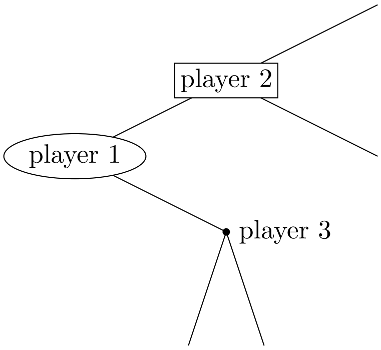


The previous example shows how `\istrooto` arranges its branches: `counterclockwise`, by default.

4.2.2 `\istrooto'` – clockwise: swap version

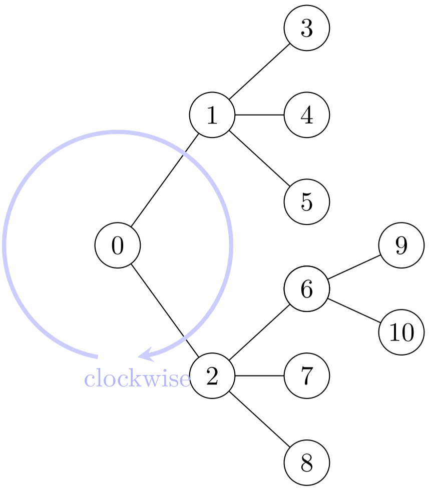
The swap version of the oval version, `\istrooto'`, works just like `\istroot'` with one exception that it puts an owner within an `oval node`, by default.

```
% Example 2: \istrooto' (clockwise)
\begin{istgame}
\xtdistance{20mm}{20mm}
\istrooto'[right](0)[oval node]<left>{player 1}
  \istb  \istb  \endist
\istrooto'[right](b)(0-1)[box node]<135>{player 2}
  \istb  \istb  \endist
\istroot(a)(0-2)<right>{player 3}+15mm..10mm+
  \istb  \istb  \endist
\end{istgame}
```



The swap version is useful when a tree grows northwards or eastwards. The example below shows a tree rotated to the east by `\setistgrowdirection`. Note that `\istrooto'` arranges its branches `clockwise`.

```
% Example 3: \istrooto' (clockwise)
\begin{istgame}
\setistgrowdirection'{east}
\setistOvalNodeStyle{.6cm}
\istrooto'(0){0}+{12.5mm}..{3.45cm}+
  \istb  \istb  \endist
\xtShowEndPoints[oval node]
\xtdistance{12.5mm}{11.5mm}
\istrooto(1)(0-1){1}
  \istb{}{3}[center]  \istb{}{4}[center]
  \istb{}{5}[center]  \endist
\istrooto(2)(0-2){2}
  \istb{}{6}[center]  \istb{}{7}[center]
  \istb{}{8}[center]  \endist
\istrooto(6)(2-1){6}
  \istb{}{9}[center]  \istb{}{10}[center]
  \endist
\draw [->,ultra thick,blue!20](260:1.5)
  arc (260:-80:1.5cm)
  node [below,blue!30] {clockwise};
\end{istgame}
```



Remark: When you use the swap version `\setistgrowdirection'`, using either `\istroot` or `\istroot'` makes no difference. (For more details, see Section 9 on page 41.)

5 Core macro: `\istb`

5.1 `\istb`: basics

5.1.1 Basics: branches, action labels, and payoffs

The macro `\istb`, basically, prints a *branch*. Having all arguments as options, a simple `\istb` draws a branch from a parent node designated by `\istroot` to a child node (or endpoint of `\istb`). If, for example, a parent node is named `(A)` by `\istroot`, the first child node is automatically named `(A-1)`, the second child node `(A-2)`, and so on.

The macro `\istb` also puts an *action* label and *payoffs* along with a branch, and does other functions. Note that the *action labels* and *payoffs* are to be typeset in *math mode*. If you want to change the input mode to text mode, you can use the macro `\setistmathTF(*)`. This issue is discussed in Section 5.1.2 below. (You can also see Section 7, on page 36, for more details).

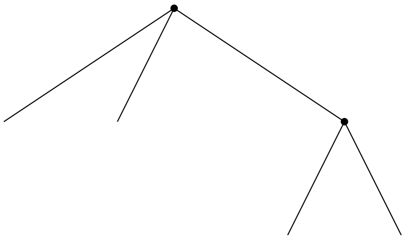
```
% \istb
% syntax:
  \istb<grow, distance, missing>[<line style>]{<action>}[<pos>]{<payoff>}[<pos>]
% defaults:
  <grow=south>{-}{}[center]{}[\istgrowdirection]
% arguments: all arguments are optional
  <grow=keyval> % the direction of a branch <default: south>
  [line style] % branch line style <default: solid>
  {action} % action label (in math mode)
  [action pos] % position of action label <default: center>
  {payoff} % payoffs (in math mode)
  [payoff pos] % position of payoffs <default: \istgrwodirection: south>
```

Remark: The macro `\istb` has many variants including `\istb*` (starred version), `\istbt` (terminal version), and `\istbm` (missing version). It also has other very close friends: `\istB` (dual label version) and `\istbA` (alternative or arc version). Each friend of `\istb` has its starred version and terminal version, of course, except `\istbm`. `\cntmistb(*)` and `\cntmAistb(*)` are also friends only when a continuum of branches is in play.

branches

In the example below, each `\istb` draws a branch. With the option missing, `\istb` prints an *invisible* branch. Since the third child is *missing*, the last child is named `(0-4)`.

```
% Example: \istb<missing> (simply, \istbm)
\begin{istgame}
\istroot(0)
  \istb  \istb  \istb<missing>  \istb  \endist
\istroot(D)(0-4)
  \istb  \istb  \endist
\end{istgame}
```



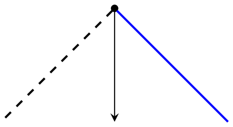
To make it simple, you can use the *missing version* `\istbm` instead of `\istb<missing>`.

```
% \istbm
  \newcommand\istbm{\istb<missing>}
```

The macro `\istb` also has various options to control the line style of a branch, and the direction and length of a branch. `\istb` can also place payoffs to the direction of tree growth by default.

For various line styles of each branch, you can use any Ti*k*Z options of arrows, line style, color, and so on.

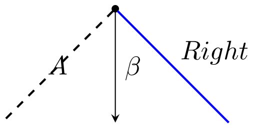
```
% Example: \istb (branch line styles)
\begin{istgame}
\istroot(0)
  \istb[dashed,thick]
  \istb[->]
  \istb[draw=blue,thick]
  \endist
\end{istgame}
```



action labels

By default, an action label is put on the midpoint of the corresponding branch, in *math* mode.

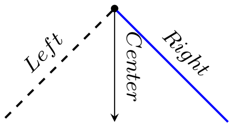
```
% Example: action labels
\begin{istgame}
\istroot(0)
  \istb[dashed,thick]{A}
  \istb[->]{\beta}[right]
  \istb[draw=blue,thick]{Right}[ar]
  \endist
\end{istgame}
```



To specify the position of an action label, you can use the positional words or their *abbreviations*:
`[a]` for `[above]`, `[b]` for `[below]`, `[l]` for `[left]`, `[r]` for `[right]`,
`[al]` for `[above left]`, `[ar]` for `[above right]`,
`[bl]` for `[below left]`, and `[br]` for `[below right]`.

Remark: Note that these abbreviations must be used with no other options, otherwise you will get a compile *error*. For example, when you want to print sloped labels for actions, you should do like `[above,sloped]` but not like `[a,sloped]`.

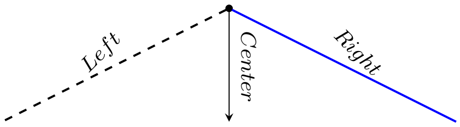
```
% Example: sloped labels
\begin{istgame}[font=\footnotesize]
\istroot(0)
  \istb[dashed,thick]{Left}[above,sloped]
  \istb[->]{Center}[above,sloped]
  \istb[draw=blue,thick]{Right}[above,sloped]
  \endist
\end{istgame}
```



Warning: Issues in *sloped labels* in Ti*k*Z with *asymmetric scales*:

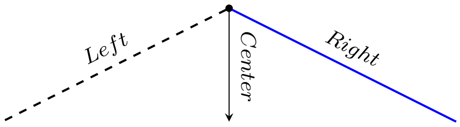
- The `tree` library in Ti*k*Z does not seem to treat sloped labels properly, when `xscale` or `yscale` is used asymmetrically.

```
% Example: sloped labels problem
\begin{istgame}[xscale=2,font=\footnotesize]
%\xtcureslopedlabelsNS
\istroot(0)
  \istb[dashed,thick]{Left}[above,sloped]
  \istb[->]{Center}[above,sloped]
  \istb[draw=blue,thick]{Right}[above,sloped]
  \endist
\end{istgame}
```



- This package provides a temporary solution to cure this issue: `\xtcureslopedlabelsNS` for trees growing northwards or southwards. If a tree grows eastwards or westwards, then use `\xtcureslopedlabelsEW`. These must be used in a TeX group with caution. *These are only temporary solutions that have not been tested for every occasion.*

```
% Example: sloped labels problem cured
\begin{istgame}[xscale=2,font=\footnotesize]
\xtcureslopedlabelsNS
\istroot(0)
  \istb[dashed,thick]{Left}[above,sloped]
  \istb[->]{Center}[above,sloped]
  \istb[draw=blue,thick]{Right}[above,sloped]
  \endist
\end{istgame}
```

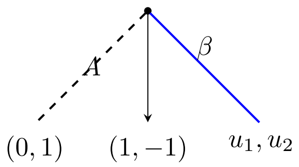


payoffs

The second curly braces option of the macro `\istb` is for *payoffs* and the last bracket option for the position of payoffs.

By default, `\istb` prints payoffs in *math mode*, which can be changed by `\setistmathTF(*)`. By default, payoffs are put in the direction of `\istgrowdirection` (`[south]` by default).

```
% Example: payoffs
\begin{istgame}
\istroot(0)
  \istb[dashed,thick]{A}{(0,1)}
  \istb[->]{}{(1,-1)}
  \istb[draw=blue,thick]{\beta}[a]{u_1,u_2}
  \endist
\end{istgame}
```

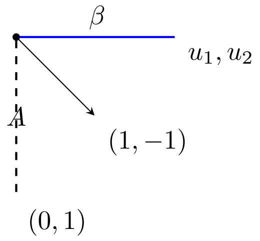


Remark: What is `\istgrowdirection` and what is it used for?

- `\istgrowdirection` has the value of `[<grow keyval>]` typed in `\istroot` (default: `south`).
- The value of `\istgrowdirection` is (internally) used to determine the direction of putting *payoffs* (`south` by default).

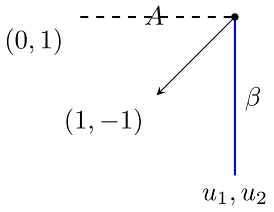
In the following example, notice that the tree grows south-eastwards, so the payoffs are placed to the south-east of the endpoints.

```
% Example: tree toward south east (or -45 degree)
\begin{istgame}
\istroot[-45](0)
  \istb[dashed,thick]{A}{(0,1)}
  \istb[->]{}{(1,-1)}
  \istb[draw=blue,thick]{\beta}[a]{u_1,u_2}
  \endist
\end{istgame}
```



If you do not like the position of payoffs, you can change it by using degrees, the compass directions, or the positional words and their *abbreviations* mentioned above. In the example below, the tree grows south-westwards, but the position of payoffs at the end of the blue branch is changed to `[below]` or `[b]`.

```
% Example: tree toward south west (or 225 degree)
\begin{istgame}
\istroot[south west](0)
  \istb[dashed,thick]{A}{(0,1)}
  \istb[->]{}{(1,-1)}
  \istb[draw=blue,thick]{\beta}[r]{u_1,u_2}[below]
  \endist
\end{istgame}
```



Remark:

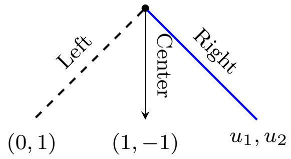
- `\istb` expects a *directional word* to be input as the last optional argument (by default, `\istgrowdirection`).
- You can add more options to change the position of payoffs, but the first entry of the option list *must* be a *directional word*, like `[below,xshift=5mm]`, but not like `[xshift=5mm,below]`. Otherwise, a compile *error* will be produced.
- Note also that you can do like `[[xshift=5mm]below`, instead (see also page 41).

5.1.2 Printing action labels in italics in text mode: `\setistmathTF*`

By default, an owner is printed in *text mode*, action labels and payoffs in *math mode*. With the macro `\setistmathTF(*)`, you can change the input mode for those labels. These are discussed in Section 7 on page 36.

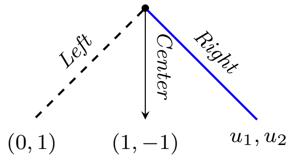
Here, it is briefly discussed how to change the input mode of action labels to *text mode*. To do that, just declare `\setistmathTF001`. The second zero means that the input mode for action labels is in text mode. (The first number is for owners and the third one for payoffs.)

```
\begin{istgame}[font=\footnotesize]
\setistmathTF001
\istroot(0)
  \istb[dashed,thick]{Left}[above,sloped]{(0,1)}
  \istb[->]{Center}[above,sloped]{(1,-1)}
  \istb[draw=blue,thick]{Right}[above,sloped]{u_1,u_2}
  \endist
\end{istgame}
```



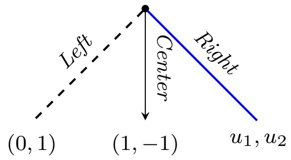
If you use the starred version, like `\setistmathTF*001`, action labels are printed in *italics*, by default.

```
\begin{istgame}[font=\footnotesize]
\setistmathTF*001
\istroot(0)
  \istb[dashed,thick]{Left}[above,sloped]{(0,1)}
  \istb[->]{Center}[above,sloped]{(1,-1)}
  \istb[draw=blue,thick]{Right}[above,sloped]{u_1,u_2}
  \endist
\end{istgame}
```



You can, of course, do the same thing in the default math mode.

```
\begin{istgame}[font=\footnotesize]
%\setistmathTF011 (default modes)
\istroot(0)
  \istb[dashed,thick]
       {\textit{Left}}[above,sloped]{(0,1)}
  \istb[->]{\textit{Center}}[above,sloped]{(1,-1)}
  \istb[draw=blue,thick]
       {\textit{Right}}[above,sloped]{u_1,u_2}
  \endist
\end{istgame}
```

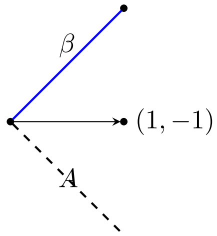


5.2 `\istb*`: starred version

5.2.1 `\istb*`: basics

The starred version `\istb*` prints a `solid node` at the end of the corresponding branch. This is the only difference between `\istb` and `\istb*`.

```
\begin{istgame}
\istroot[east](0)
  \istb[dashed,thick]{A}
  \istb*[->]{}{(1,-1)}
  \istb*[draw=blue,thick]{\beta}[a]
  \endist
\end{istgame}
```



5.2.2 `\xtShowEndPoints` and `\xtHideEndPoints`

Each endpoint is printed by each execution of `\istb*`. You can print solid nodes (by default) at *all* the endpoints of *simple trees* by the macro `\xtShowEndPoints`. You can also change the style of nodes for all the endpoints of simple trees, by specifying it as an optional argument, like `\xtShowEndPoints[oval node]`.

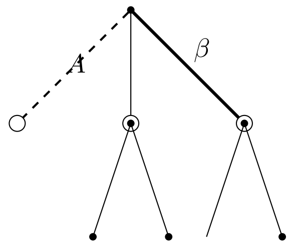
```
% \xtShowEndPoints
% syntax:
  \xtShowEndPoints[<node style>]
% default:
  [solid node]
```

The macro `\xtHideEndPoints` turns off the effects of `\xtShowEndPoints`.

It is too early to say that the starred version `\xtShowEndPoints*` additionally prints the two outermost endpoints of a continuum. `\xtHideEndPoints*` turns off only the outermost endpoints a continuum, but not the other endpoints. (See Section 11.4.3 on page 67, for more details.)

Here is an example of using `\xtShowEndPoints` and `\xtHideEndPoints`.

```
\begin{istgame}
\xtShowEndPoints[oval node,minimum size=6pt]
\istroot(0)[solid node]
  \istb[dashed,thick]{A}
  \istb
  \istb[very thick]{\beta}[ar]
  \endist
\xtShowEndPoints % default: solid nodes
\xtdistance{15mm}{10mm}
\istroot(b)(0-2) \istb  \istb  \endist
\xtHideEndPoints % \istb* overrides
\istroot(b)(0-3) \istb  \istb*  \endist
\end{istgame}
```



Note that `\xtShowEndPoints` and `\xtHideEndPoints` should be in an istgame environment to avoid unexpected results. Note also that `\istb*` overrides all the effects of these two macros by forcing to print a solid node.

5.3 `\istbt`: terminal version

5.3.1 `\istbt`: basics

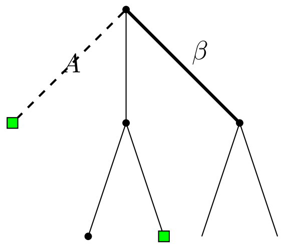
The terminal version `\istbt` is designed to represent a *terminal move* in a game tree. Basically, `\istbt` works exactly the same way as `\istb` does. However, using `\istbt` together with the macro `\xtShowTerminalNodes` you can control the shape of the terminal nodes, *all at once*.

5.3.2 `\xtShowTerminalNodes` and `\xtHideTerminalNodes`

The terminal version `\istbt` used with `\xtShowTerminalNodes` prints a `solid node` (by default). You can change the style of the terminal nodes, like `\xtShowTerminalNodes[oval node]`. This effect can be turned off by `\xtHideTerminalNodes`.

```
% \xtShowTerminalNodes (works only with \istbt and other terminal versions)
% syntax:
  \xtShowTerminalNodes[<node style>]
% default:
  [solid node]
```

```
% Example: \xtShowTerminalNodes, \xtHideTerminalNodes
\begin{istgame}
\xtShowTerminalNodes[box node,fill=green]
\istroot(0)[solid node]
  \istbt[dashed,thick]{A}
  \istb
  \istb[very thick]{\beta}[ar]
  \endist
\xtdistance{15mm}{10mm}
\istroot(b)(0-2) \istbt*  \istbt  \endist
\xtHideTerminalNodes
\istroot(b)(0-3) \istbt   \istbt  \endist
\end{istgame}
```



Remark: Note that controlling terminal nodes by using `\xtShowTerminalNodes` works only with the terminal versions `\istbt`, `\istBt` (documented below in Section 5.4) and `\istbAt` (in Section 5.5), but there will be no effect and no harm with other versions. Note also that `\istbt*` overrides the effects of `\xtShowTerminalNodes` and `\xtHideTerminalNodes`.

5.4 `\istB`: dual label version

5.4.1 `\istB`: basics

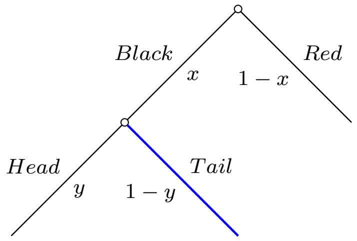
The macro `\istB` works just like `\istb`, except one thing: `\istB` prints *dual labels* for a branch.

The starred version `\istB*` prints a solid node at the end of the corresponding branch, just like `\istb*`. And the terminal versions `\istBt` and `\istBt*` work just like `\istbt` and `\istbt*`, respectively, except for dual labels.

```
% \istB
% syntax:
  \istB<grow>[<opt>]{<action1>}[<pos1>]{<action2>}[<pos2>]{<payoff>}[<pos>]
% defaults:
  <grow=south>{-}{}[center]{}[center]{}[\istgrowdirection]
% arguments: all arguments are optional
  <grow=keyval> % the direction of a branch <default: south>
  [line style] % branch line style <default: solid>
  {action label 1} % action label 1 (in math mode)
  [action pos 1] % position of action label 1 <default: center>
  {action 2} % action label 2 (in math mode)
  [action label pos 2] % position of action label 2 <default: center>
  {payoff} % payoffs (in math mode)
  [payoff pos] % position of payoffs <default: \istgrwodirection: south>
```

With `\istB`, do not forget to put two labels. Otherwise, you might get an unexpected result.

```
% \istB
\begin{istgame}[font=\footnotesize]
\xtdistance{15mm}{30mm}
\istroot(0)[chance node]
  \istB{Black}[al]{x}[br]
  \istB{Red}[ar]{1-x}[bl]
  \endist
\istroot(1)(0-1)[initial node]
  \istBt{Head}[al]{y}[br]
  \istBt[draw=blue,thick]{Tail}[ar]{1-y}[bl]
  \endist
\end{istgame}
```



5.4.2 `\xtActionLabel` and `\xtActionLabel*`

You can also use the supplementary macro `\xtActionLabel` to print additional action labels (see also Section 12.2 on page 73). The macro `\xtActionLabel` controls, from outside of a simple tree, a branch (by default `draw=none`) with a label (in math mode by default), connecting two nodes.

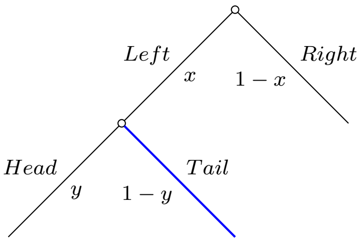
```
% \xtActionLabel
% syntax:
  \xtActionLabel*[<line opt>](<from>)(<to>){<action>}[<pos>,<node opt>]
% default:
  [-,draw=none](<m>)(<m>){}[black,text depth=.25]
```

Remark: In some cases, it is not a good idea to use `\istB` to print dual action labels.

- With `\istB`, the features of *middle arrows* provided by this package does not work well (see Section 12.2 on page 73).
- With respect to a *continuum* of branches, `\istB` has little role to play.

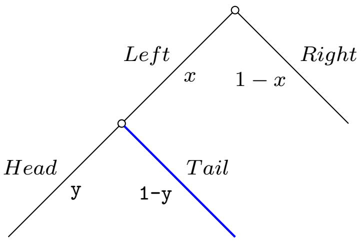
You can do the same thing as `\istB` does by using `\istb` and together with `\xtActionLabel`.

```
% Example: \istb and \xtActionLabel
\begin{istgame}[font=\footnotesize]
\xtdistance{15mm}{30mm}
\istroot(0)[chance node]
  \istb{Left}[al]
  \istb{Right}[ar]
  \endist
\istroot(1)(0-1)[initial node]
  \istbt{Head}[al]
  \istbt{Tail}[ar]
  \endist
\xtActionLabel(0)(0-1){x}[br]
\xtActionLabel(0)(0-2){1-x}[bl]
\xtActionLabel(1)(1-1){y}[br]
\xtActionLabel[draw,blue,thick](1)(1-2){1-y}[bl]
\end{istgame}
```



The starred version `\xtActionLabel*` prints its label in the input mode as set by `\setistmathTF*` (see Section 7 on page 36, for more details on `\setistmatTF`).

```
% Example: \istb and \xtActionLabel*
\begin{istgame}[font=\footnotesize]
\xtdistance{15mm}{30mm}
\istroot(0)[chance node]
  \istb{Left}[al]
  \istb{Right}[ar]
  \endist
\istroot(1)(0-1)[initial node]
  \istbt{Head}[al]
  \istbt{Tail}[ar]
  \endist
\xtActionLabel(0)(0-1){x}[br]
\xtActionLabel(0)(0-2){1-x}[bl]
\setistmathTF*001{texttt}
\xtActionLabel*(1)(1-1){y}[br]
\xtActionLabel*[draw,blue,thick](1)(1-2){1-y}[bl]
\end{istgame}
```

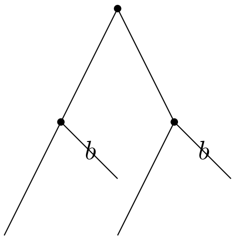


5.5 `\istbA`: alternative (or arc) version

5.5.1 `\istbA`: basics

The macro `\istbA` is an alternative (or arc) version, doing one more thing than `\istb`. With `\istbA` you can *easily change the level distance* of an individual branch using a factor (`1`, by default) *as the first optional argument* in parentheses. All other arguments are the same as in `\istb`. For example, `\istbA(.5)` is an abbreviation of `\istb<level distance=.5*\xtlevdist>`, as you can see in the example below:

```
% Example: \istbA
\begin{istgame}
\istroot(0)      \istb \istbA            \endist
\istroot(1)(0-1) \istb \istbA(.5){b}     \endist
\istroot(2)(0-2) \istb
 \istbA<level distance=.5*\xtlevdist>{b} \endist
\end{istgame}
```



You can interchangeably use `\istbA` and `\istb`, except for one case, in which you are working with a continuum of branches.

Remark: (too early to comment)

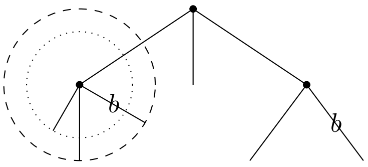
- The macro `\istbA` is originally created to work with `\istrootcntmA` as an arc version.
- `\istbA` draws, by default, a branch up to an arc when used with the continuum arc version `\istrootcntmA`, but `\istb` does not. (For more details, see Section 11.2.1 on page 61.)
- Except for the case of using a continuum of branches, `\istbA` is equivalent to `\istb` for users.

Its terminal version `\istbAt` is also available, but the dual label version of `\istbA` is not provided. The starred versions `\istbA*` and `\istbAt*` print an solid node (by default) at the end of the corresponding branch.

5.5.2 `\istbA`: application

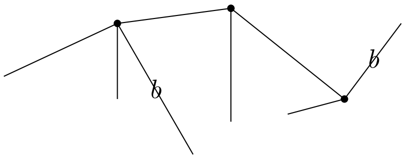
By specifying the `grow` key of `\istb` you can draw a branch with the exact length you want.

```
% Example: \istbA
\begin{istgame}
\xtdistance{10mm}{15mm}
\istroot(0) \istb \istb \istb    \endist
\istroot(1)(0-1)
  \istbA(.7)<grow=-120>  %  7mm long
  \istb                  % 10mm long
  \istbA(1)<grow=-30>{b} % 10mm long
  \endist
\istroot(2)(0-3) \istb \istbA{b} \endist
\draw [dashed] (1) circle (10mm);
\draw [dotted] (1) circle (7mm);
\end{istgame}
```



You can also apply `\istbA` to easily ruin your regular balanced trees.

```
% Example: \istbA
\begin{istgame}
\xtdistance{10mm}{15mm}
\istroot(0) \istbA(.2) \istbA(1.5) \istbA(1.2) \endist
\istroot(1)(0-1)
  \istbA(.7) \istb \istbA(2)<grow=-60>{b} \endist
\istroot(2)(0-3) \istbA(.2) \istbA(-1){b} \endist
\end{istgame}
```



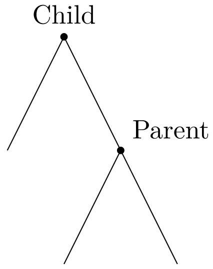
6 Important labels: players, action labels, and payoffs

6.1 How to put players

6.1.1 Players: basics

The macro `\istroot` specifies the (sub)root of a simple tree and puts its owner (or a player). The direction to which a player label is put is set by the angle `< >` option of `\istroot`, like `<above>` (by default), `<east>`, or `<45>`. To specify the direction you can use degrees, the compass directions, or positional words.

```
% Example: node owner
\begin{istgame}
\istroot(0)<above>{Child} % default: <above>
  \istb  \istb  \endist
\istroot(1)(0-2)<45>{Parent}
  \istb  \istb  \endist
\end{istgame}
```

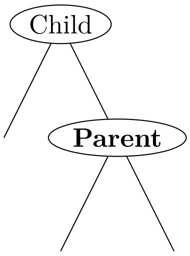


Remark:

- The supplementary macro `\xtOwner` is provided as an extra way of putting owners of decision nodes (see Section 13.1 on page 78).
- Though an owner is to be input in text mode, by default, it is also possible to change the input mode to math mode with `\setistmathTF(*)` (see below in Section 6.1.4 and Section 7 on page 36).

Note that the oval version `\istrooto` produces a bubble type node with an owner in it, so the directional option `<angle>` is redundant with `\istrooto` (see Section 4.2 on page 17).

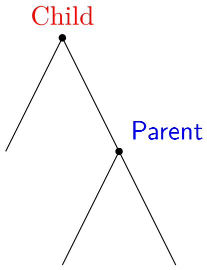
```
% Example: node owner (with \istrooto)
\begin{istgame}
\istrooto(0){Child}
  \istb  \istb  \endist
\istrooto(1)(0-2)<45>{\textbf{Parent}}
  \istb  \istb  \endist
\end{istgame}
```



6.1.2 Coloring players or a whole simple tree using `\istroot`

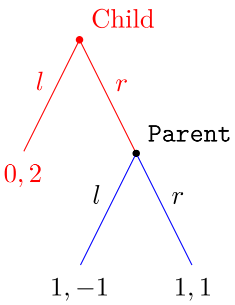
You can change the color of a player's name by giving `color` in the angle option of `\istroot`, like `<[red]>` or `<[blue]45>`. (This is the Ti*k*Z way of giving options for a node `label`.)

```
% Example: coloring players
\begin{istgame}
\istroot(0)<[red]>{Child}
  \istb
  \istb
  \endist
\istroot(1)(0-2)<[blue]45>{\textsf{Parent}}
  \istb
  \istb
  \endist
\end{istgame}
```



The example below shows how you color a whole simple tree: `red` for the Child's simple tree and `blue` for the Parent's simple tree.

```
% Example: coloring a simple tree
\begin{istgame}
\istroot[south,red](0) % whole simple tree, but node
        [draw=red,fill=red] % for node only
        <[red]45>{Child}    % for onwer only
  \istb{l}[al]{0,2}[south]
  \istb{r}[ar]
  \endist
\istroot[south,draw=blue](1)(0-2)<45>{\texttt{Parent}}
  \istb{l}[al]{1,-1}[south]
  \istb{r}[ar]{1,1}[-90]
  \endist
\end{istgame}
```



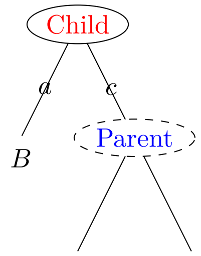
Remark:

- In the above example, the first `\istroot` has three bracket `[option]`'s: the first bracket option is for a *whole simple tree* including *branches* and *labels* (except a node style), the second for a a *node* style of a decision node, and the third for the *owner* of the (sub)root.
  - If you want to have all the branches, action labels, and payoffs in red, just use the color name like `[south,red]`. If you want to have just the branches in blue, use the `draw` option key like `[south,draw=blue]`. (This is what Ti*k*Z does.)
- Note that the *first entry* in the first `[option]` list of `\istroot` *must* be *the growing direction of a simple tree*. So it should be input like `[south,red]`, but never like `[red,south]`. (This is what `\istroot` requires.) Otherwise, you will get a compile *error*.

6.1.3 Decorating players or a whole simple tree using `\istrooto`

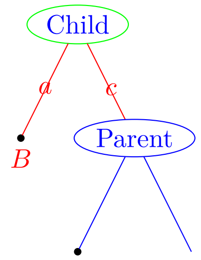
The oval version `\istrooto` puts an owner whithin a bubble type node. With `\istrooto`, the color of a player can be changed by the second bracket `[option]`, such as `[red]` or `[blue]`. (This is the Ti*k*Z way for giving options for a `node`.)

```
% Example: decorating players (with \istrooto)
\begin{istgame}
\istrooto(0)[red]{Child}
  \istb{a}{B}[below]  \istb{c}  \endist
\istrooto(1)(0-2)[blue,dashed]{Parent}
  \istb  \istb  \endist
\end{istgame}
```



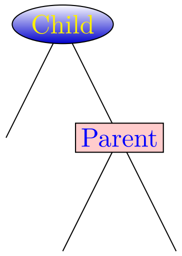
The following example shows how you can color, with `\istrooto`, a whole simple tree including action labels and payoffs in `red` or `blue`, except for node styles.

```
% Example: decorating players (with \istrooto)
\begin{istgame}
\istrooto[south,red](0)[draw=green,blue]{Child}
  \istb*{a}{B}[below]  \istb{c}  \endist
\istrooto[-90,blue](1)(0-2)[draw=blue,blue]{Parent}
  \istb*  \istb  \endist
\end{istgame}
```



What if you want to paint some color into the background of each oval node or box node? You can also do this simply by using the Ti*k*Z way of giving options for nodes.

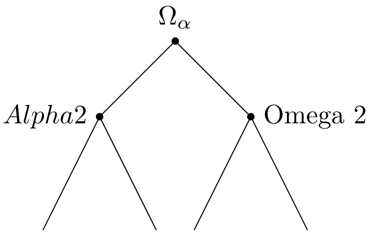
```
% Example: decorating oval/box nodes
\begin{istgame}
\istrooto(0)[top color=white,
             bottom color=blue!80!black,yellow]{Child}
  \istb  \istb  \endist
\istrooto(1)(0-2)[box node,fill=red!20,blue]{Parent}
  \istb  \istb  \endist
\end{istgame}
```



6.1.4 Changing the input mode and text font style: `\setistmathTF(*)`

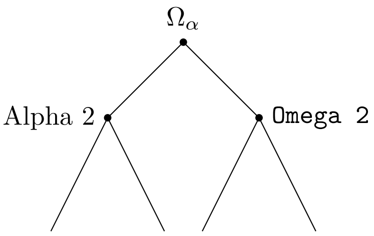
You can change the input mode for owners from text mode (by default) to math mode by using the macro `\setistmathTF111` which has three mandatory arguments. The first number `1` means that an owner is printed in *math mode*. (The second number `1` and third number `1` mean that action labels and payoffs are in math mode, respectively.) `\setistmath011` means that an owner is printed in text mode (this is default).

```
% Example: \setistmathTF
\begin{istgame}
\setistmathTF111   % owner: math mode
\istroot(0){\Omega_{\alpha}}+10mm..20mm+
                               \istb \istb \endist
\istroot(1)(0-1)<180>{Alpha 2} \istb \istb \endist
\setistmathTF011   % owner: text mode
\istroot(2)(0-2)<0>{Omega 2}   \istb \istb \endist
\end{istgame}
```



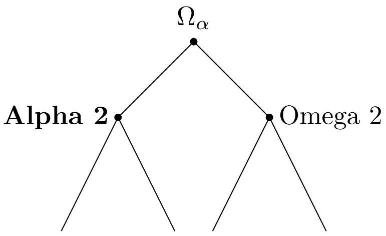
The starred version, for example, `\setistmathTF*011`, enables you to change the font style of an owner. Moreover, by specifying a text font style, like `\setistmathTF*011<texttt>`, in angle brackets, you can print an owner in typewriter font. (See Section 7 on page 36, for more details.)

```
% Example: \setistmathTF(*)
\begin{istgame}
\setistmathTF111          % owner: math mode
\istroot(0){\Omega_{\alpha}}+10mm..20mm+
                               \istb  \istb  \endist
\setistmathTF011          % owner: text mode
\istroot(1)(0-1)<180>{Alpha 2} \istb \istb \endist
\setistmathTF*011<texttt> % owner: in texttt
\istroot(2)(0-2)<0>{Omega 2}   \istb \istb \endist
\end{istgame}
```



The supplementary macro `\xtOwner` gives an alternative way of putting, outside of a simple tree, an owner in *text mode* at a node. The starred version `\xtOwner*` prints an owner as in the input mode set by `\setistmathTF*`. (See Section 13.1 on page 78, for more details on `\xtOwner`.)

```
% Example: \setistmathTF(*)
\begin{istgame}
\setistmathTF111 % owner: math mode
\istroot(0){\Omega_{\alpha}}+10mm..20mm+
                  \istb  \istb  \endist
\setistmathTF*011<textbf>   % owner: text mode
\istroot(1)(0-1) \istb \istb \endist
\istroot(2)(0-2) \istb \istb \endist
\xtOwner*(1){Alpha 2}[left] % owner: textbf
\xtOwner(2){Omega 2}[r]     % owner: normal font
\end{istgame}
```

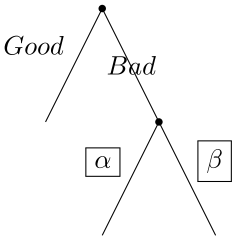


6.2 How to put action labels

6.2.1 Action labels: basics

The macro `\istb` prints a branch and its action label. Note that action labels should be input in *math mode*, by default.

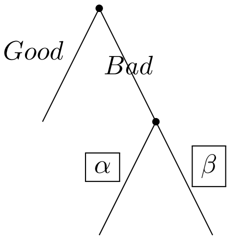
```
% Example: action labels
\begin{istgame}
\istroot(0)
  \istb{Good}[above left]  \istb{Bad}  \endist
\istroot(1)(0-2)
  \istb{\text{\fbox{$\alpha$}}}[above left]
  \istb{\text{\fbox{$\beta$}}}[above right]
  \endist
\end{istgame}
```



By default, `\istb` prints its action label on the midpoint of the corresponding branch. You can specify the position of an action label with the positional words and their abbreviations but not by the compass directions or degrees.

In the following example, the abbreviations are used to place action labels.

```
% Example: action labels with abbreviations
\begin{istgame}
\istroot(0)
  \istb{Good}[al]  \istb{Bad}  \endist
\istroot(1)(0-2)
  \istb{\text{\fbox{$\alpha$}}}[al]
  \istb{\text{\fbox{$\beta$}}}[ar]
  \endist
\end{istgame}
```



Notice that with the *abbreviations* the position of an action label is (internally) adjusted to get better result (for more details, see Section 8.2.2 on page 40).

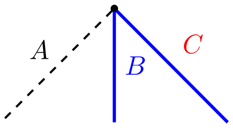
Remark:

- The supplementary macro `\xtActionLabel` is provided for an extra way of putting an action label (see Section 13.1 on page 78, for more details).
- By default, action labels are to be input in *math mode*. You can change the input mode to text mode by using `\setistmathTF` or `\setistmathTF*` (see below in Section 6.2.5 and Section 7 on page 36).

6.2.2 Decorating action labels

You can change the color of action labels with TikZ options. In the example below, the bracket options before an action label are for branches and those after are only for the action labels.

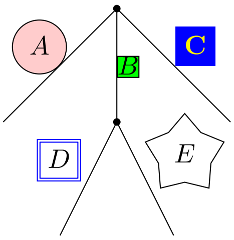
```
% Example: color and line style
\begin{istgame}
\istroot(0)
  \istb[dashed,thick]{A}[al]
  \istb[blue,very thick]{B}[r]
  \istb[blue,very thick]{C}[above right,red]
  \endist
\end{istgame}
```



Remark: It is *important* to remember that you cannot use the abbreviations with additional options. For example, you can do like `\itsb{C}[ar]` but not like `\istb{B}[ar,red]`. Instead, you should do like `\istb{C}[above right,red]`.

You can also express action labels in a box or a circle, or other shapes with colors.

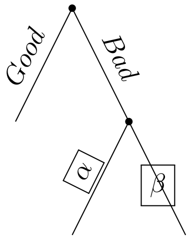
```
% Example: decorating action labels
\begin{istgame}
\istroot(0)
  \istb{A}[above left,draw=black,circle,fill=red!20]
  \istb{B}[right,draw=black,fill=green,inner sep=0pt]
  \istb{\textbf{C}}[above right,fill=blue,yellow]
  \endist
\istroot(1)(0-2)
  \istb{D}[above left,xshift=-3pt,draw=blue,double]
  \istb{E}[above right,xshift=5pt,draw,star]
  \endist
\end{istgame}
```



6.2.3 Sloped action labels

You can print sloped labels for actions by using the Ti*k*Z option `sloped`. Still you should remember that you cannot use the abbreviations to place action labels with other options like `sloped`.

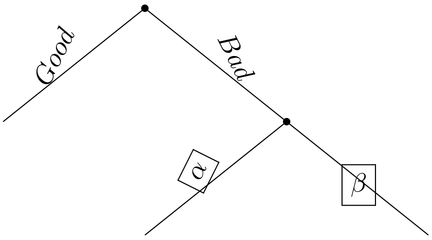
```
\begin{istgame}
\istroot(0)
  \istb{Good}[above,sloped]
  \istb{Bad}[above,sloped]      \endist
\istroot(1)(0-2)
  \istb{\text{\fbox{$\alpha$}}}[above,sloped]
  \istb{\text{\fbox{$\beta$}}}  \endist
\end{istgame}
```



Warning: Issues in *sloped labels* in Ti*k*Z with *asymmetric scales*:

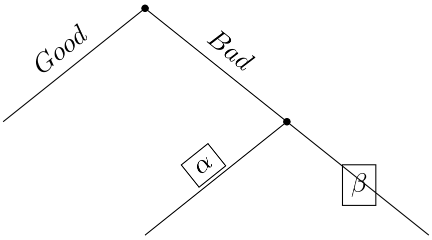
- The `tree` library in Ti*k*Z does not seem to treat sloped labels properly, when `xscale` or `yscale` is used asymmetrically.

```
\begin{istgame}[xscale=2.5]
\istroot(0)
  \istb{Good}[above,sloped]
  \istb{Bad}[above,sloped]      \endist
\istroot(1)(0-2)
  \istb{\text{\fbox{$\alpha$}}}[above,sloped]
  \istb{\text{\fbox{$\beta$}}}  \endist
\end{istgame}
```



- A temporary solution to cure this issue is to declare the macro `\xtcureslopedlabelsNS`, provided with this package, in a TeX group with caution. For a tree growing eastwards or westwards, use `\xtcureslopelabelsEW`, instead. (*These solutions are only temporary. They have not been tested for every occasion.*)

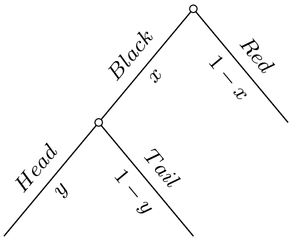
```
\begin{istgame}[xscale=2.5]
\xtcureslopedlabelsNS
\istroot(0)
  \istb{Good}[above,sloped]
  \istb{Bad}[above,sloped]      \endist
\istroot(1)(0-2)
  \istb{\text{\fbox{$\alpha$}}}[above,sloped]
  \istb{\text{\fbox{$\beta$}}}  \endist
\end{istgame}
```



6.2.4 Dual action labels

You can also express dual labels for actions by using `\istB` or `\istBt`.

```
\begin{istgame}[sloped,font=\footnotesize]
\xtdistance{15mm}{25mm}
\istroot(0)[chance node]
  \istB{Black}[a]{x}[b]
  \istB{Red}[a]{1-x}[b]    \endist
\istroot(1)(0-1)[chance node]
  \istBt{Head}[a]{y}[b]
  \istBt{Tail}[a]{1-y}[b]  \endist
\end{istgame}
```



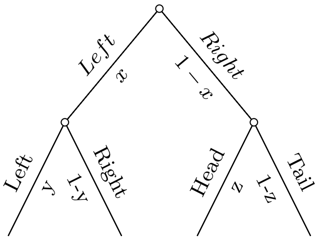
You can do the same thing as `\istB` does by using `\istb` together with the supplementary macro `\xtActionLabel` (see Section 5.4.2 for more details on dual action labels).

6.2.5 Changing the input mode and text font style: `\setistmathTF(*)`

By default the input mode for action labels is in *math mode*. With `\setistmathTF`, you can change the input mode for labels. The package's default input mode is set as `\setistmathTF011`, meaning that the input mode for an owner is in text mode (denoted by the first `0`), for action labels in math mode (denoted by the second number `1`), and for payoffs in math mode (denoted by the third number `1`).

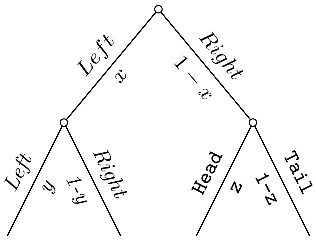
So if do like `\setistmathTF001` (with the second number `0`), action labels are in *text mode*.

```
% Example: \setistmathTF001
\begin{istgame}[sloped,font=\footnotesize]
\istroot(0)[chance node]+15mm..25mm+
  \istB{Left}[a]{x}[b] \istB{Right}[a]{1-x}[b] \endist
\setistmathTF001         % action labels: in text mode
\istroot(1)(0-1)[chance node]
  \istBt{Left}[a]{y}[b] \istBt{Right}[a]{1-y}[b] \endist
\istroot(2)(0-2)[chance node]
  \istBt{Head}[a]{z}[b] \istBt{Tail}[a]{1-z}[b] \endist
\end{istgame}
```



Moreover, if you use the starred vesrion, like `\setistmathTF*001`, action labels automatically are printed in *italics*. If you specify a font style in *curly braces*, like `\setistmathTF*001{texttt}`, you can even print action labels in typewriter font. For more details, see Section 7 on page 36.

```
% Example: \setistmathTF*001 (in italics)
\begin{istgame}[sloped,font=\footnotesize]
\istroot(0)[chance node]+15mm..25mm+
  \istB{Left}[a]{x}[b] \istB{Right}[a]{1-x}[b] \endist
\setistmathTF*001         % action labels: in italics
\istroot(1)(0-1)[chance node]
  \istB{Left}[a]{y}[b] \istB{Right}[a]{1-y}[b] \endist
\setistmathTF*001{texttt} % action labels: in texttt
\istroot(2)(0-2)[chance node]
  \istB{Head}[a]{z}[b] \istB{Tail}[a]{1-z}[b] \endist
\end{istgame}
```



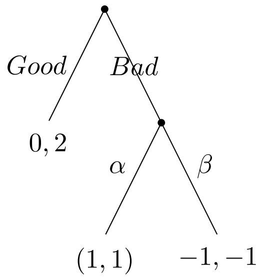
The supplementary macro `\xtActionLabel` prints, from outside of a simple tree, an action label in *math mode*. The starred version `\xtActionLabel*` prints an action label in the input mode as set by `\setistmathTF*`. For more details, see Section 5.4.2 (on page 25) and Section 7 (on page 36).

6.3 How to put payoffs

6.3.1 Payoffs: basics

The macro `\istb` can print a branch and also the corresponding payoffs, in *math mode* by default, near at its endpoint. The payoffs are put in the direction set by `\setistgrowdirection` (`south`, by default), unless it is changed by `<grow keyval>` of `\istroot`.

```
% Example: payoffs
\begin{istgame}
\istroot(0)
  \istb{Good}[l]{0,2}
  \istb{Bad}
  \endist
\istroot(1)(0-2)
  \istb{\alpha}[al]{(1,1)}
  \istb{\beta}[ar]{-1,-1}
  \endist
\end{istgame}
```



Remark:

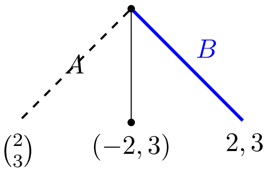
- The supplementary macro `\xtPayoff` is provided for an extra way of putting payoffs (see Section 13.1 on page 78).
- The payoffs are to be input in math mode, by default, you change the input mode to text mode by using `\setistmathTF(*)` (see Section 7 on page 36).

6.3.2 Payoffs and `\istgrowdirection`

The direction of where payoffs are put from a terminal node follows `\istgrowdirection` typed in as the first optional argument of `\istroot`. The default direction is `south` and can be changed by `\setistgrowdirection`. For example, `\setistgrwodirection{north}` changes the default direction to `north`.

To specify the tree growing direction or the position of payoffs to be put, you can use degrees, the compass directions, or the positional words and their *abbreviations*.

```
% Example: grow=south (default)
\begin{istgame}
\istroot(0)
  \istb[dashed,thick]{A}{\binom23}
  \istb*{}{(-2,3)}
  \istbt[blue,very thick]{B}[ar]{2,3}
  \endist
\end{istgame}
```

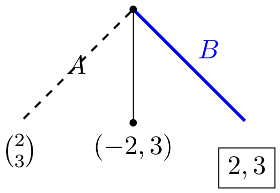


By default, `grow=south`, so `\istgrowdirection` is `south` (or `below` or `-90`). The above example shows payoffs at the south (by default) of terminal nodes.

Remark:

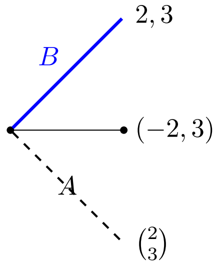
- You can simply omit the position of payoffs. Then they are printed, by default, in the direction as set by `\istgrowdirection`.
- However, if you use options other than the direction of payoffs, you must specify the direction with others.
- Moreover, the direction *must* be the *first* entry of the option list for payoffs and the abbreviations cannot be used. For example, `[b,yshift=-3mm]` and `[yshift=-3mm,below]` are not accetable. It must be that `[below,yshift=-3mm]`.
- Note also that you can do like `[[yshift=-3mm]below`, instead (see also page 41).

```
% Example: grow=south (default)
\begin{istgame}
\istroot(0)
  \istb[dashed,thick]{A}{\binom23}
  \istb*{}{(-2,3)}
  \istbt[blue,very thick]{B}[ar]{2,3}
        [below,yshift=-3mm,draw]
  \endist
\end{istgame}
```



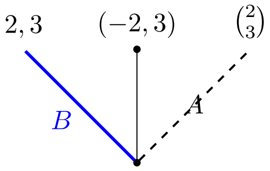
You can see more examples, below, that show the positions of payoffs, by default, depend on the tree growing directions, unless you specify different directions.

```
% Example: grow=right (or east or 0 degree)
\begin{istgame}
\istroot[right](0)
  \istb[dashed,thick]{A}{\binom23}
  \istb*{}{(-2,3)}
  \istbt[blue,very thick]{B}[al]{2,3}
  \endist
\xtHideTerminalNodes
\end{istgame}
```



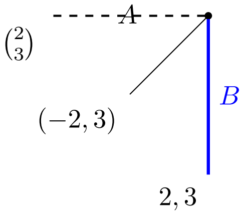
`grow=right=\istgrowdirection`, so payoffs are put on the right.

```
% Example: grow=north
\begin{istgame}
\setistgrowdirection{north}
\istroot(0)
  \istb[dashed,thick]{A}{\binom23}
  \istb*{}{(-2,3)}
  \istbt[blue,very thick]{B}[bl]{2,3}
  \endist
\end{istgame}
```



`grow=north=\istgrowdirection`, so payoffs are put above the terminal nodes.

```
% Example: grow=south west
\begin{istgame}
\istroot[south west](0)
  \istb[dashed,thick]{A}{\binom23}
  \istb{}{(-2,3)}
  \istb[blue,very thick]{B}[right]{2,3}
  \endist
\end{istgame}
```



`grow=south west=\istgrowdirection`, so payoffs are put below left of the terminal node.

You can adjust the direction of putting payoffs by specifying a directional word right after payoffs, like `\istb[blue,very thick]{B}[right]{2,3}[below]`.

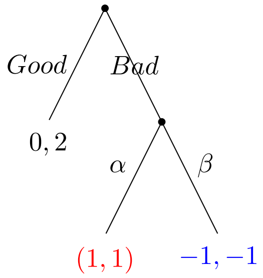
You can use the abbreviations `[l]`, `[r]`, `[a]`, and `[b]` for `[left]`, `[right]`, `[above]`, and `[below]`, respectively. The abbreviations `[al]`, `[ar]`, `[bl]`, and `[br]` can also be used for `[above left]`, `[above right]`, `[below left]`, and `[below right]`, respectively, to put payoffs (for more details about abbreviations, see Section 8.2.2 on page 40).

Notice also that, instead of the positional words, you can use the compass directions or degrees, like `\istb[blue,very thick]{B}[right]{2,3}[-90]`.

6.3.3 Decorating payoffs

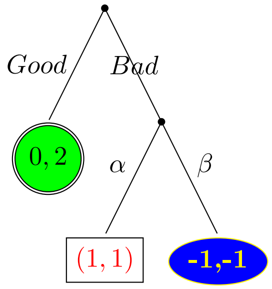
You can change the color of payoffs by giving TikZ options right before the positional words for payoffs. For example you can do like `\istb...{(1,1)}[[blue]below]`. Note that, in this case, you cannot use the abbreviation of the positional words.

```
% Example: coloring payoffs
\begin{istgame}
\istroot(0)
  \istb{Good}[l]{0,2}  \istb{Bad}  \endist
\istroot(1)(0-2)
  \istb{\alpha}[al]{(1,1)}[[red]below]
  \istb{\beta}[ar]{-1,-1}[[blue]below]
  \endist
\end{istgame}
```



You can also put payoffs in a box, a circle, or other shapes, even with color in the background.

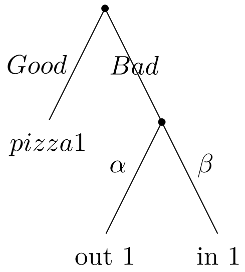
```
% Example: decorating payoffs
\begin{istgame}
\istroot(0)
  \istb{Good}[l]{0,2}[[draw,circle,double,fill=green]below]
  \istb{Bad}  \endist
\istroot(1)(0-2)
  \istb{\alpha}[al]{(1,1)}[[red,draw=black]below]
  \istb{\beta}[ar]{\textbf{-1,-1}}
       [[draw,ellipse,fill=blue,yellow,inner sep=2pt]below]
  \endist
\end{istgame}
```



6.3.4 Changing the input mode and text font style: `\setistmathTF(*)`

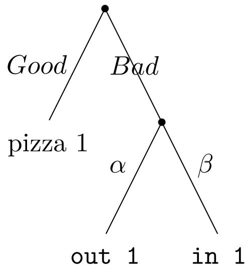
You can change the input mode for payoffs from math mode (by default) to text mode, by using `\setistmathTF010`. The last `0` means that payoffs are in text mode.

```
% Example: \setistmathTF
\begin{istgame}
\istroot(0)
  \istb{Good}[l]{pizza 1}  \istb{Bad}  \endist
\setistmathTF010 % payoffs: in text mode
\istroot(1)(0-2)
  \istb{\alpha}[al]{out 1}  \istb{\beta}[ar]{in 1}  \endist
\end{istgame}
```



The starred version, for example, `\setistmathTF*010[texttt]`, with a font style specified in brackets, prints payoffs in typewriter font.

```
% Example: \setistmathTF*
\begin{istgame}
\setistmathTF010 % payoffs: in text mode
\istroot(0)
  \istb{Good}[l]{pizza 1}  \istb{Bad}  \endist
\setistmathTF*010[texttt] % payoffs: in texttt
\istroot(1)(0-2)
  \istb{\alpha}[al]{out 1}  \istb{\beta}[ar]{in 1}  \endist
\end{istgame}
```



The supplementary macro `\xtPayoff` prints, outside of a simple tree, payoffs in *math mode*. The starred version `\xtPayoff*` prints payoffs in the input mode as set by `\setistmathTF*`. See Section 7 on page 36.

7 Input mode and text font style changer: `\setistmathTF(*)`

The macro `\setistmathTF` enables you to change the input mode for important labels: owners, action labels, and payoff. It takes three numbers (`0` or `1`) as *mandatory* arguments. Here, `1` (`true`) means the input mode is in *math mode* and `0` (`false`) in *text mode*. The three numbers represents the input mode for owners, action labels, and payoffs, respectively. In this package, it is initially set as `\setistmathTF{0}{1}{1}`, meaning that owners are to be input in text mode, and action labels and payoffs in math mode.

```
% \setistmathTF
% syntax:
  \setistmathTF{<owner input mode>}{<action input mode>}{<payoff input mode>}
% initial values:
  {0}{1}{1}
```

The starred version `\setistmathTF*` accepts (in addition to three *mandatory* numbers) three *optional* arguments, each of which is effective *only when* the corresponding input mode is in *text mode*. The arguments are to be one of text font styles *without a backslash* such as `textrm`, `textit`, `itshape`, `textbf`, `scriptsize`, `tiny`, and so on. Each of the three optional arguments is ignored when used with `\setistmathTF` and when the corresponding input mode is in math mode.

```
% \setistmathTF
% syntax:
  \setistmathTF{}{}{}<owner font>{<action label font>}[<payoff font>]
% defaults
  {0}{1}{1}<>{textit}[]
```

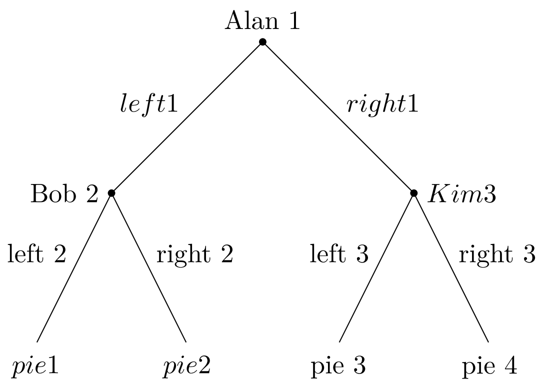
The first optional argument in *angle brackets* is the font style for an owner, the second in *curly braces* for action labels, and the third *in brackets* for payoffs. By default action labels are printed in *italics*.

For example, with `\setistmathTF*000<texttt>{textit}[tiny]` delared, an owner is printed in typewriter font, action labels in italics, and payoffs in the tiny size of normal text font (roman, upright). With `\setistmathTF000<texttt>{textit}[tiny]`, all the optional arguments are ignored and all the labels are printed in normal text font. Note also that with `\setistmathTF*001`, you can print action labels in *italics* (by default).

7.1 `\setistmathTF`: input mode changer

The macro `\setistmathTF` is an *input mode changer*, taking three numbers as mandatory arguments. The default input modes for important labels are set as `\setistmathTF011`.

```
% Example: \setistmathTF
\begin{istgame}
%\setistmathTF011 % (default mode)
\xtdistance{20mm}{20mm}
\istroot(0){Alan 1}+20mm..40mm+
  \istb{left 1}[al]
  \istb{right 1}[ar]         \endist
\setistmathTF001 % mode: text,text,math
\istroot(1)(0-1)<180>{Bob 2}
  \istb{left 2}[al]{pie 1}
  \istb{right 2}[ar]{pie 2}  \endist
\setistmathTF100 % mode: math,text,text
\istroot(2)(0-2)<0>{Kim 3}
  \istb{left 3}[al]{pie 3}
  \istb{right 3}[ar]{pie 4}  \endist
\end{istgame}
```



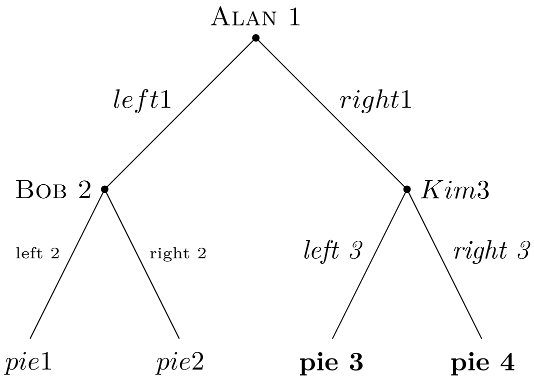
7.2 `\setistmathTF*`: input mode and text font style changer

The starred version `\setistmathTF*` is a *text font style changer* as well as an *input mode changer*. It takes three numbers as *mandatory* arguments followed by three *optional* arguments in the order of `<>{}[]`. The first option `<>` is for an owner, the second `{}` for action labels, and the third `[]` for payoffs.

Each optional argument should be one of valid font shapes and sizes, but without a textbackslash (not as a command), such as `textbf`, `textsc`, `textit`, `large`, `tiny`, and so on. Each of the options is effective only when it is used with the starred version `\setistmathTF*` and the corresponding input mode is in text mode.

Remark: Note that if the second optional argument is omitted when action labels are in text mode, for example, like `\setistmathTF*001`, *action labels* are printed in *italics* by default.

```
% Example: \setistmathTF*
\begin{istgame}
\xtdistance{20mm}{20mm}
\setistmathTF*011<textsc>{tiny}[textbf]
\istroot(0){Alan 1}+20mm..40mm+
  \istb{left 1}[al]
  \istb{right 1}[ar]         \endist
\setistmathTF*001<textsc>{tiny}[textbf]
\istroot(1)(0-1)<180>{Bob 2}
  \istb{left 2}[al]{pie 1}
  \istb{right 2}[ar]{pie 2}  \endist
\setistmathTF*100[textbf] % actions labels
    in italics (by default)
\istroot(2)(0-2)<0>{Kim 3}
  \istb{left 3}[al]{pie 3}
  \istb{right 3}[ar]{pie 4}  \endist
\end{istgame}
```

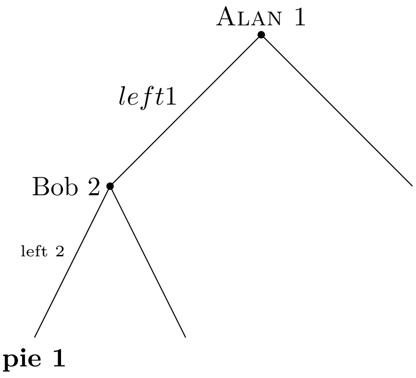


7.3 `\setistmathTF*` and supplementary macros `\xtFoo*` printing labels

Many supplementary macros, working outside of a simple tree, optionally print important labels: owners, action labels, and payoffs. These include `\xtOwner`, `\xtActionLabel`, `\xtPayoff`, `\xtInfosetOwner` and many more. All of the macros for information sets (except `\cntmAInfoset` and `\cntmAInfosetO`) such as `\xtInfoset` or `\xtCInfosetO` also print the owners of information sets. With any supplementary macros, by default, an owner is printed in text mode and action labels and payoffs are printed in math mode.

All of these macros have their own *starred* (`*`) versions, which print the labels in the input mode as set by `\setistmathTF*` (but not by `\setistmathTF`). For example, `\xtActionLabel*` prints action labels in *italics* with `\setistmathTF*001`, while `\xtActionLabel` prints action labels in *math mode*.

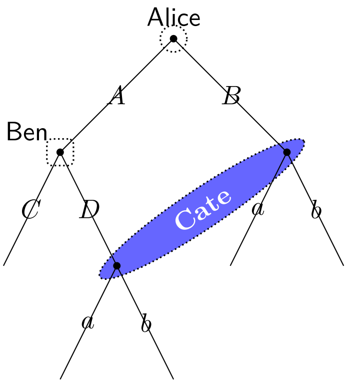
```
% Example: \setistmathTF*
\begin{istgame}
\xtdistance{20mm}{20mm}
\setistmathTF*000<textsc>{tiny}[textbf]
\istroot(0)+20mm..40mm+ \istb \istb \endist
\istroot(1)(0-1)        \istb \istb \endist
\xtOwner*(0){Alan 1}  \xtOwner(1){Bob 2}[l]
\xtActionLabel(0)(0-1){left 1}[al]
\xtActionLabel*(1)(1-1){left 2}[al]
\xtPayoff*(1-1){pie 1}
\end{istgame}
```



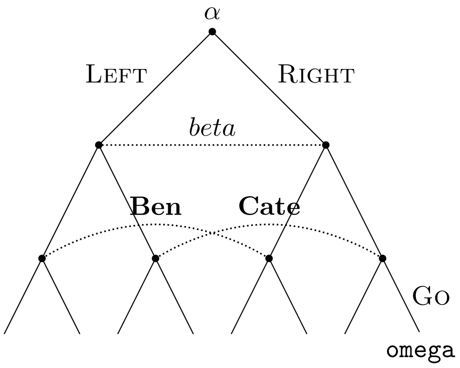
You can find more examples for supplementary macros in this regard that are scattered here and there throughout the manual in appropriate places. See Section 6.1.4 Section 6.2.5 Section 6.3.4.

Here are some more examples on owners of information sets (see Section 10 for more details on information sets).

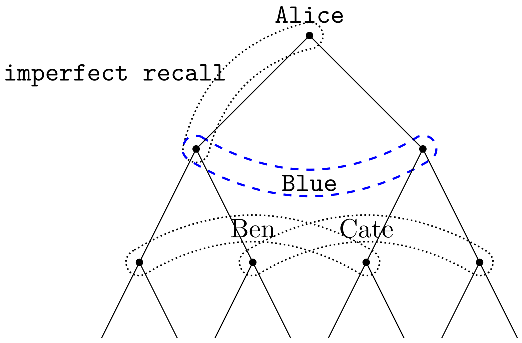
```
\begin{istgame}
\setistmathTF*011<textsf>
\istroot(0){Alice}+15mm..30mm+  \istb{A} \istb{B} \endist
\istroot(1)(0-1)<135>{Ben}  \istb{C} \istb{D} \endist
\istroot(2)(0-2)   \istb{a} \istb{b} \endist
\istroot(3)(1-2)   \istb{a} \istb{b} \endist
\setistmathTF*011<textbf>
\xtInfosetO(0)(0)
\xtInfosetO[rectangle,rounded corners=.2em](1)(1)
\xtInfosetO*[ellipse,fill=blue!60]
           (3)(2){Cate}[sloped,white](1.5em)
\end{istgame}
```



```
\begin{istgame}
\setistmathTF*100{textsc}[texttt]
\istroot(0){\alpha}+15mm..30mm+
  \istb{Left}[al] \istb{Right}[ar] \endist
\istroot(1)(0-1)  \istb \istb \endist
\istroot(2)(0-2)  \istb \istb \endist
\xtdistance{10mm}{10mm}
\istroot(3)(1-1)  \istb \istb \endist
\istroot(4)(1-2)  \istb \istb \endist
\istroot(5)(2-1)  \istb \istb \endist
\istroot(6)(2-2)
  \istb \istb{Go}[r]{omega} \endist
\xtInfoset*(1)(2)
\xtInfosetOwner*(1)(2){beta}[a]
\setistmathTF*011<textbf>
\xtCInfoset*(1-1)(2-1){Ben}
\xtCInfoset*(1-2)(2-2){Cate}
\end{istgame}
```



```
\begin{istgame}
\setistmathTF*011<texttt>
\istroot(0){Alice}+15mm..30mm+
  \istb \istb \endist
\istroot(1)(0-1)  \istb \istb \endist
\istroot(2)(0-2)  \istb \istb \endist
\xtdistance{10mm}{10mm}
\istroot(3)(1-1)  \istb \istb \endist
\istroot(4)(1-2)  \istb \istb \endist
\istroot(5)(2-1)  \istb \istb \endist
\istroot(6)(2-2)  \istb \istb \endist
\xtCInfosetO*(1)(0){imperfect recall}[left]
\xtCInfosetO*[dashed,blue,thick](1)(2)<.7>{Blue}
\xtCInfosetO(1-1)(2-1){Ben}
\xtCInfosetO(1-2)(2-2){Cate}
\end{istgame}
```



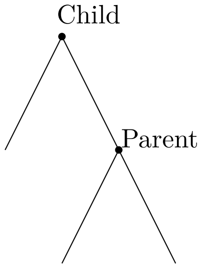
8 Fine-tuning positions of players, action labels, and payoffs (experimental)

8.1 Fine-tuning positions: owners

If you are not satisfied the position of an owner (or a player), you can change it by using the Ti*k*Z options such as `xshift`, `yshift`, or `label distance` with the angle option of `\istroot`.

Examples are `<[xshift=10pt]above>{Child}` and `<[label distance=-5pt]45>{Parent}`, as shown in the following:

```
% Example: node owner
\begin{istgame}
\istroot(0)<[xshift=10pt]above>{Child}
  \istb
  \istb
  \endist
\istroot(1)(0-2)<[label distance=-5pt]45>{Parent}
  \istb
  \istb
  \endist
\end{istgame}
```



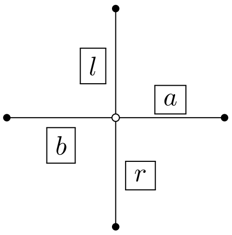
8.2 Fine-tuning positions: action labels

8.2.1 Abbreviations: [l], [r], [a], and [b]

As discussed in 5.1 on page 18, the macro `\istb` deals with the action labels.

In order to determine the direction of action labels for branches to put, you can use degrees, the compass directions, or the positional words and their abbreviations as mentioned above. (Internally, the abbreviations for payoffs and those for action labels work slightly differently in terms of `xshift` and `yshift`.)

```
% Example: action labels (default position)
\begin{istgame}[scale=1.2]
\xtShowEndPoints
\xtdistance{12mm}{16mm}
\istroot(0)[initial node]
  \istb<grow=0>{\fbox{$a$}}[a]   \istb<grow=90>{\fbox{$l$}}[l]
  \istb<grow=180>{\fbox{$b$}}[b] \istb<grow=-90>{\fbox{$r$}}[r]
  \endist
\end{istgame}
```

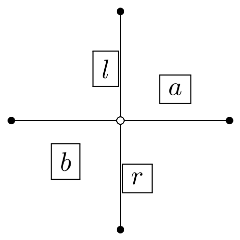


When you use these abbreviations you can manipulate the horizontal and/or the vertical shifts toward branches by using `\xtALPush`. (This is experimental!)

```
% syntax:
  \xtALPush{<xshift dim> for l and r}{<yshift dim> for a and b}
% default:
  {0pt}{0pt}
```

For example, `\xtALPush{-3pt}{5pt}` draws the labels left and right by `3pt` to the branch and push those put above and below `5pt` away from the branch, as shown in the following:

```
% Example: \xtALPush
\begin{istgame}[scale=1.2]
\xtShowEndPoints
\xtALPush{-3pt}{5pt} % look here
\xtdistance{12mm}{16mm}
\istroot(0)[initial node]
  \istb<grow=0>{\fbox{$a$}}[a]    \istb<grow=90>{\fbox{$l$}}[l]
  \istb<grow=180>{\fbox{$b$}}[b]  \istb<grow=-90>{\fbox{$r$}}[r]
  \endist
\end{istgame}
```



8.2.2 Abbreviations: `[al]`, `[ar]`, `[bl]`, and `[br]`

You can also use the abbreviations `al`, `ar`, `bl`, and `br` to represent `above left`, `above right`, `below left`, and `below right`, respectively, to position action labels. Precise representation of abbreviations is as follows:

- `[al]` represents `[above left,xshift=1pt,yshift=-2pt,black]`
- `[ar]` represents `[above right,xshift=-1pt,yshift=-2pt,black]`
- `[bl]` represents `[below left,xshift=1pt,yshift=2pt,black]`
- `[br]` represents `[below right,xshift=-1pt,yshift=2pt,black]`

```
%\begin{tcblisting}{listing above text}
% Example: action labels with abbreviations
\begin{istgame}[scale=1.1]
\xtShowTerminalNodes[oval node,line width=.3pt]
\def\xbox#1{\fbox{$#1$}}  \def\ybox#1{#1}
\xtdistance{12mm}{16mm}
\istroot(0)[initial node]
  \istb<grow=0>{\ybox{a}}[a]
  \istb<grow=90>{\ybox{l}}[l]
  \istb<grow=180>{\ybox{b}}[b]
  \istb<grow=-90>{\ybox{r}}[r]
  \endist
\begin{scope}[line width=1.2]
\xtdistance{10mm}{20mm}
\istroot[90](N)(0-2)
  \istbt[red]{\xbox{br}}[br]{1,1}
  \istbt[blue,dashed]{\xbox{bl}}[bl]{2,2}
  \endist
\istroot[180](W)(0-3)
  \istbt[green]{\xbox{ar}}[ar]{0,3}
  \istbt[red]{\xbox{br}}[br]{2,1}
  \endist
\istroot[-90](S)(0-4)
  \istbt[dotted]{\xbox{al}}[al]{1,-1}
  \istbt[green]{\xbox{ar}}[ar]{-1,1}
  \endist
\istroot[0](E)(0-1)
  \istbt[blue,dashed]{\xbox{bl}}[bl]{1,3}
  \istbt[dotted]{\xbox{al}}[al]{2,0}
  \endist
\foreach \x in {1,...,4} {\xtNode(0-\x)}
\end{scope}
\end{istgame}
```

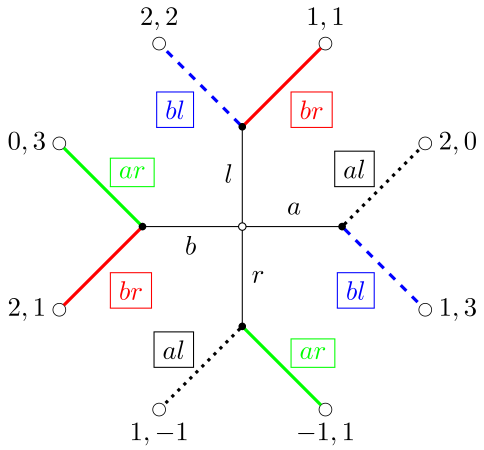


Figure 1: positioning action labels with the abbreviations

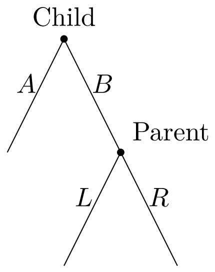
In Figure 1, $\boxed{al}$'s are put in the same position from dotted branches, and so are the other labels from their corresponding branches, like $\boxed{bl}$'s from the blue dashed branches.

You can also use `\xtALShift` to put and push labels horizontally and vertically. (This is experimental!)

```
% syntax:
  \xtALShift{<horizontal shift dim>}{<vertical shift dim>}
% defaults:
  {1pt}{2pt}
```

When the dimensions get bigger than the defaults (`1pt` and `2pt`) the labels get closer to the midpoints of the corresponding branches, and when the numbers get smaller the labels get farther from their branches.

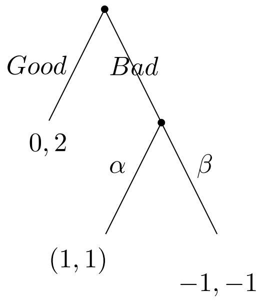
```
% Example: node owner
\begin{istgame}
\xtALShift{4pt}{3pt}
\istroot(0)<above>{Child} % default: <above>
  \istb{A}[al]  \istb{B}[ar]  \endist
\istroot(1)(0-2)<45>{Parent}
  \istb{L}[al]  \istb{R}[ar]  \endist
\end{istgame}
```



8.3 Fine-tuning positions: payoffs

You can change the position of payoffs with Ti*k*Z options: `xshift` and `yshift`. For example, you can do as shown in the following:

```
% Example: payoffs
\begin{istgame}
\istroot(0)
  \istb{Good}[l]{0,2}  \istb{Bad}  \endist
\istroot(1)(0-2)
  \istb{\alpha}[al]{(1,1)}[[xshift=-10pt]below]
  \istb{\beta}[ar]{-1,-1}[[yshift=-10pt]below]
  \endist
\end{istgame}
```



9 Growing direction of trees

You can draw a game tree that grows in any direction. By default, a game tree grows down and the `child` branches are arranged and named counterclockwise with respect to their parent node. When a tree grows down, the branches are arranged from left to right. When a tree grows to the right, the branches are arranged and named from below to above.

Sometimes you may want a tree with the branches arranged clockwise with respect to their parent node because it seems to look more natural, especially when a tree grows north or east. In Ti*k*Z, `grow'=<direction>` enables you to draw a tree developed clockwise.

To deal with the direction of the tree growth and the order of arranging branches, this package provides `\setistgrowdirection'` as well as `\setistgrowdirection`.

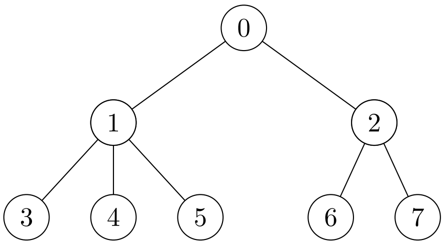
```
% default: growing south counterclockwise
  \def\xtgrow{grow}
  \def\istdefault@grow{south} % tree growing direction

% \setistgrowdirection(')
  \NewDocumentCommand\setistgrowdirection{t'm}
  {\IfBooleanTF {#1}
    { \renewcommand\xtgrow{grow'}
      \renewcommand\istdefault@grow{#2}
    }
    { \renewcommand\xtgrow{grow}
      \renewcommand\istdefault@grow{#2}
    }
  }
```

9.1 `\setistgrowdirection` – counterclockwise

Our first example is a tree drawn using the default values: growing `south` with branches going `counterclockwise` with respect to their parent nodes (from left to right).

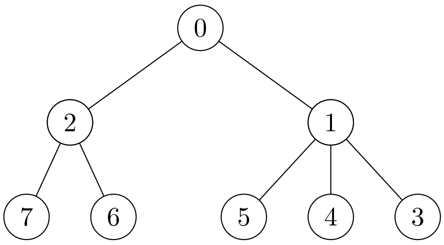
```
% Example 1: \setistgrowdirection{south}
\begin{istgame}
\setistOvalNodeStyle{.6cm}
\xtShowEndPoints[oval node]
\istrooto(0){0}+{12.5mm}..{3.45cm}+
  \istb  \istb  \endist
\xtdistance{12.5mm}{11.5mm}
\istrooto(1)(0-1){1}
  \istb{}{3}[center]  \istb{}{4}[center]
  \istb{}{5}[center]  \endist
\istrooto(2)(0-2){2}
  \istb{}{6}[center]  \istb{}{7}[center]  \endist
\end{istgame}
```



By default or with `\setistgrowdirection{south}`, the numbers written in the child nodes increase counterclockwise with respect to their parent nodes.

9.2 `\setistgrowdirection'` – clockwise

```
% Example 2: \setistgrowdirection'{south}
\begin{istgame}
\setistgrowdirection'{south}
% same codes as in Example 1
\setistOvalNodeStyle{.6cm}
\xtShowEndPoints[oval node]
\istrooto(0){0}+{12.5mm}..{3.45cm}+
  \istb  \istb  \endist
\xtdistance{12.5mm}{11.5mm}
\istrooto(1)(0-1){1}
  \istb{}{3}[center]  \istb{}{4}[center]
  \istb{}{5}[center]  \endist
\istrooto(2)(0-2){2}
  \istb{}{6}[center]  \istb{}{7}[center]  \endist
\end{istgame}
```



With `\setistgrowdirection'{south}`, the numbers written in the child nodes increase `clockwise` with respect to their parent nodes, which does not look natural.

9.3 Examples of rotating trees with `\setistgrowdirection`

This macro allows you to to rotate a game tree.

When you rotate a game tree to the north or to the east, it is a good idea to use the swap version `\setistgrowdirection'`.

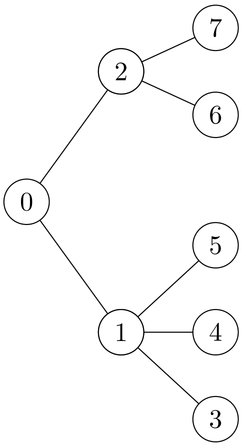
Tips: Though it is not necessary, it is suggested to use the following combinations of the macros and the directions.

`\setistgrowdirection{south}`: branches going counterclockwise (from left to right)
`\setistgrowdirection{west}`: branches going counterclockwise (downward)
`\setistgrowdirection'{north}`: branches going clockwise (from left to right)
`\setistgrowdirection'{east}`: branches going clockwise (downward)

Remark: When you use the swap version `\setistgrowdirection'`, using either `\istroot` or `\istroot'` gives you the same result.

9.3.1 A tree growing east – counterclockwise

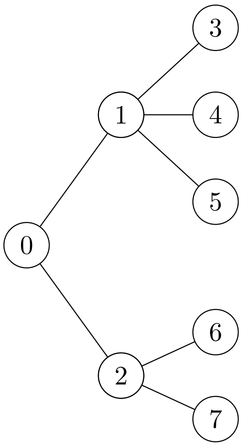
```
% Example 3: \setistgrowdirection{east}
\begin{istgame}
\setistgrowdirection{east}
% same codes as in Example 1
\setistOvalNodeStyle{.6cm}
\xtShowEndPoints[oval node]
\istrooto(0){0}+{12.5mm}..{3.45cm}+
  \istb  \istb  \endist
\xtdistance{12.5mm}{11.5mm}
\istrooto(1)(0-1){1}
  \istb{}{3}[center]  \istb{}{4}[center]
  \istb{}{5}[center]  \endist
\istrooto(2)(0-2){2}
  \istb{}{6}[center]  \istb{}{7}[center]
  \endist
\end{istgame}
```



Numbers increase upward (or counterclockwise).

9.3.2 A tree growing east – clockwise

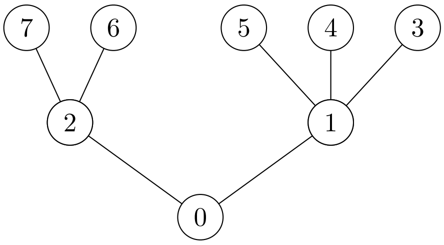
```
% Example 4: \setistgrowdirection'{east}
\begin{istgame}
\setistgrowdirection'{east}
% same codes as in Example 1
\setistOvalNodeStyle{.6cm}
\xtShowEndPoints[oval node]
\istrooto(0){0}+{12.5mm}..{3.45cm}+
  \istb  \istb  \endist
\xtdistance{12.5mm}{11.5mm}
\istrooto(1)(0-1){1}
  \istb{}{3}[center]  \istb{}{4}[center]
  \istb{}{5}[center]  \endist
\istrooto(2)(0-2){2}
  \istb{}{6}[center]  \istb{}{7}[center]
  \endist
\end{istgame}
```



Here, numbers increase downward (or clockwise). This looks more natural.

9.3.3 A tree growing north – counterclockwise

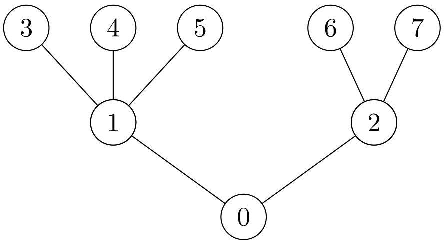
```
% Example 5: \setistgrowdirection{north}
\begin{istgame}
\setistgrowdirection{north}
% same codes as in Example 1
\setistOvalNodeStyle{.6cm}
\xtShowEndPoints[oval node]
\istrooto(0){0}+{12.5mm}..{3.45cm}+
  \istb  \istb  \endist
\xtdistance{12.5mm}{11.5mm}
\istrooto(1)(0-1){1}
  \istb{}{3}[center]  \istb{}{4}[center]
  \istb{}{5}[center]  \endist
\istrooto(2)(0-2){2}
  \istb{}{6}[center]  \istb{}{7}[center]
  \endist
\end{istgame}
```



Numbers increase from right to left (or counterclockwise).

9.3.4 A tree growing north – clockwise

```
% Example 6: \setistgrowdirection'{north}
\begin{istgame}
\setistgrowdirection'{north}
\setistOvalNodeStyle{.6cm}
\xtShowEndPoints[oval node]
\xtdistance{12.5mm}{11.5mm}
\istrooto(0){0}+{12.5mm}..{3.45cm}+
  \istb  \istb  \endist
\xtdistance{12.5mm}{11.5mm}
\istrooto(1)(0-1){1}
  \istb{}{3}[center]  \istb{}{4}[center]
  \istb{}{5}[center]  \endist
\istrooto(2)(0-2){2}
  \istb{}{6}[center]  \istb{}{7}[center]
  \endist
\end{istgame}
```



Numbers increase from left to right (or clockwise). This looks more natural.

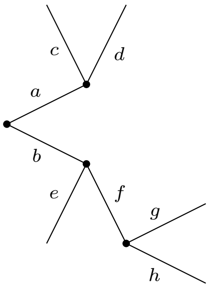
9.3.5 `\setxtgrowkey` for one simple tree

`\setxtgrowkey` can be used to change the key between `grow` and `grow'`, which is useful especially for one simple tree, not a whole tree.

```
% \setxtgrowkey: definition
\NewDocumentCommand\setxtgrowkey{m}
{
   \renewcommand\xtgrow{#1}
}
% #1 is either grow or grow'
```

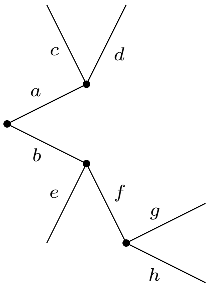
The example below shows that the branches are arranged clockwise by `\setistgrowdirection'`. So you will need to use `\setistgrow{grow}` to locally get back to *counterclockwise*.

```
% Example: using \setistgrowdirection' (swap version)
\begin{istgame}[scale=.7,font=\scriptsize]
\setistgrowdirection'{east}
\istroot(0)              \istb{a}[al] \istb{b}[bl] \endist
\istroot[north](1)(0-1)  \istb{c}[bl] \istb{d}[br] \endist
{\setxtgrowkey{grow}
\istroot[south](2)(0-2)  \istb{e}[al] \istb{f}[ar] \endist
}
\istroot'(3)(2-2)        \istb{g}[al] \istb{h}[bl] \endist
\end{istgame}
```



You can do the same thing by using `\setistgrowdirection` and `\istroot'`. See the following example, in which you do not need to use `\setxtgrowkey`.

```
% Example: using \istroot' (swap version)
\begin{istgame}[scale=.7,font=\scriptsize]
\setistgrowdirection{east}
\istroot'(0)             \istb{a}[al] \istb{b}[bl] \endist
\istroot'[north](1)(0-1) \istb{c}[bl] \istb{d}[br] \endist
\istroot[south](2)(0-2)  \istb{e}[al] \istb{f}[ar] \endist
\istroot'(3)(2-2)        \istb{g}[al] \istb{h}[bl] \endist
\end{istgame}
```



10 Information sets

10.1 `\xtInfoset`: standard version

The macro `\xtInfoset` draws an information set, connecting two decision nodes.

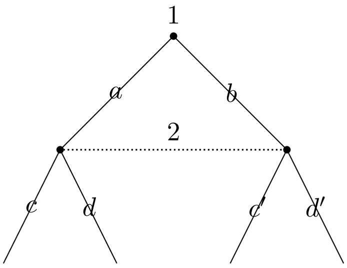
The starred version `\xtInfoset*` prints the owner of an information set in input mode and text font style as set by `\setistmathTF(*)` (see page 38 for examples).

```
% \xtInfoset
% syntax: from left to right
  \xtInfoset[<info line type>](<from>)(<to>){<owner>}[<pos>,<node opt>]
% defaults:
  [-,infoset style](<m>)(<m>){}[above]
% option style for (all) information sets
  infoset style={semithick,densely dotted}
```

The two coordinates `(<from>)` and `(<to>)` are mandatory and all other arguments are optional.

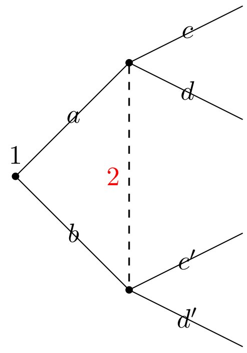
Remark: This package provides macros to draw various types of information sets. They all include an option style `[infoset style]`, which is equivalent to `[semithick,densely dotted]`. You can change the style or add more options to it by using the macro `\setxtinfosetstyle`. For more details, see Section 10.2.3 on page 48.

```
% Example: \xtInfoset
\begin{istgame}
\istroot(0){1}+15mm..30mm+
  \istb{a} \istb{b}   \endist
\istroot(1)(0-1)
  \istb{c} \istb{d}   \endist
\istroot(2)(0-2)
  \istb{c'} \istb{d'} \endist
\xtInfoset(1)(2){2}
\end{istgame}
```



When you specify the owner of an information set, it appears (by default) above the midpoint of the two nodes. To change the location of the information set owner, you can use the abbreviations of directional words like `a`, `l`, `ar`, or `bl`. However, when you use other options with the position you cannot use the abbreviations. If you want an owner in red on the left, you can do it like `[left,red]` as shown below.

```
\begin{istgame}
\setistgrowdirection'{east}
\setxtinfosetstyle{dashed} % changes line style
\istroot(0){1}+15mm..30mm+
  \istb{a}  \istb{b}  \endist
\istroot(1)(0-1)
  \istb{c}  \istb{d}  \endist
\istroot(2)(0-2)
  \istb{c'} \istb{d'} \endist
\xtInfoset(1)(2){2}[left,red]
\end{istgame}
```



For a curved information set, you can do, for example, like `\xtInfoset[bend left](1)(2)`. However this depends on the direction of tree growing or swapping the arrangement of branches. So, in order to draw a curved information set, it is recommended for you to use the curved version `\xtCInfoset`, documented in Section 10.3 on page 50.

10.2 `\xtInfosetO`: oval version

10.2.1 `\xtInfosetO`: basics

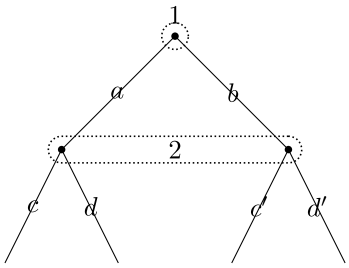
The oval version `\xtInfosetO` prints a bubble type (by default, a `rounded rectangle`) information set connecting two nodes, on the background layer by default. However, when you specify two identical nodes, it prints a densely dotted `circle` (by default) at the node to express a *singleton information set.*

Its starred version `\xtInfosetO*` prints the owner of an information set in input mode and text font style as set by `\setistmathTF(*)` (see page 38 for examples).

```
% \xtInfosetO (from left to right)
% syntax:
  \xtInfosetO[<bubble opt>](<from>)(<to>){<owner>}[<pos>,<owner opt>](min. height)
% defaults: connecting two nodes: \xtInfoset(coor1)(coor2)
  [ rectangle,rounded corners=.5*<minimum height>*<\xtscale>,
    minimum width=\n1+<minimum height>,minimum height=1em,inner sep=0pt,
    infoset style ]
  (<m>)(<m>){}[](1em)
% option style for (all) information sets
  infoset style={semithick,densely dotted}
```

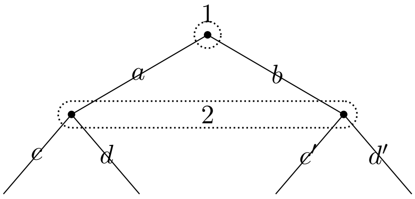
Here is an example that shows a bubble type information set as a rounded rectangle and a singleton information set as a circle.

```
% Example: \xtInfsetO
\begin{istgame}
\istroot(0){1}+15mm..30mm+
  \istb{a} \istb{b} \endist
\istroot(1)(0-1)  \istb{c}  \istb{d}  \endist
\istroot(2)(0-2)  \istb{c'} \istb{d'} \endist
\xtInfosetO(0)(0)
\xtInfosetO(1)(2){2}
\end{istgame}
```



The the height (or thickness) of an information set does not depend on `scale`, `xscale`, nor `yscale`.

```
% Example: \xtInfosetO with x-y-scale
\begin{istgame}[xscale=1.2,yscale=.7]
\istroot(0){1}+15mm..30mm+
  \istb{a} \istb{b} \endist
\istroot(1)(0-1)  \istb{c}  \istb{d}  \endist
\istroot(2)(0-2)  \istb{c'} \istb{d'} \endist
\xtInfosetO(0)(0)
\xtInfosetO(1)(2){2}
\end{istgame}
```

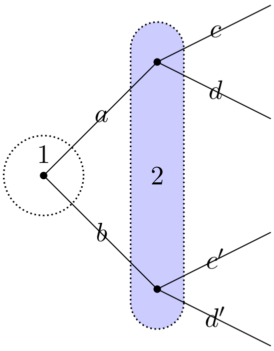


The size of an information set adjusts accordingly to the scale values.

Remark:

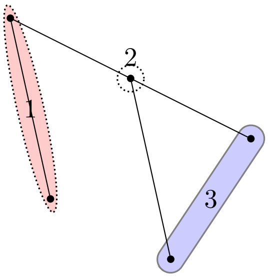
- The (minimum) height of a bubble type information set is `1em` by default. This can be changed by the last optional argument of `\xtInfosetO`, like `\xtInfosetO(1)(2)(2em)`.
- The width of a bubble information set is `\n1+<minimum height>` by default, where `\n1` is the Euclidean distance (measured by Ti*k*Z) between two nodes in an information set.

```
% Example: \xtInfosetO: changing shape or color
\begin{istgame}
\setistgrowdirection'{east}
\istroot(0){1}+15mm..30mm+
  \istb{a} \istb{b} \endist
\istroot(1)(0-1)  \istb{c}  \istb{d}  \endist
\istroot(2)(0-2)  \istb{c'} \istb{d'} \endist
\xtInfosetO[fill=blue!20,minimum width=\n1+3em]
          (1)(2){2}(2em)
\xtInfosetO(0)(0)(3em)
\end{istgame}
```



The shape of an information set does not depend on the direction of tree growing, either. You can change the shape to, for example, an ellipse by specifying it in the first bracket option list. You can also change the color of the bubble representing an information set by specifying it in the option list.

```
% Example: south east and yscale
\begin{istgame}[yscale=1.5]
\setistgrowdirection'{south east}
\istroot(0)    \istb \istb* \endist
\istroot(1)(0-1){2}  \istb* \istb* \endist
\setxtinfosetstyle{fill=red!20,ellipse}
\xtInfosetO(0)(0-2){1}
\setxtinfosetstyle{solid,fill=blue!40,opacity=.5}
\xtInfosetO(1-1)(1-2){3}
\setxtinfosetstyle % restore defaults
\xtInfosetO(1)(1)
\end{istgame}
```

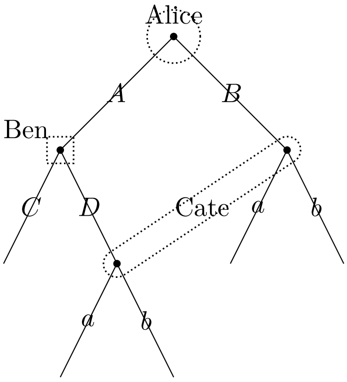


Note that, in the above example, `\setxtinfosetstyle` is used to change the style of information sets. In order to restore the option value to default (i.e., `semithick,densely dotted`) just declare `\setxtinfosetstyle`. (See Section 10.2.3 on page 48.)

10.2.2 Sloped information sets

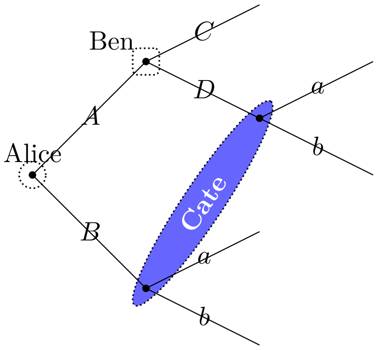
With the istgame package, a sloped information set is not special. Just connect any two nodes using `\xtInfoset` or `\xtInfoset0`.

```
% Example: sloped infoset
\begin{istgame}
\istroot(0){Alice}+15mm..30mm+
  \istb{A} \istb{B} \endist
\istroot(1)(0-1)<135>{Ben}  \istb{C} \istb{D} \endist
\istroot(2)(0-2)   \istb{a} \istb{b} \endist
\istroot(3)(1-2)   \istb{a} \istb{b} \endist
\xtInfoset0(0)(0)(2em)
\xtInfoset0[rectangle](1)(1)
\xtInfoset0(3)(2){Cate}
\end{istgame}
```



If you want to have an owner sloped too, you need the option `[sloped]`, as shown below.

```
% Example: sloped infoset with sloped text
\begin{istgame}
\setistgrowdirection'{east}
\istroot(0){Alice}+15mm..30mm+  \istb{A} \istb{B} \endist
\istroot(1)(0-1)<135>{Ben}  \istb{C} \istb{D} \endist
\istroot(2)(0-2)   \istb{a} \istb{b} \endist
\istroot(3)(1-2)   \istb{a} \istb{b} \endist
\setistmathTF*011<textbf>
\xtInfoset0(0)(0)
\xtInfoset0[rectangle,rounded corners=.2em](1)(1)
\xtInfoset0*[ellipse,fill=blue!60]
           (3)(2){Cate}[sloped,white](1.5em)
\end{istgame}
```



Warning: Issues in *sloped labels* in Ti*k*Z with *asymmetric scales*:

- The `tree` library in Ti*k*Z does not seem to treat sloped labels properly, when `xscale` or `yscale` is used asymmetrically.
- To cure this problem you can use `\xtcureslopedlabelsNS` for trees growing northwards and southwards and `\xtcureslopedlabelsWS` for trees growing eastwards and westwards. (See Section 6.2.3 and page 20, for more details with examples.)
- Note, however, that sloped labels for information owners printed by `\xtInfoset0` do not depend on scaling.

10.2.3 `\setxtinfosetstyle`

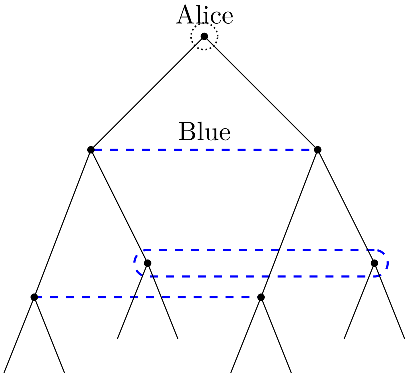
With the macro `\setxtinfosetstyle` you can change the style of all information sets, at once. For this end, a simple new style `infoset style` is defined as follows:

```
% \setxtinfosetstyle
\NewDocumentCommand \setxtinfosetstyle { m }
{ \tikzset { infoset style/.style = { semithick , densely dotted , #1 } } }
```

You can change the line style of an information set or add more options to the option list, for example, like `\setxtinfosetstyle{thin,dashed}`, or, like `\setxtinfosetstyle{blue}`.

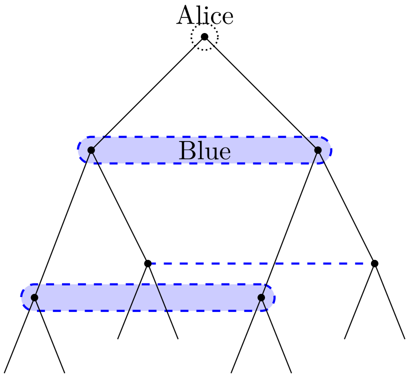
If you want get the option values back to the default values, then just declare `\setxtinfosetstyle`.

```
% Example: \setxtinfosetstyle
\begin{istgame}
\istroot(0){Alice}+15mm..30mm+
  \istb \istb \endist
\istroot(1)(0-1)  \istbA(1.3) \istb \endist
\istroot(2)(0-2)  \istbA(1.3) \istb \endist
\xtdistance{10mm}{8mm}
\istroot(3)(1-1)  \istb \istb \endist
\istroot(4)(1-2)  \istb \istb \endist
\istroot(5)(2-1)  \istb \istb \endist
\istroot(6)(2-2)  \istb \istb \endist
\setxtinfosetstyle{dashed,blue,thick}
\xtInfoset(1)(2){Blue}
\xtInfoset(1-1)(2-1)
\xtInfosetO(1-2)(2-2)
\setxtinfosetstyle % restore defaults
\xtInfosetO(0)(0)
\end{istgame}
```



With `\setxtinfosetstyle`, you can also change the background color of information sets.

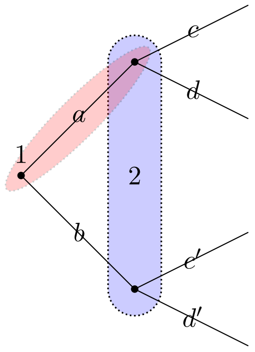
```
% Example: \setxtinfosetstyle
\begin{istgame}
\istroot(0){Alice}+15mm..30mm+
  \istb \istb \endist
\istroot(1)(0-1)  \istbA(1.3) \istb \endist
\istroot(2)(0-2)  \istbA(1.3) \istb \endist
\xtdistance{10mm}{8mm}
\istroot(3)(1-1)  \istb \istb \endist
\istroot(4)(1-2)  \istb \istb \endist
\istroot(5)(2-1)  \istb \istb \endist
\istroot(6)(2-2)  \istb \istb \endist
\setxtinfosetstyle{dashed,blue,thick,fill=blue!20}
\xtInfosetO(1)(2){Blue}
\xtInfosetO(1-1)(2-1)
\xtInfoset(1-2)(2-2)
\setxtinfosetstyle % restore defaults
\xtInfosetO(0)(0)
\end{istgame}
```



10.2.4 `\setxtinfosetlayer`

You can use the macro `\setxtinfosetlayer` to change the layer on which an information set lies from `background` (by default) to `behind`, `main`, `above`, or `foreground`, in that order. To go back to the default layer, just declare `\setxtinfosetlayer` or `\setxtinfosetlayer{}`.

```
% Example: \setxtinfosetlayer
\begin{istgame}
\setistgrowdirection'{east}
\istroot(0){1}+15mm..30mm+  \istb{a}  \istb{b} \endist
\istroot(1)(0-1)  \istb{c}  \istb{d}  \endist
\istroot(2)(0-2)  \istb{c'} \istb{d'} \endist
\setxtinfosetlayer{behind}
\xtInfosetO[fill=red,ellipse,opacity=.2](0)(1)(1.5em)
\setxtinfosetlayer
\xtInfosetO[fill=blue!20](0-1)(0-2){2}(2em)
\end{istgame}
```



10.3 `\xtCInfoset`: curved version

With the macro `\xtCInfoset` you can draw, by default, a *curved* information set and even a *skewed curved* information set, on the *background layer* by default.

Its starred version `\xtCInfoset*` prints the owner of an information set with the input mode and text font style as set by `\setistmathTF(*)` (see page 38 for examples).

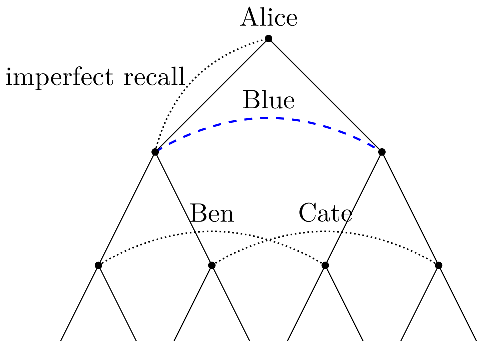
10.3.1 Curved information sets with `\xtCInfoset`: basics

The macro `\xtCInfoset` connects two nodes with a curved information set like an arch that looks like a left-bent curve (by default) from start point to end point. The basic usage of `\xtCInfoset` is the same as of `\xtInfoset`.

```
% \xtCInfoset : basics
% syntax:
  \xtCInfoset[<bubble opt>](<from>)(<to>){<owner>}[<pos>,<owner opt>](min. height)
% defaults: connecting two nodes: \xtInfoset(coor1)(coor2)
  [ rectangle,rounded corners=.5*<minimum height>*<\xtscale>,
    minimum width=\n1+<minimum height>,minimum height=1em,inner sep=0pt,
    infoset style ] %  infoset style = { semithick , densely dotted }
  (<m>)(<m>){}[](1em)
```

Here is an example of drawing curved information sets.

```
% Example: \xtCInfoset
\begin{istgame}
\istroot(0){Alice}+15mm..30mm+
  \istb \istb \endist
\istroot(1)(0-1)  \istb \istb \endist
\istroot(2)(0-2)  \istb \istb \endist
\xtdistance{10mm}{10mm}
\istroot(3)(1-1)  \istb \istb \endist
\istroot(4)(1-2)  \istb \istb \endist
\istroot(5)(2-1)  \istb \istb \endist
\istroot(6)(2-2)  \istb \istb \endist
\xtCInfoset(1)(0){imperfect recall}[left]
\xtCInfoset[dashed,blue,thick](1)(2){Blue}
\xtCInfoset(1-1)(2-1){Ben}
\xtCInfoset(1-2)(2-2){Cate}
\end{istgame}
```



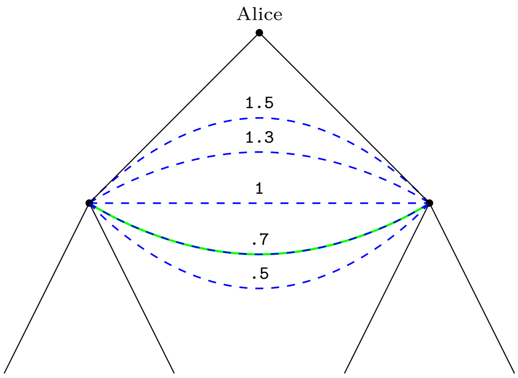
10.3.2 Skewed `\xtCInfoset`: full function

Besides the basic functions, `\xtCInfoset` has two additional optional arguments to control the shape of a curved information set. With the macro `\xtCInfoset`, you can control the shape of an information set curve by using plot factor like `<1.3>` or `<0.7>` (by default `<1.3>`) and midpoint factor like `!.4!` or `!.6!` (by default `!.5!`).

```
% \xtCInfoset : full defninition
% syntax:
  \xtCInfoset[<bubble opt>](<from>)!midpoint factor!(<to>)
             <plot factor>{<owner>}[<pos>,<owner opt>](min. height)
% defaults: connecting two nodes: \xtInfoset(coor1)(coor2)
  [ rectangle,rounded corners=.5*<minimum height>*<\xtscale>,
    minimum width=\n1+<minimum height>,minimum height=1em,inner sep=0pt,
    infoset stlye ]   % infoset style = {semithick , densely dotted}
  (<m>)(<m>){}[](1em)
```

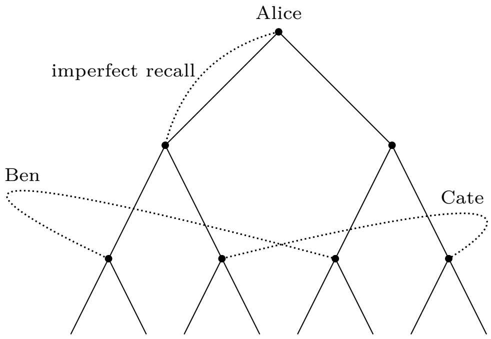
By plot factor, we mean that it determines the maximum or minimum value of a curve. If the `plot factor` is greater than `1` it prints a *concave* curve, equal to `1` a *straight* line, and less than `1` a *convex* curve, connecting form left to right. For example, `\xtCInfoset(1)(2)<.7>` (left to right) and `\xtCInfoset(2)(1)<1.3>` (right to left) will give the same result.

```
\begin{istgame}[scale=1.5,font=\scriptsize]
\istroot(0){Alice}+15mm..30mm+
  \istb \istb \endist
\istroot(1)(0-1)  \istb \istb \endist
\istroot(2)(0-2)  \istb \istb \endist
\setistmathTF*011<texttt>
\xtCInfoset*[dashed,blue](1)(2)<1.5>{1.5}
\xtCInfoset*[dashed,blue](1)(2)<1.3>{1.3}
\xtCInfoset*[dashed,blue](1)(2)<1>{1}
\setxtinfosetlayer{above}
\xtCInfoset*[dashed,blue](1)(2)<.7>{.7}
\setxtinfosetlayer{background}
\xtCInfoset*[solid,green,thick](2)(1)<1.3>
\xtCInfoset*[dashed,blue](1)(2)<.5>{.5}
\end{istgame}
```



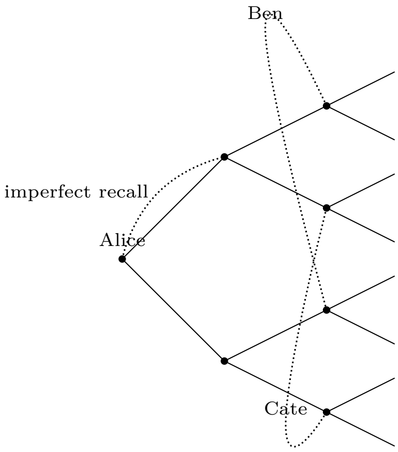
By midpoint factor, we mean that, roughly speaking, it determines the maximum or minimum point of a curve. If the `midpoint factor` is less than `.5` the curve is positively skewed, and greater than `.5` negatively skewed.

```
\begin{istgame}[font=\scriptsize]
\istroot(0){Alice}+15mm..30mm+
                  \istb \istb \endist
\istroot(1)(0-1)  \istb \istb \endist
\istroot(2)(0-2)  \istb \istb \endist
\xtdistance{10mm}{10mm}
\istroot(3)(1-1)  \istb \istb \endist
\istroot(4)(1-2)  \istb \istb \endist
\istroot(5)(2-1)  \istb \istb \endist
\istroot(6)(2-2)  \istb \istb \endist
\xtCInfoset(0)(1)<.7>{imperfect recall}[left]
\xtCInfoset(1-1)!-.05!(2-1)<1.6>{Ben}
\xtCInfoset(1-2)!.9!(2-2)<1.4>{Cate}
\end{istgame}
```



A curved information set drawn by `\xtCInfoset` does not depend on the tree growing direction. It does not depend on scaling nor swapping branches, either.

```
\begin{istgame}[scale=.9,font=\scriptsize]
\setistgrowdirection'{east}
\istroot(0){Alice}+15mm..30mm+
                  \istb \istb \endist
\istroot(1)(0-1)  \istb \istb \endist
\istroot(2)(0-2)  \istb \istb \endist
\xtdistance{10mm}{10mm}
\istroot(3)(1-1)  \istb \istb \endist
\istroot(4)(1-2)  \istb \istb \endist
\istroot(5)(2-1)  \istb \istb \endist
\istroot(6)(2-2)  \istb \istb \endist
\xtCInfoset(0)(1)<.7>{imperfect recall}[left]
\xtCInfoset(1-1)!-.05!(2-1)<1.6>{Ben}
\xtCInfoset(1-2)!.9!(2-2)<1.4>{Cate}
\end{istgame}
```



10.4 `\xtCInfosetO`: curved oval version

The macro `\xtCInfosetO` enables you to draw a *curved bubble type* information set and even a *skewed curved bubble type* information set on the *background layer* by default.

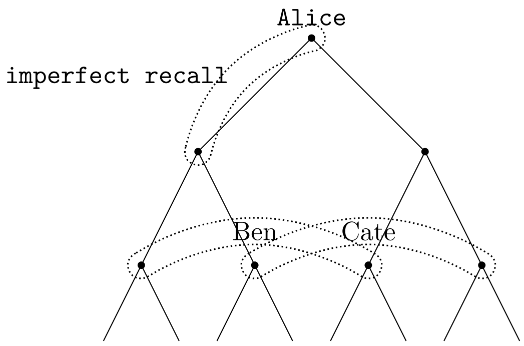
Its starred version `\xtCInfosetO*` prints the owner of an information set in input mode and text font style in accordance with `\setistmathTF(*)` (see page 38 for examples).

10.4.1 Curved bubble type information sets with `\xtCInfosetO`: basics

The basic usage of `\xtCInfosetO` is the same as of `\xtInfosetO`. If the two mandatory coordinates are identical, a *circle* is drawn to represent a *singleton information set*, like the case of `\xtInfosetO`.

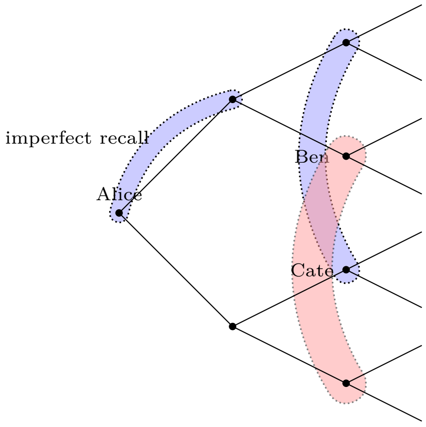
```
% \xtCInfosetO : basics
% syntax:
  \xtCInfosetO[<bubble opt>](<from>)(<to>){<owner>}[<pos>,<owner opt>](min. height)
% defaults: connecting two nodes: \xtInfoset(coor1)(coor2)
  [ rectangle,rounded corners=.5*<minimum height>*<\xtscale>,
    minimum width=\n1+<minimum height>,minimum height=1em,inner sep=0pt,
    infoset stlye ]   % infoset style = { semithick , densely dotted }
  (<m>)(<m>){}[](1em)
```

```
\begin{istgame}
\setistmathTF*011<texttt>
\istroot(0){Alice}+15mm..30mm+
                  \istb \istb \endist
\istroot(1)(0-1)  \istb \istb \endist
\istroot(2)(0-2)  \istb \istb \endist
\xtdistance{10mm}{10mm}
\istroot(3)(1-1)  \istb \istb \endist
\istroot(4)(1-2)  \istb \istb \endist
\istroot(5)(2-1)  \istb \istb \endist
\istroot(6)(2-2)  \istb \istb \endist
\xtCInfosetO*(1)(0){imperfect recall}[left]
\xtCInfosetO(1-1)(2-1){Ben}
\xtCInfosetO(1-2)(2-2){Cate}
\end{istgame}
```



The shape of information sets does not depend on the direction of tree growing. You also can change the background color and the height, like `(1.5em)` as the last option.

```
\begin{istgame}[font=\scriptsize]
\setistgrowdirection'{east}
\istroot(0){Alice}+15mm..30mm+
                  \istb \istb \endist
\istroot(1)(0-1)  \istb \istb \endist
\istroot(2)(0-2)  \istb \istb \endist
\xtdistance{10mm}{10mm}
\istroot(3)(1-1)  \istb \istb \endist
\istroot(4)(1-2)  \istb \istb \endist
\istroot(5)(2-1)  \istb \istb \endist
\istroot(6)(2-2)  \istb \istb \endist
\xtCInfosetO[fill=blue!20]
   (1)(0){imperfect recall}[left](.7em)
\xtCInfosetO[fill=blue!20]
   (1-1)(2-1){Ben}
\xtCInfosetO[fill=red!40,opacity=.5]
   (1-2)(2-2)<1.3>{Cate}(1.5em)
\end{istgame}
```



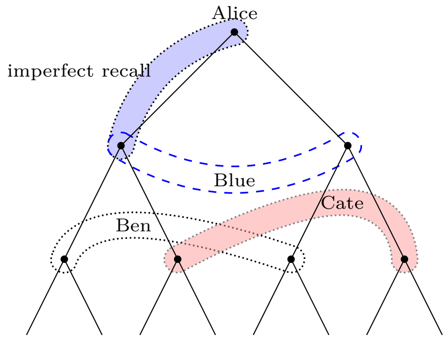
10.4.2 Skewed `\xtCInfoset0`: full function

Besides the basic functions, the macro `\xtCInfoset0` has two additional optional arguments to control the shape of a bubble representing a curved information set. The two optional arguments are plot factor and midpoint factor. (The meanings are documented on page 51.)

The `plot factor` makes a curved information set higher or lower and is used in angle brackets, like `<1.5>` or `<.7>` (by default `<1.3>`), right after the two mandatory coordinates. The `midpoint factor` controls skewness of a curved information set and is used in between the two mandatory arguments, like `!.35!` (by defaut `!.5!`).

```
% \xtCInfoset0 : full definition
% syntax:
  \xtCInfoset0[<bubble opt>](<from>)!<midpoint factor>!(<to>)
              <plot factor>{<owner>}[<pos>,<owner opt>](min. height)
% defaults: connecting two nodes: \xtInfoset(coor1)(coor2)
  [ rectangle,rounded corners=.5*<minimum height>*<\xtscale>,
    minimum width=\n1+<minimum height>,minimum height=1em,
    semithick,densely dotted,inner sep=0pt ]
  (<m>)(<m>){}[](1em)
```

```
% Example: \xtCInfoset0: skewed
\begin{istgame}[font=\scriptsize]
\istroot(0){Alice}+15mm..30mm+
                  \istb \istb \endist
\istroot(1)(0-1)  \istb \istb \endist
\istroot(2)(0-2)  \istb \istb \endist
\xtdistance{10mm}{10mm}
\istroot(3)(1-1)  \istb \istb \endist
\istroot(4)(1-2)  \istb \istb \endist
\istroot(5)(2-1)  \istb \istb \endist
\istroot(6)(2-2)  \istb \istb \endist
\xtCInfoset0[fill=blue!20]
   (1)(0){imperfect recall}[left]
\xtCInfoset0[dashed,blue](1)(2)<.7>{Blue}
\xtCInfoset0(1-1)!.35!(2-1){Ben}
\xtCInfoset0[fill=red!40,opacity=.5]
   (1-2)!.65!(2-2)<1.5>{Cate}
\end{istgame}
```



Remark:

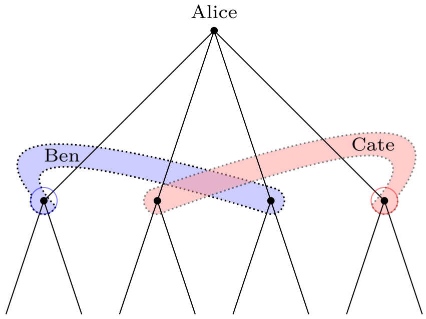
- When using `\xtCInfoset0`, the *recommended range* of the midpoint factor is between `.4` and `.6`, at most between `.35` and `.65`, otherwise you might get a result with which you are not satisfied. (You can see below Section 10.5.1 on this issue, only when you are interested in.)
- You do not need to bother if the tree is swapped, because the package internally takes care of that instead.
- An information set drawn by `\xtCInfoset0` is appropriately adjusted with `scale`, `xscale`, or `yscale`, but the height (or thickness) is independent of the scale values. (You can see below Section 10.5.2 on the issue of scaling, only when you are interested in.)

10.5 Fine-tuning `\xtCInfoset0`: Not for most users

10.5.1 `\xtCInfoset0TurnX`: too high or too low `midpoint factor`

What if you really want to use the `midpoint factor` close to `0` or `1`? In this case, you will possibly get some unsatisfactory result as shown in the example below. (Note that this being unsatisfied might not happen if you use the `midpoint factor` within the recommended range.)

```
% Example: \xtCInfoset (to be cured)
\begin{istgame}[yscale=1.5,font=\scriptsize]
\xtdistance{10mm}{10mm}
\istroot(0){Alice}+15mm..15mm+
      \istb \istb \istb \istb \endist
\istroot(1)(0-1)  \istb \istb \endist
\istroot(2)(0-2)  \istb \istb \endist
\istroot(3)(0-3)  \istb \istb \endist
\istroot(4)(0-4)  \istb \istb \endist
\xtCInfosetO[fill=blue!20]
   (1)!.2!(3)<1.4>{Ben}
\xtCInfosetO[fill=red!40,opacity=.5]
   (2)!.8!(4)<1.5>{Cate}
\tikzset{tmp/.style={draw,circle,opacity=.5,
                     minimum size=1em}}
\node at (1) [tmp,blue] {};
\node at (4) [tmp,red] {};
\end{istgame}
```



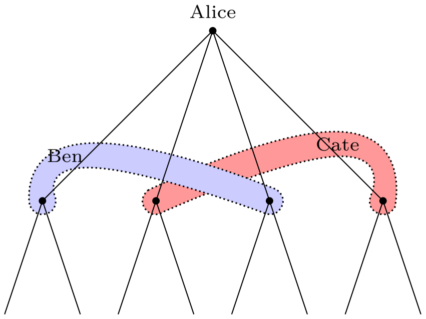
The package provides the macro `\xtCInfosetOTurnX` to make it straight.

```
% \xtCInfosetOTurnX
% syntax: X angle is mandatory
  \xtCInfosetOTurnX {turn X angle}{turn Y angle}
```

First, find the blue circle and the red circle in the above example. Let us call by `X` the (blue) circle at the beginning of an information set and by `Y` at the end. With `\xtCInfosetOTurnX` you can turn these circles to get a better result.

In the following example, `\xtCInfosetOTurnX{45}{0}` and `\xtCInfosetOTurnX{0}{-45}` are used to correct the result. If you omit the second angle like, for example, `\xtCInfosetOTurnX{45}`, it is equivalent to `\xtCInfosetOTurnX{45}{-45}`, meaning that it turns the both circles symmetrically.

```
\begin{istgame}[yscale=1.5,font=\scriptsize]
\xtdistance{10mm}{10mm}
\istroot(0){Alice}+15mm..15mm+
      \istb \istb \istb \istb \endist
\istroot(1)(0-1)  \istb \istb \endist
\istroot(2)(0-2)  \istb \istb \endist
\istroot(3)(0-3)  \istb \istb \endist
\istroot(4)(0-4)  \istb \istb \endist
\setxtinfosetlayer{behind}
\xtCInfosetOTurnX{40}{0}
\xtCInfosetO[fill=blue!20]
   (1)!.3!(3)<1.4>{Ben}[left]
\setxtinfosetlayer{background}
\xtCInfosetOTurnX{0}{-45}
\xtCInfosetO[fill=red!40]
   (2)!.7!(4)<1.5>{Cate}
\end{istgame}
```



Remark:

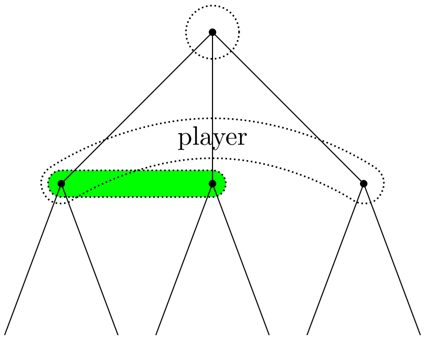
- `\xtCInfosetOTurnX` works just once only for the following `\xtCInfosetO`, without affecting any other.
- You can also change the layer of an information set, as shown in the previous example, by using `\setxtinfosetlayer` (see Secion 10.2.4).

10.5.2 Scaling information sets according to the value of Ti*k*Z `scale`

The height (or thickness) of bubble type information sets drawn by `\xtInfosetO` and `\xtCInfosetO`, does not depend on the values of Ti*k*Z scales. What if you want to make them scaled according the value of Ti*k*Z symmetric scale, for example, `[scale=.5]`? Then use, instead, `[scale=.5,xscale=1]` or equivalently `[yscale=1,scale=.5]` in any order of the options.

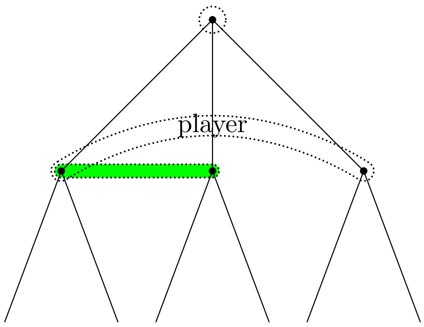
Here is an ordinary example of using `[scale=.5]`.

```
\begin{istgame}[scale=.5]
\xtdistance{40mm}{30mm}
\istroot(0)+40mm..40mm+
            \istb \istb \istb \endist
\istroot(1)(0-1)  \istb \istb \endist
\istroot(2)(0-2)  \istb \istb \endist
\istroot(3)(0-3)  \istb \istb \endist
\xtCInfosetO(0)(0)(2em)
\xtInfosetO[fill=green](1)(2)[green]
\xtCInfosetO(1)(3){player}(1.5em)
\end{istgame}
```



Here is an example to use `[scale=.5,xscale=1]`, where the height (or thickness) of information sets is scaled according to the value of Ti*k*Z scale.

```
\begin{istgame}[scale=.5,xscale=1]
\xtdistance{40mm}{30mm}
\istroot(0)+40mm..40mm+
            \istb \istb \istb \endist
\istroot(1)(0-1)  \istb \istb \endist
\istroot(2)(0-2)  \istb \istb \endist
\istroot(3)(0-3)  \istb \istb \endist
\xtInfosetO(0)(0)(2em)
\xtInfosetO[fill=green](1)(2)
\xtCInfosetO(1)(3){player}(1.5em)
\end{istgame}
```



Remark: (**Not for most users**) How the istgame environment works with the Ti*k*Z scales, regarding oval type information sets.

- In the istgame package, it is intended that the height (or thickness) of any bubble type information set is not affected by the values of Ti*k*Z scale, unless users change it. (This is a desirable feature.)
- The shapes of bubble type information sets work perfectly fine, as intended, when:
  - scale only (not with xscale nor yscale) is used in the environment option list
  - and either `xscale` or `yscale` or both (not with `scale`) are used in the option list.
- Now, this is how the istgame environment works when `scale` is used with `xscale` or `yscale`.
  - When this is the case, the value of `scale` is not taken care of by the package. Only the values of `xscale` and `yscale` are internally extracted and used to compensate the distortion of the shapes of information sets caused by the *asymmetric scaling*.
  - For example, `[scale=.5]` and `[scale=.5,xscale=1]` are treated as different (with respect to bubble type information sets) by the istgame environment, while the tikzpicture environment treats them as equal. With the latter, the istgame environment does not extract the value of `scale` to use it internally, but does only the value of `xscale`. So `scale` can affect the sizes of bubble type information sets.
  - Still, the shapes of bubble type information sets drawn by this package are not distorted by the mixed (even asymmetrical) use of Ti*k*Z scales. (In the case of asymmetric scaling, there is one exception for `\cntmAInfosetO`, which is equipped with an option to cure the distortion. See Section 11.5.3 on page 69 for more details.)

11 Continuum of branches

The package `istgame` provides the macro `\istrootcntm` and its several variants (all prefixed by `\istroot` or `\istrooto`) to express a continuum of branches and an action taken. As you will see, all the supplemental macros to a continuum of branches are prefixed by `\cntm`.

Two types of graphic objects, a triangle type and an arc type, are provided in the package to represent a continuum of branches. So you can start by choosing one type of a continuum of branches.

11.1 `\istrootcntm`: standard continuum version

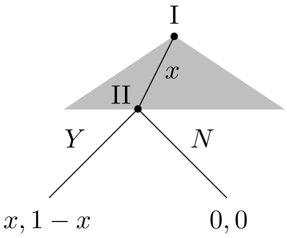
11.1.1 `\istrootcntm`: basics

The standard version `\istrootcntm` prints a background triangle, in `black!25` by default, representing a continuum of branches, on the *background layer* by default. It works just like `\istroot`, for one exception: it controls, by the last two options `+<cntmlevdist>..<cntmsibdist>+`, the distances for only the background triangle, but not the `\istb`'s following it. You can regard `\istrootcntm` as the sum `\istroot + cntm`.

```
% \istrootcntm
% syntax:
  \istrootcntm[<grow keyval>,<opt>](<coor1>)(<coor2>)[<node style>,<opt>]%
               <owner label angle>{<owner>}+<cntm-levdist>..<cntm-sibdist>+
% defaults:
  [south](<m>)(0,0)[decision node]<above>{}+8mm..24mm+
% arguments: (coor1) is mandatory, all others are optional arguments
  [grow] % the direction of growing <default: south>
  (coor1) % name of the (sub)root: mandatory
  (coor2) % the (sub)root is at (coor2) <default: (0,0)>
  [node style] % node style <default: decision node>
  <angle> % position of owner name <default: above>
  {owner} % name of the owner of the (sub)root
  +cntmlevdist..cntmsibdist+ % <defaults: 8mm,24mm>
```

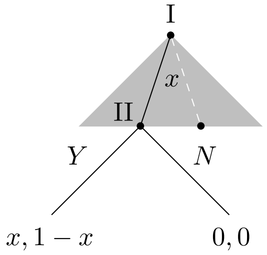
The background is a isosceles triangle, with the height of `\cntmlevdist` (`8mm` by default) and the base length of `\cntmsibdist` (`24mm` by default).

```
% Example: \istrootcntm
\begin{istgame}[scale=1.2]
\cntmlevdist{10mm}
\istrootcntm(0){I}
  \istb{x}[r]  \istbm  \endist
\xtdistance{10mm}{20mm}
\istroot(1)(0-1)<[label distance=-3pt]120>{II}
  \istb{Y}[al]{x,1-x}  \istb{N}[ar]{0,0}  \endist
\end{istgame}
```



The action taken by the owner of the root has the sibling distance (to a missing branch represented by `\istbm`, if exists) of one third (by default) of the base length of the continuum triangle.

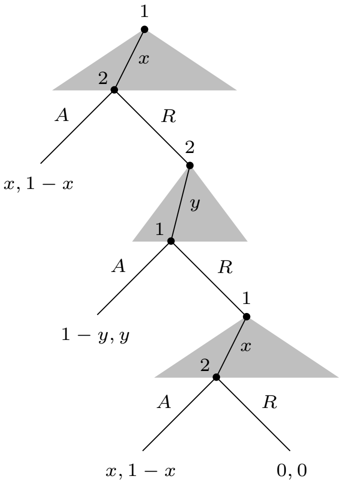
```
% Example: action sibling distance
\begin{istgame}[scale=1.2]
\xtdistance{10mm}{20mm}
\istrootcntm(0){I}+10mm..20mm+
  \istb{x}[r]  \istb*[dashed,white]  \endist
\istroot(1)(0-1)<[label distance=-3pt]120>{II}
  \istb{Y}[al]{x,1-x}  \istb{N}[ar]{0,0}  \endist
\end{istgame}
```



You can change the level and sibling distances of an individual continuum triangle by specifying the last two options of the macro `\istrootcntm` like, for example, `+10mm..15mm+`. Note that these options do not affect the action sibling distance.

Remark: Below we will see the convenient *distance changers* `\cntmdistance` and `\cntmdistance*` to change the distances for all types of continua. These macros also have a control on the *action sibling distance.*

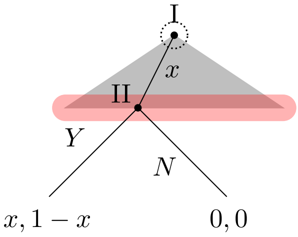
```
\begin{istgame}[font=\scriptsize]
\xtdistance{10mm}{20mm}
\istrootcntm(1){1}                   % period 1
  \istb{x}[r] \istbm \endist
\istroot(A1)(1-1)<[label distance=-3pt]135>{2}
  \istb{A}[al]{x,1-x} \istb{R}[ar]      \endist
\istrootcntm(2)(A1-2){2}+10mm..15mm+ % period 2
  \istb{y}[r] \istbm \endist
\istroot(A2)(2-1)<[label distance=-3pt]135>{1}
  \istb{A}[al]{1-y,y} \istb{R}[ar]      \endist
\istrootcntm(3)(A2-2){1}             % period 3
  \istb{x}[r] \istbm \endist
\istroot(A1)(3-1)<[label distance=-3pt]135>{2}
  \istb{A}[al]{x,1-x} \istb{R}[ar]{0,0} \endist
\end{istgame}
```



Remark:

- Once the work of `\istrootcntm` is completed by `\endist`, `\istrootcntm` internally produces three node coordinates: `(cntm)`, `(cntm-1)`, and `(cntm-2)`.
- You can use these coordinates after `\istrootcntm` and before another `\istrootcntm` or one of its variant, like `\istrootcntmA`, is used.

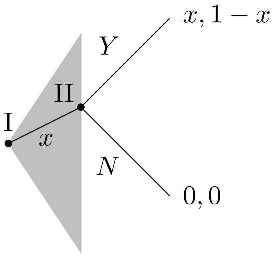
```
% Example: using coordinates: (cntm),(cntm-1),(cntm-2)
\begin{istgame}[scale=1.2]
\istrootcntm(0){I}
  \istb{x}[r]  \istbm  \endist
\xtdistance{10mm}{20mm}
\istroot(1)(0-1)<[label distance=-3pt]120>{II}
  \istb{Y}[al]{x,1-x}  \istb{N}[bl]{0,0}  \endist
\xtInfosetO(cntm)(cntm)
\xtInfosetO[draw=none,fill=red,opacity=.3](cntm-1)(cntm-2)
\end{istgame}
```



`\istrootcntm'`: swap version

The swap version `\istrootcntm'` arranges branches clockwise just like `\istroot'` does. With `\setistgrowdirection'`, both `\istrootcntm` and its swap version `\istrootcntm'` end up with the same result.

```
\begin{istgame}[scale=1.2]
\setistgrowdirection'{east}
\istrootcntm'(0){I}
  \istb{x}[b]  \istbm  \endist
\xtdistance{10mm}{20mm}
\istroot(1)(0-1)<[label distance=-3pt]120>{II}
  \istb{Y}[al]{x,1-x}  \istb{N}[bl]{0,0}  \endist
\end{istgame}
```



11.1.2 `\cntmdistance`

The macro `\cntmdistance` working for `\istrootcntm` (and `\istrootcntmA`) is analogous to `\xdistance` working for `\istroot`. As `\xtdistance` controls the height and width of a simple tree, `\cntmdistance` controls the height and width of a background triangle representing a continuum, but it has one more function. With `\cntmdistance` you can also control the sibling distance of action branches (*action sibling distance*)that are taken by the owner of the root.

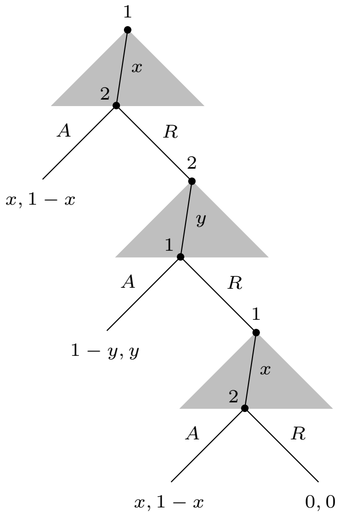
```
% \cntmdistance
% syntax:
  \cntmdistance{<cntm levdist>}{<cntm sibdist>}{<action sibdist>}
% defaults: <cntm levdist> is mandatory, and all others are optional
  {8mm}{3*<cntm lev dist>}{(1/3)*<cntm sib dist>}
```

Note that the first argument of `\cntmdistance` is *mandatory*, and all the others are optional. For example, `\cntmdistance{10mm}{20mm}` means the followings:

```
\cntmlevdist = 10mm, \cntmsibdist = 20mm, and
\cntmactsibdist = (1/3)*\cntmsibdist, from now on.
```

And `\cntmdistance{10mm}{20mm}{3mm}` means now `\cntmactsibdist = 3mm`.

```
% Example: \cntmdistance
\begin{istgame}[font=\scriptsize]
\xtdistance{10mm}{20mm}
\cntmdistance{10mm}{20mm}{3mm}
\istrootcntm(1){1} % period 1
  \istb{x}[r] \istbm \endist
\istroot(A1)(1-1)<[label distance=-3pt]135>{2}
  \istb{A}[al]{x,1-x} \istb{R}[ar] \endist
\istrootcntm(2)(A1-2){2} % period 2
  \istb{y}[r] \istbm \endist
\istroot(A2)(2-1)<[label distance=-3pt]135>{1}
  \istb{A}[al]{1-y,y} \istb{R}[ar] \endist
\istrootcntm(3)(A2-2){1} % period 3
  \istb{x}[r] \istbm \endist
\istroot(A1)(3-1)<[label distance=-3pt]135>{2}
  \istb{A}[al]{x,1-x} \istb{R}[ar]{0,0} \endist
\end{istgame}
```



11.1.3 `\cntmdistance*`: combined with `\xtdistance`

The macro `\cntmdistance*` incorporates `\cntmdistance` with `\xtdistance`.

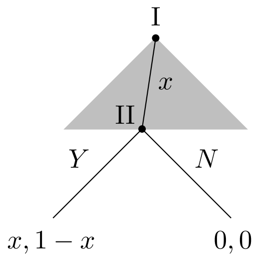
```
% \cntmdistance*
% syntax:
  \cntmdistance*{<lev dist>}{<sib dist>}{<action sib dist>}
% defaults:
  {15mm}{15mm}{(1/3)*<sib dist>}
```

The macro `\cntmdistance` does not have anything to do with `\xdistance`, but the starred version `\cntmdistance*` does. For example, `\cntmdistance*{10mm}{20mm}` means the followings:

```
\cntmlevdist = \xtlevdist = 10mm, \cntmsibdist = \xtsibdist = 20mm, and
\cntmdistance = (1/3)*\cntmdisdist, from now on.
```

And `\cntmdistance*{10mm}{20mm}{3mm}` means now `\cntmactsibdist = 3mm`.

```
% Example: \cntmdistance*: with \xtdistance
\begin{istgame}[scale=1.2]
\cntmdistance*{10mm}{20mm}{3mm}
\istrootcntm(0){I}
  \istb{x}[r]  \istbm  \endist
%\xtdistance{10mm}{20mm}
\istroot(1)(0-1)<[label distance=-3pt]120>{II}
  \istb{Y}[al]{x,1-x}  \istb{N}[ar]{0,0}  \endist
\end{istgame}
```



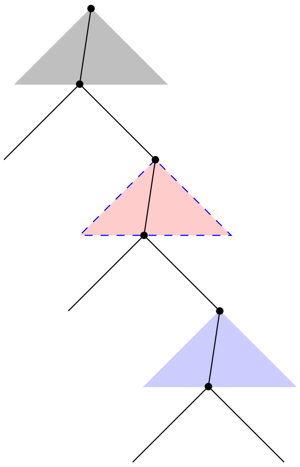
11.1.4 `\cntmpreset`: controlling continuum triangles

The macro `\cntmpreset` controls the line style and color, size (via the `shrink factor` with `1` as default), and fill color of a continuum triangle drawn by `\istrootcntm`.

```
% \cntmpreset
% syntax:
  \cntmpreset[<line style>,<opts>]{<shrink factor>}[<fill color>]
% defaults
  [-,solid,draw=black!25,fill=<draw color>]{1}[]
```

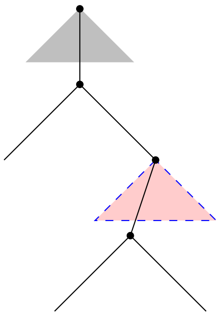
To change the background color you can do like, for example, either `\cntmpreset[blue!20]` or `\cntmpreset{1}[blue!20]`. However, like `\cntmpreset[blue!20][red!20]`, if you specify both of the bracket optional arguments, then the *first one wins*. The second bracket option is just for your convenience to put *fill color* easily.

```
\begin{istgame}[font=\scriptsize]
\cntmdistance*{10mm}{20mm}{3mm}
\istrootcntm(1)
  \istb \istbm \endist
\istroot(A1)(1-1)
  \istb \istb  \endist
\cntmpreset[draw=blue,dashed][red!20]
\istrootcntm(2)(A1-2)
  \istb \istbm \endist
\istroot(A2)(2-1)
  \istb \istb  \endist
\cntmpreset[blue!20][red!20]
\istrootcntm(3)(A2-2)
  \istb \istbm \endist
\istroot(A1)(3-1)
  \istb \istb  \endist
\end{istgame}
```



You can also draw a *smaller* triangle, by specifying shrink factor as a decimal number in the curly braces.

```
\begin{istgame}[font=\scriptsize]
\cntmdistance*{10mm}{20mm}
\cntmpreset{.7}
\istrootcntm(1)       \istb        \endist
\istroot(A1)(1-1)     \istb \istb  \endist
\cntmpreset[draw=blue,dashed]{.8}[red!20]
\istrootcntm(2)(A1-2) \istb \istbm \endist
\istroot(A2)(2-1)     \istb \istb  \endist
\end{istgame}
```



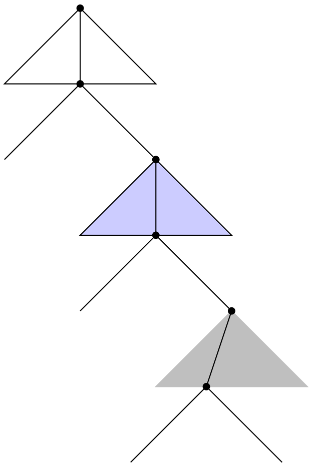
11.1.5 `\cntmpreset*`: simple triangles with no background color

The macro `\cntmpreset*` works like `\cntmpreset`, for one exception: it prints, by default, a *simple triangle with sides drawn, but with no background color.*

```
% \cntmpreset
% syntax:
  \cntmpreset*[<line style>,<opts>]{<shrink factor>}[<fill color>]
% defaults
  [-,solid,draw=black,fill=none]{1}[]
```

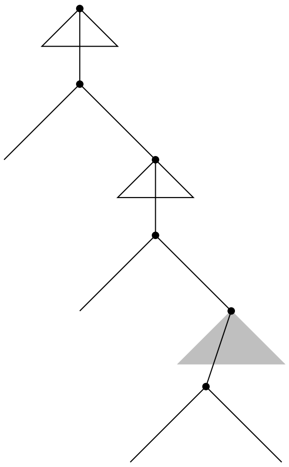
Remark: In order to restore the standard triangle options (i.e., `[-,draw=none,black!25]`), you can just declare `\cntmpreset` followed by nothing. More precisely, if you use `\cntmpreset` without specifying the first bracket option, like `\cntmpreset{.7}`, the default options are restored.

```
% Example: \cntmpreset(*)
\begin{istgame}[font=\scriptsize]
\cntmdistance*{10mm}{20mm}
\cntmpreset*
\istrootcntm(1)
  \istb        \endist
\istroot(A1)(1-1)
  \istb \istb  \endist
\cntmpreset*[fill=blue!20]
\istrootcntm(2)(A1-2)
  \istb        \endist
\istroot(A2)(2-1)
  \istb \istb  \endist
\cntmpreset % restore default triangle options
\istrootcntm(3)(A2-2)
  \istb \istbm \endist
\istroot(A1)(3-1)
  \istb \istb  \endist
\end{istgame}
```



You can also draw a *smaller* triangle, by specifying shrink factor as a decimal number in the curly braces.

```
% Example: \cntmpreset(*)
\begin{istgame}[font=\scriptsize]
\cntmdistance*{10mm}{20mm}
\cntmpreset*{.5}
\istrootcntm(1)
  \istb        \endist
\istroot(A1)(1-1)
  \istb \istb  \endist
\istrootcntm(2)(A1-2)
  \istb        \endist
\istroot(A2)(2-1)
  \istb \istb  \endist
\cntmpreset{.7} % restore default triangle options
\istrootcntm(3)(A2-2)
  \istb \istbm \endist
\istroot(A1)(3-1)
  \istb \istb  \endist
\end{istgame}
```



11.2 `\istrootcntmA`: continuum arc version

11.2.1 `\istrootcntmA`: basics

The macro `\istrootcntmA` works just like `\istroot`, but it draws an arc to express a continuum of branches. Here `cntmA` is an abbreviation of `cntmarc`. You can regard `\istrootcntmA` as the sum `\istroot + cntmA`.

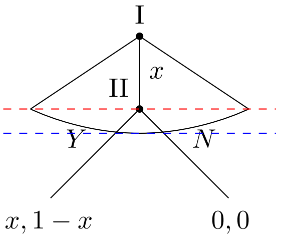
Remark:

- There is one important difference between `\istroot` and `\istrootcntmA`. By the last two options (for example, `+10mm..20mm+`), the `\istrootcntmA` controls the level and sibling distances of a continuum arc, while `\istroot` controls the level and sibling distances of action branches (represented by the `\istb`) following it.
- Note that the distance changers `\cntmdistance` and `\cntmdistance*` work the same for `\istrootcntmA` as well as `\istrootcntm`. (*There is no such thing as `\cntmAdistance`.*)

```
% \istrootcntmA
% syntax:
  \istrootcntmA[<grow keyval>,<opt>](<coor1>)(<coor2>)[<node style>,<opt>]%
               <arc pos>{<owner>}+<cntm levdist>..<cntm sibdist>+
% defaults:
  [south](<m>)(0,0)[decision node]<above>{}+8mm..24mm+
% arguments: (coor1) is mandatory, all others are optional arguments
  [grow] % the direction of growing <default: south>
  (coor1) % name of the (sub)root: mandatory
  (coor2) % the (sub)root is at (coor2) <default: (0,0)>
  [node style] % node style <default: decision node>
  <owner pos> % position of the ownter's name
  {owner} % name of the owner of the (sub)root
  +level dist..sibling dist+ % <defaults: 8mm,24mm>
```

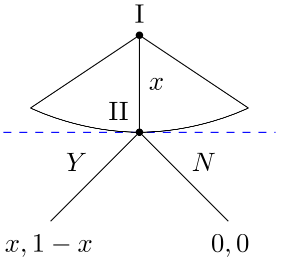
The default level and sibling distance of a continuum of branches are `8mm` and `3*8mm`. With the default distances, the distance between the root and the lowest point of the arc is one third longer (with all default values) than the continuum level distance.

```
\begin{istgame}[scale=1.2]
\istrootcntmA(0){I}  \istb{x}[r]  \endist
\xtdistance{10mm}{20mm}
\istroot(1)(0-1)<120>{II}
  \istb{Y}[al]{x,1-x}  \istb{N}[ar]{0,0}  \endist
\draw [dashed,red]
      ([xshift=-3mm]cntm-1) -- ([xshift=3mm]cntm-2);
\coordinate (C) at ($(cntm)!1.333!(0-1)$);
\draw [dashed,blue] (C-|{-1.5,0}) -- (C-|{1.5,0});
\end{istgame}
```



With the macro `\istbA` you can draw a single branch ending up with an endpoint on the continuum arc. Just remember `\istb` always reaches the *red dashed line* and `\istbA` the *blue dashed line*. For more details, see below Section 11.2.2 (though it is not for first readers).

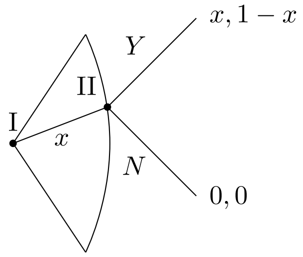
```
\begin{istgame}[scale=1.2]
\istrootcntmA(0){I}
  \istbA{x}[r]  \endist
\xtdistance{10mm}{20mm}
\istroot(1)(0-1)<120>{II}
  \istb{Y}[al]{x,1-x}  \istb{N}[ar]{0,0}  \endist
\xtTimeLineH[dashed,blue](0-1){-1.5}{1.5}
\end{istgame}
```



\istrootcntmA'

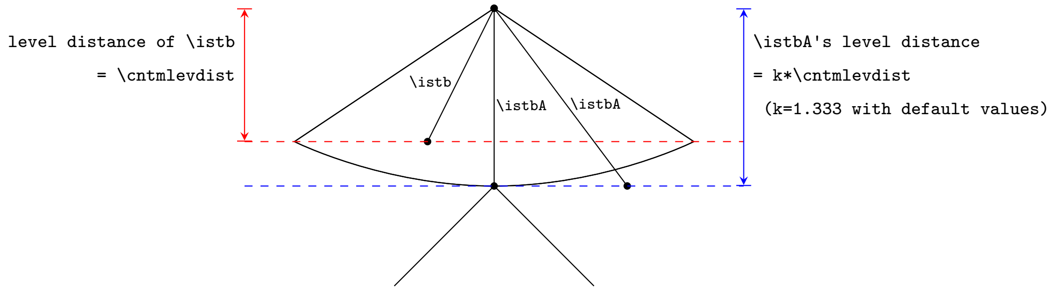
The swap version `\istrootcntmA'` arranges branches clockwise, like `\istroot'` does. Note also that if you use `\setistgrowdirection'`, then using the swap version makes no difference in results.

```
% Example: \istrootcntmA' (growing east)
\begin{istgame}[scale=1.2]
\setistgrowdirection'{east}
\istrootcntmA'(0){I}
  \istb<level distance=1.3*\cntmlevdist>{x}[b]  \istbm
    \endist
\xtdistance{10mm}{20mm}
\istroot(1)(0-1)<120>{II}
  \istb{Y}[al]{x,1-x}  \istb{N}[bl]{0,0}  \endist
\end{istgame}
```



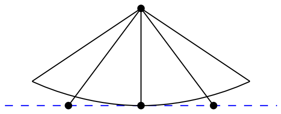
11.2.2 \istrootcntmA and \istbA

As discussed in Section 5.5 on page 26 `\istb` and `\istbA` can be used interchangeably (but with one exception). However, when it comes to `\istrootcntmA`, `\istbA` has a unique function: it reaches a continuum arc.



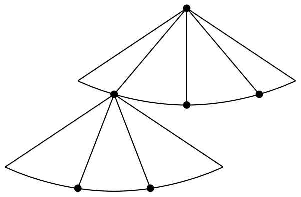
To be precise, `\istbA` reaches the *blue dashed line*, as shown in the above picture, while `\istb` reaches the *red dashed line*. So when `\istbA` is the only child of the root, it arrives exactly at the continuum arc.

```
% Example: \istrootcntmA and \istbA
\begin{istgame}[scale=1.2]
\istrootcntmA(0)
  \istbA*
  \istbA*
  \istbA*
  \endist
\xtTimeLineH[dashed,blue](0-2){-1.5}{1.5}
\end{istgame}
```



Moreover, as also discussed in Section 5.5 on page 26, `\istbA` has one more function than `\istb`: it can easily change its level distance by using the first parenthesis option. This allows you to make any `\istbA` reach the continuum arc, but after some trial and errors.

```
% Example: \istrootcntmA and \istbA
\begin{istgame}[scale=1.2]
\istrootcntmA(0)
  \istbA*(.89)  \istbA*    \istbA*(.89)
  \endist
\istrootcntmA(1)(0-1)
  \istbA*(.97) \istbA*(.97) \endist
\end{istgame}
```



11.2.3 `\cntmApreset`: controlling continuum arcs and wedges

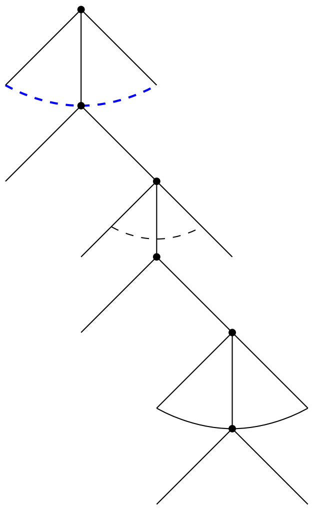
You can control the line style, position (via `arc position factor`), and *curvature* (via `arc plot factor`) of a continuum arc with `\cntmApreset`. You can also change the background color (by default `transparent`) of the *wedge* formed by a continuum arc.

```
% syntax: \cntmApreset
  \cntmApreset*[<arc line opt>]<arc plot factor>{<arc pos factor>}[<fill color>]
% defaults: all arguments are optional
  [-,tension=1]<1.333>{1}[transparent]
```

You can change the line style or the color of a continuum arc, in the usual Ti*k*Z way, like `[dashed,thick,blue]`. If you do not specify any optional argument, the corresponding argument will have the default values. So, by declaring `\cntmApreset` followed by nothing, you can restore all the default values.

A continuum arc connects the two endpoints, by default, of the two outermost branches. If the arc position factor (between zero and one) is less than `{.5}` the arc gets closer to the root, and if greater then it gets closer to the endpoints. (What if the factor is greater than `1`? You try!)

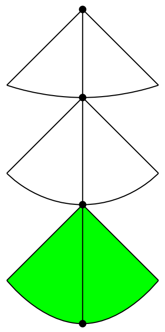
```
\begin{istgame}
\cntmdistance*{10mm}{20mm}
\cntmApreset[dashed,blue,thick]
\istrootcntmA(1)
  \istbA       \endist
\istroot(A1)(1-1)
  \istb \istb  \endist
\cntmApreset[dashed]{.6}
\istrootcntmA(2)(A1-2)
  \istb        \endist
\istroot(A2)(2-1)
  \istb \istb  \endist
\cntmApreset % restore defaults
\istrootcntmA(3)(A2-2)
  \istbA       \endist
\istroot(A1)(3-1)
  \istb \istb  \endist
\end{istgame}
```



Note that `\istbA` (if it is the only child) always prints a branch with its endpoint on an arc . Note also that the effect of `\cntmApreset` goes on until another `\cntmApreset`, which restores the values to defautls, is used.

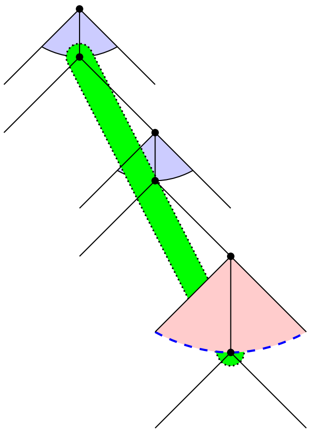
As for the curvature, if the factor (usually, greater than 1) typed in angle brackets is smaller than the default value `<1.333>` the arc gets flatter, and if larger it gets sharper. (If the factor is `1`, it becomes a straight line. What if the factor is less than `1`? You guess!)

```
\begin{istgame}
\cntmdistance*{10mm}{20mm}
\cntmApreset<1.2>
\istrootcntmA(1)
  \istbA        \endist
\cntmApreset[tension=1.12]<1.508>
\istrootcntmA(2)(1-1)
  \istbA        \endist
\cntmApreset<1.7>[green]
\istrootcntmA(3)(2-1)
  \istbA*       \endist
\end{istgame}
```



In order to change the background color of the wedge formed by an arc, you should use the *second bracket option*. The first one is for a continuum arc line. The color filled wedge lies on the `behind` layer, by default. You can change the layer by using `\cntmAlayerpreset`. Available layers are: `background`, `behind`, `main`, `above`, and `foreground`, in that order. To restore the default layer (i.e., `behind`), just declare `\cntmAlayerpreset`.

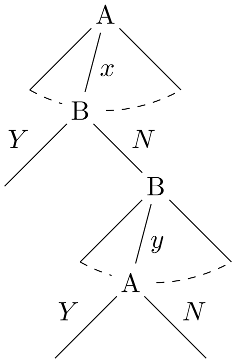
```
\begin{istgame}
\cntmdistance*{10mm}{20mm}
\cntmApreset{.5}[blue!20]
\cntmAlayerpreset{background}
\istrootcntmA(1)          \istbA \endist
\istroot(A1)(1-1)   \istb \istb  \endist
\istrootcntmA(2)(A1-2)    \istbA \endist
\istroot(A2)(2-1)   \istb \istb  \endist
\cntmApreset[dashed,blue,thick][red!20]
\cntmAlayerpreset % restore default (behind)
\istrootcntmA(3)(A2-2)    \istbA \endist
\istroot(A3)(3-1)   \istb \istb  \endist
\xtInfosetO[fill=green](A1)(A3)
\end{istgame}
```



11.3 `\istrootocntm` and `\istrootocntmA`: oval continuum versions

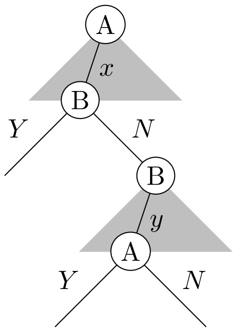
In some cases, you may want to use `\istrootocntmA`, the oval version of `\istrootcntmA`. It can be regarded as the sum `\istrooto + cntmA`. The swap version `\istrootocntmA'` is also provided.

```
% Example: \istrooto+cntmA
\begin{istgame}
\setistEllipseNodeStyle[white]
\cntmdistance*{10mm}{20mm}
\cntmApreset[dashed]
\istrootocntmA(1){A}  \istbA{x}[r]        \istbm  \endist
\istrooto(1a)(1-1){B}
                      \istb{Y}[al]  \istb{N}[ar]  \endist
\istrootocntmA(2)(1a-2){B}  \istbA{y}[r]  \istbm  \endist
\istrooto(2a)(2-1){A}
                      \istb{Y}[al]  \istb{N}[ar]  \endist
\end{istgame}
```



It is anyway possible to use `\istrootocntm`, which is the oval version of `\istrootcntm`. It can be regarded as the sum `\istrooto + cntm`. You can also use its swap version `\istrootocntm'`.

```
% Example: \istrooto+cntm
\begin{istgame}
\cntmdistance*{10mm}{20mm}
\cntmApreset[dashed]
\istrootocntm(1){A}  \istbA{x}[r]        \istbm  \endist
\istrooto(1a)(1-1){B}
                     \istb{Y}[al]  \istb{N}[ar]  \endist
\istrootocntm(2)(1a-2){B}  \istbA{y}[r]  \istbm  \endist
\istrooto(2a)(2-1){A}
                     \istb{Y}[al]  \istb{N}[ar]  \endist
\end{istgame}
```



11.4 Doing some chores: not for most users

11.4.1 `\cntmistb`: controlling outermost branches

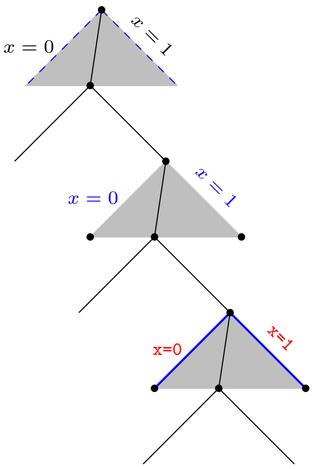
The macro `\cntmistb` controls the outermost branches of a continuum triangle. This works similar to `\istb`, but you need to use this macro *right before* the `\istrootcntm`. And it works just *once* only for the immediately following `\istrootcntm` and its close variants, but not for the continuum arc versions.

Remark: The macro `\cntmistb*` prints `solid nodes` at the endpoints of the outermost branches.

- The effect of printing `solid nodes` affects *all the following continua*. It can be *turned off* by the starred version `\xtHideEndPoints*` (see Section 11.4.3, for more details).

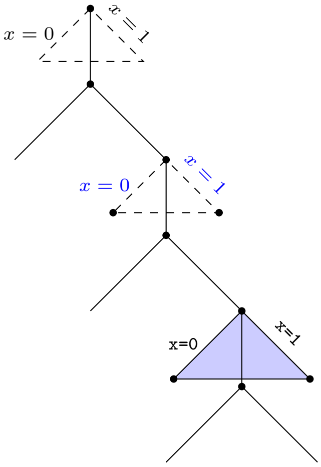
```
% syntax: \cntmistb
  \cntmistb[<branch opt>]{<left action label>}[<pos>]{<right action label>}[<pos>]
% defaults: all arguments are optional
  [-,draw=none]{}[]{}[]
```

```
\begin{istgame}[font=\scriptsize]
\cntmdistance*{10mm}{20mm}{3mm}
\cntmistb[draw=blue,dashed]{x=0}[l]{x=1}[above,sloped]
\istrootcntm(1)
  \istb \istbm \endist
\istroot(A1)(1-1)
  \istb \istb \endist
\cntmistb*[blue]{x=0}[l]{x=1}[above,sloped]
\istrootcntm(2)(A1-2)
  \istb \istbm \endist
\istroot(A2)(2-1)
  \istb \istb \endist
\setistmathTF*000{texttt}
\cntmistb[red,draw=blue,thick]{x=0}[l]{x=1}[above,sloped]
\istrootcntm(3)(A2-2)
  \istb \istbm \endist
\istroot(A1)(3-1)
  \istb \istb \endist
\end{istgame}
```



`\cntmistb(*)` prints action labels in math mode by default, and the input mode can be changed by `\setistmathTF(*)`.

```
\begin{istgame}[font=\scriptsize]
\cntmpreset*[dashed]{.7}
\cntmdistance*{10mm}{20mm}{3mm}
\cntmistb{x=0}[l]{x=1}[above,sloped]
\istrootcntm(1)       \istb        \endist
\istroot(A1)(1-1)     \istb \istb  \endist
\cntmistb*[blue]{x=0}[l]{x=1}[above,sloped]
\istrootcntm(2)(A1-2) \istb        \endist
\istroot(A2)(2-1)     \istb \istb \endist
\setistmathTF*000{texttt}
\cntmpreset*{.9}[blue!20]
\cntmistb{x=0}[l]{x=1}[above,sloped]
\istrootcntm(3)(A2-2) \istb        \endist
\istroot(A1)(3-1)     \istb \istb \endist
\end{istgame}
```



11.4.2 `\cntmAistb`: controlling outermost branches

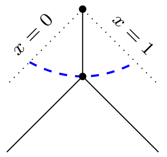
The macro `\cntmAistb` controls the outermost branches of a continuum arc. This works similar to `\istb`, but you need to use this macro *right before* the `\istrootcntmA`. And it works just *once* only for the immediately following `\istrootcntmA` and its close variants, but not for the continuum triangle versions.

The macro `\cntmAistb*` prints `solid nodes` at the endpoints of the outermost branches. The effect of printing `solid nodes` affects *all the following continua*. It can be *turned off* by the starred version `\xtHideEndPoints*` (see Section 11.4.3, for more details).

```
% syntax: \cntmAistb
  \cntmistb[<branch opt>]{<left action label>}[<pos>]{<right action label>}[<pos>]
% defaults: all arguments are optional
  [-]{}[]{}[]
```

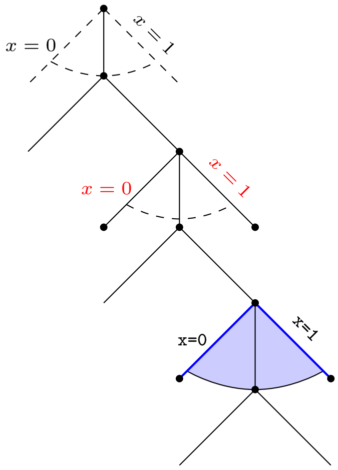
Note that `\cntmAistb` controls outermost branches of a continuum, while `\cntmApreset` controls the arc line.

```
\begin{istgame}[font=\scriptsize]
\cntmdistance*{10mm}{20mm}
\cntmAistb[dotted]{x=0}[above,sloped]{x=1}[above,sloped]
\cntmApreset[dashed,blue,thick]{.7}
\istrootcntmA(1)       \istbA         \endist
\istroot(A1)(1-1)     \istbA \istbA   \endist
\end{istgame}
```



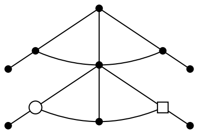
`\cntmAistb(*)` prints action labels in math mode by default, and the input mode can be changed by `\setistmathTF(*)`.

```
\begin{istgame}[font=\scriptsize]
\cntmdistance*{10mm}{20mm}
\cntmAistb[dashed]{x=0}[l]{x=1}[above,sloped]
\cntmApreset[dashed]{.7}
\istrootcntmA(1)       \istbA         \endist
\istroot(A1)(1-1)     \istbA \istbA   \endist
\cntmAistb*[red,draw=black]{x=0}[l]{x=1}[above,sloped]
\istrootcntmA(2)(A1-2) \istb         \endist
\istroot(A2)(2-1)      \istbA \istbA \endist
\setistmathTF*000{texttt}
\cntmAistb[draw=blue,thick]{x=0}[l]{x=1}[above,sloped]
\cntmApreset{.9}[blue!20]
\istrootcntmA(3)(A2-2) \istbA         \endist
\istroot(A1)(3-1)      \istbA \istbA \endist
\end{istgame}
```



It is not impossible to print two endpoints of a continuum arc, even thought it does not make any game theoretical sense. You can easily do that by using `\cntmAexpostShowEndPoints` right after `\istrootcntmA`.

```
\begin{istgame}[font=\scriptsize]
\cntmAistb*
\cntmApreset{.7}
\istrootcntmA(1)         \istbA   \endist
\cntmAexpostShowEndPoints
\istrootcntmA(A1)(1-1)   \istbA*   \endist
\cntmAexpostShowEndPoints[oval node][box node]
\end{istgame}
```



11.4.3 \xtShowEndPoints*

The starred version `\xtShowEndPoints*` has additional control over the endpoints of outermost branches of a continuum, when it is used with `\cntmistb` or `\cntmAistb`. However, the starred versions `\cntmistb*` and `\cntmAistb*` do not obey `\xtShowEndPoints*`. They anyway print solid nodes at the endpoints of outermost branches and the effect goes on until it is blocked by `\xtShowEndPoints*` or `\xtHidePoints*`.

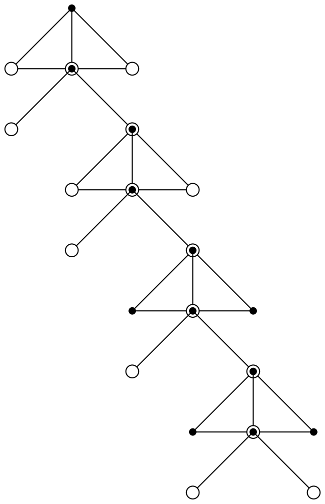
`\xtShowEndPoints*` competes with each of the starred versions `\cntmistb*` and `\cntmAistb*` for the control over `\cntmistb` and `\cntmAistb` with respect to the endpoints of outermost branches. Whichever comes later wins.

The starred version `\xtHideEndPoints*` works *only for the endpoints of the outermost branches* to turn off the effects of `\xtShowEndPoints(*)`, `\cntmistb*`, and `\cntmAistb*`. But whichever comes later wins.

Note that the non-starred versions `\xtShowEndPoints` and `\xtHideEndPoints` have nothing to do with the outermost endpoints of a continuum.

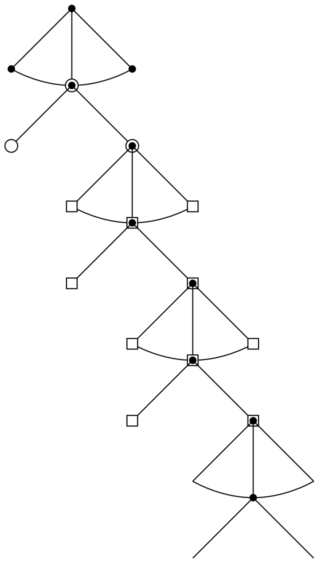
In the example below, `\cntmistb*` overrides the effect of `\xtShowEndPoint*[oval node]`, with respect to the outermost endpoints of continua.

```
\begin{istgame}
\cntmdistance*{8mm}{16mm}
\cntmpreset*
\xtShowEndPoints*[oval node]
\cntmistb
\istrootcntm(1)           \istb  \endist
\istroot(A1)(1-1)  \istb  \istb  \endist
\cntmistb
\istrootcntm(2)(A1-2)     \istb  \endist
\istroot(A2)(2-1)  \istb  \istb  \endist
\cntmistb*
\istrootcntm(3)(A2-2)     \istb  \endist
\istroot(A3)(3-1)  \istb  \istb  \endist
\cntmistb
\istrootcntm(4)(A3-2)     \istb  \endist
\istroot(A4)(4-1)  \istb  \istb  \endist
\end{istgame}
```



In the example below, `\xtHideEndPoints` and `\xtHidePoints*` cooperatively turn off the effects of `\xtShowEndPoints*` and `\cntmAistb`*.

```
\begin{istgame}
\cntmdistance*{8mm}{16mm}
\xtShowEndPoints*[oval node]
\cntmAistb*
\istrootcntmA(1)          \istbA \endist
\istroot(A1)(1-1)  \istb  \istb  \endist
\xtShowEndPoints*[box node]
\cntmAistb
\istrootcntmA(2)(A1-2)    \istbA \endist
\istroot(A2)(2-1)  \istb  \istb  \endist
\cntmAistb
\istrootcntmA(3)(A2-2)    \istbA \endist
\istroot(A3)(3-1)  \istb  \istb  \endist
\cntmAistb*
\xtHideEndPoints* % turns off only cntm endpoints
\xtHideEndPoints
\istrootcntmA(4)(A3-2)    \istbA \endist
\istroot(A4)(4-1)  \istb  \istb  \endist
\end{istgame}
```

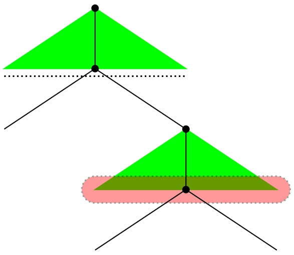


11.5 Information sets for continua of branches

11.5.1 Information sets for continuum triangles

It is not an issue to draw information sets for continuum triangles. You can just apply `\xtInfoset` or `\xtInfosetO`.

```
\begin{istgame}
\cntmdistance*{8mm}
\cntmpreset{1}[green]
\istrootcntm(1)           \istb  \endist
\xtInfoset([yshift=-1mm]cntm-1)([yshift=-1mm]cntm-2)
\istroot(A1)(1-1)  \istb  \istb  \endist
\istrootcntm(2)(A1-2)     \istb  \endist
\xtInfosetO[fill=red,opacity=.4](cntm-1)(cntm-2)
\istroot(A2)(2-1)  \istb  \istb  \endist
\end{istgame}
```



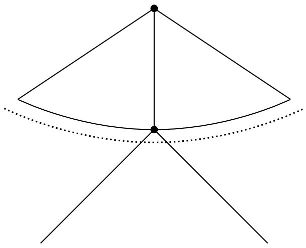
11.5.2 `\cntmAInfoset` for continuum arcs

The macro `\cntmAInfoset` draws an arc to represent an information set for an arc type continuum of branches, on the background layer. The default position of the arc information set is `1+.1`, meaning that `.1` (by default) is added to the position (`1` by default) of a continuum arc. It must be used *right after* the corresponding `\istrootcntmA` whose root coordinate is its *mandatory argument*. It does not have an option for an information set owner. (If needed, you can use supplementary macros such as `\xtInfosetOwner` and `\xtOwner`.)

```
% syntax: \cntmAInfoset
  \cntmAInfoset*[<line opt>](<cntm root>)<plot factor>{<addto cntmApos>}
% defaults: (<cntm root>) is mandatory
  [](<m>)<1.333>{.1}
```

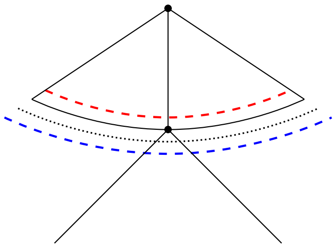
You can change, if necessary, the curvature of an information set arc by using plot factor (`<1.333>` by default).

```
% Example: \cntmAInfoset
\begin{istgame}[scale=1.5]
\istrootcntmA(0)
  \istbA  \endist
\cntmAInfoset(0)<1.34>
\xtdistance{10mm}{20mm}
\istroot(1)(0-1)
  \istb  \istb  \endist
\end{istgame}
```



You can also change the position and the line style of an information set arc, as shown in the following example:

```
% Example: \cntmAInfoset
\begin{istgame}[scale=1.5]
\istrootcntmA(0)
  \istbA  \endist
\cntmAInfoset[red,dashed,thick](0){-.1}
\cntmAInfoset(0)
\cntmAInfoset[blue,dashed,thick](0){.2}
\xtdistance{10mm}{20mm}
\istroot(1)(0-1)
  \istb  \istb  \endist
\end{istgame}
```

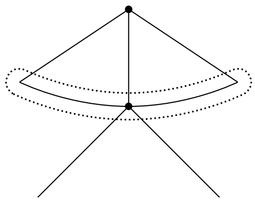


11.5.3 `\cntmAInfosetO`: oval version

The oval version `\cntmAInfosetO` enables you to draw a bubble type information set for a continuum arc. It must be used right after a simple tree with the root produced by `\istrootcntmA` and accepts the root of the continuum as the mandatory argument. `\cntmAInfosetO` can control the style of the information set and height (or thickness), but it does not have an option for the owner of a information set.

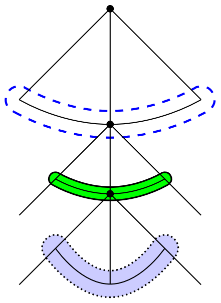
```
% syntax: \cntmAInfosetO
  \cntmAInfosetO[<line opt>](<cntm root>)
                <plot factor>{<addto cntmApos>}[<turn X-ang>](<infoset height>)
% defaults: (<cntm root>) is mandatory
  [](<m>)<1.333>{.1}[0](1em)
```

```
\begin{istgame}[scale=1.2]
\istrootcntmA(0)  \istbA        \endist
\cntmAInfosetO(0)
\xtdistance{10mm}{20mm}
\istroot(1)(0-1)  \istb  \istb  \endist
\end{istgame}
```



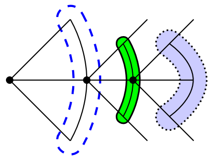
You can change the style of an information set by using usual Ti*k*Z options. You can also change its position and curvature via `\cntmApreset`.

```
\begin{istgame}[scale=1.2]
\cntmdistance*{10mm}{20mm}
\istrootcntmA(0)       \istbA  \endist
\cntmAInfosetO[blue,dashed,thick](0)
\cntmApreset{.6}
\istrootcntmA(1)(0-1)  \istbA  \endist
\cntmAInfosetO[solid,fill=green](1)(.5em)
\cntmApreset<1.8>{.6}
\istrootcntmA(2)(1-1)  \istbA  \endist
\cntmAInfosetO[fill=blue!20](2)
\end{istgame}
```



You do not need to bother if a tree is swapped or rotated. Note also that the height (or thickness) of information set are not affected by Ti*k*Z scaling.

```
\begin{istgame}[scale=.8]
\setistgrowdirection'{east}
\cntmdistance*{10mm}{20mm}
\istrootcntmA(0)       \istbA  \endist
\cntmAInfosetO[blue,dashed,thick](0)
\cntmApreset{.6}
\istrootcntmA(1)(0-1)  \istbA  \endist
\cntmAInfosetO[solid,fill=green](1)(.5em)
\cntmApreset<1.8>{.6}
\istrootcntmA(2)(1-1)  \istbA  \endist
\cntmAInfosetO[fill=blue!20](2)
\end{istgame}
```



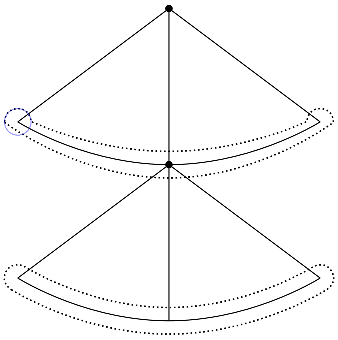
If necessary, you can use the `turn X-angle` option to get a better result (see the following Section 11.5.4 for this issue).

11.5.4 Fine-tuning `\cntmAInfoset0`: Not for most users

turing X circles

In the case of a severely asymmetrical Ti*k*Z scale (using `xsacle` and `yscale` asymmetrically), an information set drawn by `\cntmAInfoset0` will probably be distorted, as shown in the upper part of the example below. The bracket option to be used after the mandatory argument of `\cntmAInfoset0` can solve this problem by turning the *blue circle* (called, an `X` circle) drawn at the beginning point of the information set. To do that, specify an appropriate degree of rotation in brackets, like `\cntmAInfoset0(2)[60]`, as shown below. (See Section 10.5.1 for a related topic.)

```
% Example: fine-tunig \cntmAInfoset0
\begin{istgame}[xscale=4]
\cntmdistance*{15mm}{10mm}
\cntmApreset<1.8>
\istrootcntmA(1)       \istbA  \endist
\node at (cntm-1)
      [draw,circle,blue,opacity=.4,minimum size=1em]{};
\cntmAInfoset0(1)
\istrootcntmA(2)(1-1)  \istbA  \endist
\cntmAInfoset0(2)[60]
\end{istgame}
```

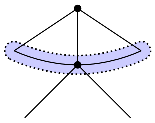


scaling `\cntmAInfoset0` according to Ti*k*Z scales

The height (or thickness) of an information set drawn by `\cntmAInfoset0` is not affected by Ti*k*Z scales. In some cases, you may want to change the height according to the values of Ti*k*Z scales.

As discussed in Section 10.5.2, the istgame environment treats, for example, `[scale=.7]` and `[scale=.7,xscale=1]` as different, while the tikzpicture environment treats them as equal. With the package istgame, if you use `[scale=.7,xscale=1]` then the value of `scale` is not used to keep the height of an oval type information set unchanged, so that you can scale the information set accordingly.

```
\begin{istgame}[scale=.7,xscale=1]
\istrootcntmA(0)  \istbA         \endist
\cntmAInfoset0[fill=blue!20](0)
\xtdistance{10mm}{20mm}
\istroot(1)(0-1)  \istb   \istb  \endist
\end{istgame}
```

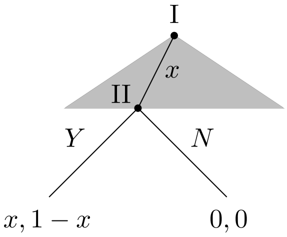


11.6 Transition from obsolete macros `\istcntm` and `\istcntmarc`

It turned out to be inconvenient to use the two macros `\istcntm` and `\istcntmarc` to express a continuum of branches because it requires two steps. Use, instead, a one step method (if you use only default settings) with `\istrootcntm` and `\istrootcntmA` that combine the two steps. Anyway, the two *obsolete* macros are provided only for backward compatibility with no maintenance and possible to be removed later.

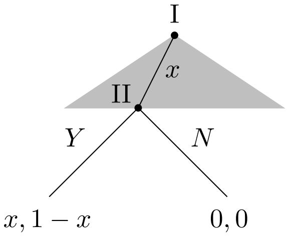
`\istcntm` The macro `\istcntm` draws only a background triangle for a continuum of branches. And then you can draw an action taken using `\istroot`.

```
% \istcntm
% syntax:
  \istcntm[<grow keyval>](<coor1>)(<coor2>)[<fill color>]+<levdist>..<sibdist>+
% defaults: [south]()(0,0)[black!25]+8mm..24mm+
% Example: \istcntm (obsolete)
\begin{istgame}[scale=1.2]
\istcntm(cntm)
\istroot(0)(cntm){I}+8mm..8mm+
  \istb{x}[r]  \istbm  \endist
\xtdistance{10mm}{20mm}
\istroot(1)(0-1)<[label distance=-3pt]120>{II}
  \istb{Y}[al]{x,1-x}  \istb{N}[ar]{0,0}  \endist
\end{istgame}
```



The above old way can be switched to the following new way:

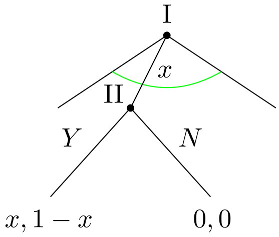
```
% Example: \istrootcntm (new way)
\begin{istgame}[scale=1.2]
\istrootcntm(0){I}
  \istb{x}[r]  \istbm  \endist
\xtdistance{10mm}{20mm}
\istroot(1)(0-1)<[label distance=-3pt]120>{II}
  \istb{Y}[al]{x,1-x}  \istb{N}[ar]{0,0}  \endist
\end{istgame}
```



`\istcntmarc` (arc version) The macro `\istcntmarc` is the arc version, which draws a background arc to represent a continuum of branches.

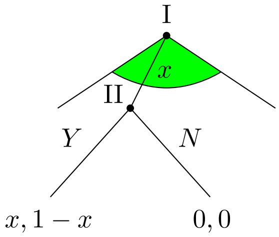
```
% \istcntmarc
% syntax:
  \istcntmarc[<grow keyval>](<coor1>)(<coor2>)[<color,opt>]{<num>}+levd..sibd+
% defaults: [south](<m>)(0,0)[bend right]{.5}+8mm..24mm+
```

```
% Example: \istcntmarc (obsolete)
\begin{istgame}[scale=1.2]
\istcntmarc(cntm)[green]
\istroot(0)(cntm){I}+8mm..8mm+
  \istb{x}[r]  \istbm  \endist
\xtdistance{10mm}{18mm}
\istroot(1)(0-1)<[label distance=-3pt]120>{II}
  \istb{Y}[al]{x,1-x}  \istb{N}[ar]{0,0}  \endist
\end{istgame}
```



The above old way can be switched to the following new way. One difference is that the position of a continuum arc is now `1` by default, while it was `.5`.

```
% Example: \istrootcntmA (new way)
\begin{istgame}[scale=1.2]
\cntmApreset<1.44>{.5}[green]
\istrootcntmA(0){I}
  \istb{x}[r]  \istbm  \endist
\xtdistance{10mm}{18mm}
\istroot(1)(0-1)<[label distance=-3pt]120>{II}
  \istb{Y}[al]{x,1-x}  \istb{N}[ar]{0,0}  \endist
\end{istgame}
```



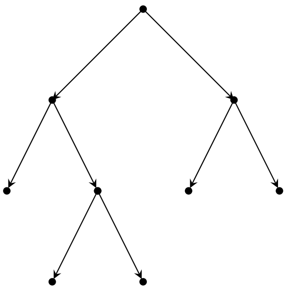
12 Arrows

12.1 Using Ti*k*Z arrow options with istgame

12.1.1 Ti*k*Z arrow option `->`

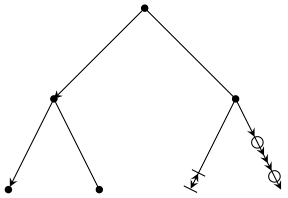
The simplest way of printing arrows in *a whole game tree* is to use the Ti*k*Z option `->` as an option for the istgame environment. To get the best result, you may want to use `\xtShowEndPoints` together with the option `->`.

```
% Example: arrows
\begin{istgame}[->,scale=.8]
\istroot(0)+15mm..30mm+  \istb \istb \endist
\xtShowEndPoints
\istroot(1)(0-1)         \istb \istb \endist
\istroot(2)(0-2)         \istb \istb \endist
\istroot(3)(1-2)         \istb \istb \endist
\end{istgame}
```



For an arrow on an individual branch, you can use arrow options with each `\istb`.

```
% Example: arrows
\begin{istgame}[scale=.8]
\istroot(0)+15mm..30mm+  \istb[->] \istb \endist
\xtShowEndPoints
\istroot(1)(0-1) \istb[->] \istb \endist
\xtHideEndPoints
\istroot(2)(0-2) \istb[-|<>|] \istb[->o>>>o>] \endist
\end{istgame}
```



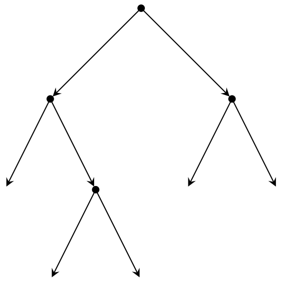
12.1.2 Fine-tuning with `\setistgameshorten`: Not for most users

Assuming that you do not want the terminal points to be shown, in order to get the best result of arrows you can use the macro `\setistgameshorten`, like `\setistgameshorten{1.3pt}`, right *before* the istgame environment. Note that `\setistgameshorten` has a global effect. So, when necessary, you should use the macro within a TeX group.

Remark: (not for most users)

- Internally, the istgame environment checks the existence of the option `->` and automatically adds `shorten >=<keyval>` (by default, `0pt`) to the option list as the first entry together with `->`.
- You can do the same thing, without using this macro, by manually adding, for example, `[->,shorten >=1.3pt]` to the list of options.

```
% Example: \setistgameshorten
\setistgameshorten{1.3pt} % use in a TeX group
\begin{istgame}[->,scale=.8]
%\xtShowEndPoints
\istroot(0)+15mm..30mm+  \istb \istb \endist
\istroot(1)(0-1)         \istb \istb \endist
\istroot(2)(0-2)         \istb \istb \endist
\istroot(3)(1-2)         \istb \istb \endist
\end{istgame}
```



You should also be aware that, in Ti*k*Z, the options `shorten` and `->` affect any lines or curves in the current environment.

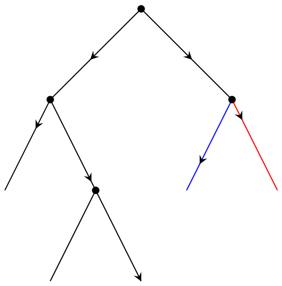
12.2 Middle arrows for a branch

With `\istb`, you can print an arrow tip in the middle of *a branch* by using a new *controllable* arrow tip style `->-`, like `\istb[->-]`.

12.2.1 A middle arrow for each `\istb`

To print an arrow tip in the middle of a branch, you can use `->-` as an option for `\istb`. The *controllable* middle arrow style `->-` takes one optional value, which is a number between 0 and 1 (by default `->-=.55`) determining a relative position from a parent node to a child node, where the end of an arrow tip is placed. For example, `\istb[->-=0.1]` prints an arrow tip near the parent node, and `\istb[->-=0.9]` near the child node.

```
% Example: new style: middle arrow tip
\begin{istgame}[scale=.8]
\istroot(0)+15mm..30mm+
  \istb[->-] \istb[->-] \endist
\istroot(1)(0-1)
  \istb[->-=.3] \istb[->-=.9] \endist
\istroot(2)(0-2)
  \istb[->-=.7,draw=blue] \istb[->-=.2,draw=red] \endist
\istroot(3)(1-2)
  \istb \istb[->-=1] \endist
\end{istgame}
```



Remark: issues on middle arrows with `\istB`:

- Middle arrows work well with `\istb`, but the exact position of a middle arrow tip does NOT work very well with `\istB`.
- So, if you want to print dual labels for branches with middle arrows, it is recommended to use `\istb` together with `\xtActionLabel`, instead of `\istB` (see also page 25).

Except for `\istB`, all the other variants of `\istb` including `\istbA`, `\cntmistb`, and `\cntmAistb` work well with the middle arrow option `->-`.

12.2.2 Middle arrow tip styles

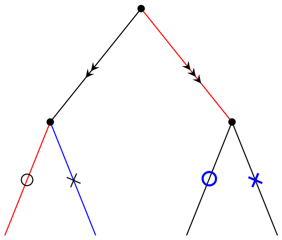
The package also provides additional styles for middle arrow tips, which have a *fixed* position and shape. (Note that the single arrow tip `->-` is *controllable*.)

Additional middle arrow tip styles:

- `->>-` : double arrow tip in the middle of a branch at the position of `.6`
- `->>>-` : triple arrow tip in the middle of a branch at the position of `.65`
- `-o-` : circle arrow tip in the middle of a branch at the position of `.55`
- `-x-` : cross arrow tip in the middle of a branch at the position of `.5`

Remark: All the middle arrow tips are printed in black, by default. Their color can be changed by `\setxtarrowtips` (see the following Section 12.2.3).

```
\begin{istgame}[xscale=.8]
\istroot(0)+15mm..30mm+
  \istb[->>-]     \istb[->>>-,red]  \endist
\istroot(1)(0-1)
  \istb[-o-,red] \istb[-x-,blue]   \endist
\setxtarrowtips[blue,thick]
\istroot(2)(0-2)
  \istb[-o-]     \istb[-x-]        \endist
\end{istgame}
```

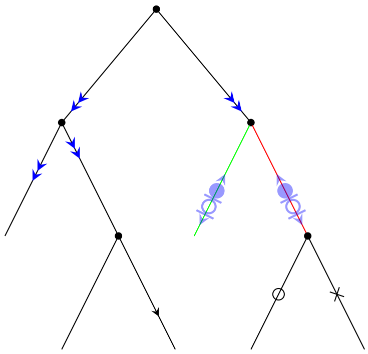


12.2.3 `\setxtarrowtips`: controlling middle arrow tips via `->-`

The macro `\setxtarrowtips` controls the position and shape of a middle arrow tip, but only works through the *controllable* middle arrow style `->-`, not the other *fixed* arrow styles. However, it controls the color of *all* the middle arrow tips. The effect of change by this macro continues until the end of the current environment unless it is changed again. To restore the default values, just declare `\setxtarrowtips` followed by nothing.

```
% \setxtarrowtips
% syntax: all arguments are optional
  \setxtarrowtips<midarrow tip pos>{<arrow shape>}[<arrow opt>]
% defaults: <.55>{>}[black,-,thin,solid,shorten >=0pt]
```

```
% Example: \setxtarrowtips
\begin{istgame}[scale=1]
\setxtarrowtips<.9>{>>}[blue,thick]
\istroot(0)+15mm..25mm+
  \istb[->-] \istb[->-] \endist
\istroot(1)(0-1)
  \istb[->-=.5] \istb[->-=.3] \endist
\setxtarrowtips<.9>{<*|o|>}[blue,thick,opacity=.4]
\istroot(2)(0-2)
  \istb[->-,green] \istb[->-,red] \endist
\setxtarrowtips<.7>
\istroot(3)(1-2)  \istb \istb[->-] \endist
\istroot(4)(2-2)  \istb[-o-] \istb[-x-] \endist
\end{istgame}
```



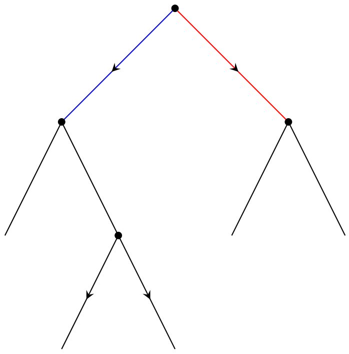
Note that, as seen in the example above, the position set individually by an option value of `\istb`, like `\istb[->-=.3]`, wins the position set by `\setxtarrowtips<.9>`.

12.3 Middle arrow for simple tress

12.3.1 `\xtShowMidArrows` and `\xtHideMidArrows`

The macro `\xtShowMidArrows` prints *middle arrows* (a single arrow tip, by default) for *all the branches* of a *simple tree* (but not for a separate branch). The effect of `\xtShowMidArrows` continues until it is changed by another `\xtShowMidArrows` or blocked by `\xtHideMidArrows`.

```
% Example: \xtShowMidArrows, \xtHideMidArrows
\begin{istgame}
\xtShowMidArrows
\istroot(0)+15mm..30mm+  \istb[blue] \istb[red] \endist
\xtHideMidArrows
\istroot(1)(0-1)  \istb \istb \endist
\istroot(2)(0-2)  \istb \istb \endist
\xtShowMidArrows
\istroot(3)(1-2)  \istb \istb \endist
\end{istgame}
```

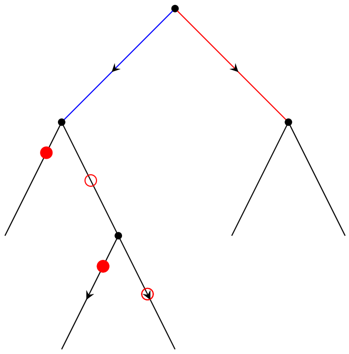


Remark:

- `\xtShowMidArrows` works for all branches of simple trees.
- The middle arrow option styles work separately only for `\istb`'s having them and `\setxtarrowtips` works only for `\istb`'s having the option `->-`.

- Since the middle arrow tip styles and `\xtShowMidArrows` work completely independently, if you use both of them, they are all printed.
- Note also that `\xtHideMidArrows` removes only the effect of `\xtShowMidArrows`.

```
% Example: \xtShowMidArrows, \xtHideMidArrows
\begin{istgame}
\setxtarrowtips<.3>{*}[red]
\xtShowMidArrows
\istroot(0)+15mm..30mm+  \istb[blue] \istb[red] \endist
\xtHideMidArrows
\istroot(1)(0-1)  \istb[->-] \istb[-o-] \endist
\istroot(2)(0-2)  \istb \istb \endist
\xtShowMidArrows
\istroot(3)(1-2)  \istb[->-] \istb[-o-] \endist
\end{istgame}
```

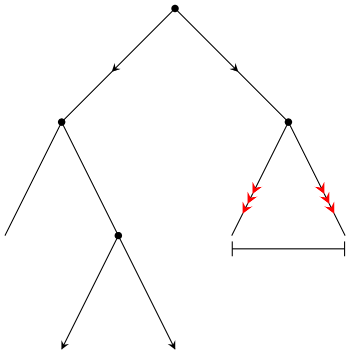


12.3.2 `\setxtshowmidarrows`: controlling `\xtShowMidArrows`

The macro `\setxtshowmidarrows`, you can control the position, color, style, and shape of middle arrow tips, to be drawn by `\xtShowMidArrows`.

```
% \setxtshowmidarrows
% syntax:
  \setxtshomidwarrows<midarrow tip relative pos>{<arrow shape>}[<arrow opt>]
% defaults: <.55>{>}[black]
```

```
% Example: \setxtshowmidarrows
\begin{istgame}
\xtShowMidArrows
\istroot(0)+15mm..30mm+
  \istb \istb \endist
\xtHideMidArrows
\istroot(1)(0-1)  \istb \istb \endist
\xtShowMidArrows
\setxtshowmidarrows<.8>{>>>}[red,thick]
\istroot(2)(0-2)  \istb \istb \endist
\setxtshowmidarrows<1>
\istroot(3)(1-2)  \istb \istb \endist
%----- Remark ---------------------------------------
%\draw ([yshift=-5pt]2-1) -- ([yshift=-5pt]2-2); % NO!
\coordinate (A) at ([yshift=-5pt]2-1);
\coordinate (B) at ([yshift=-5pt]2-2);
\draw [|-|] (A) -- (B);
\end{istgame}
```



Remark: issues in adding other graphic objects (not for most users)

- As you can see in the above example, when `\xtShowMidArrows` is used, you should redefine coordinates for endpoints before you use them for other graphic objects.
- This is because, in Ti*k*Z, `shift` or `scope` does not work in this case. So, alternatively, you can use `\draw [transform canvas={yshift=-5pt}] (2-1) -- (2-2);` (See Ti*k*Z manual about `transform canvas`).
- `\xtShowArrows` does not affect the Ti*k*Z absolute coordinates.

12.4 Arrows for simple trees

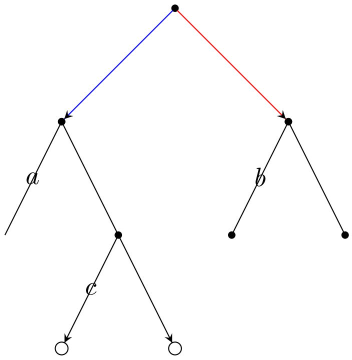
12.4.1 `\xtShowArrows` and `\xtHideArrows`

In many cases you might not need to care about the position of arrow tips, just wanting to put them at the ends of branches for simple trees. For this purpose, the macro `\xtShowArrows` (accepting one bracket option for a node style) is provided.

`\xtShowArrows`, internally, calls `\xtShowEndPoints` (with `solid node`, by default) first and then prints arrows in order to produce the best result of showing arrows. So, for example, `\xtShowArrows[oval node]` is equivalent to `\xtShowEndPoints[oval node]` followed by printing of arrows (in black by default).

The macro `\xtHideArrows` turns off all the effect of `\xtShowArrows`, while `\xtHideArrows*` has the endpoints remained. (`\xtShowArrows` does not have its starred version.)

```
% Example: \xtShowArrows, \xtHideArrows
\begin{istgame}
\xtShowArrows
\istroot(0)+15mm..30mm+  \istb[blue] \istb[red] \endist
\xtHideArrows
\istroot(1)(0-1)  \istb{a} \istb \endist
\xtHideArrows* % endpoints remain
\istroot(2)(0-2)  \istb{b} \istb \endist
\xtShowArrows[oval node]
\istroot(3)(1-2)  \istb{c} \istb \endist
\end{istgame}
```



Remark: issues in adding other graphic objects (not for most users)

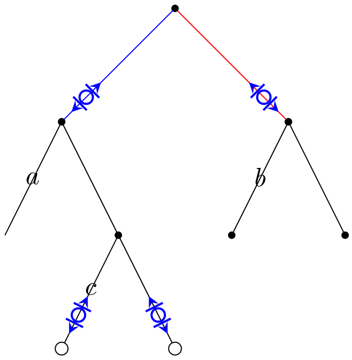
- Like `\xtShowMidArrows`, when `\xtShowArrows` is used, you should redefine coordinates for endpoints before you use them for other graphic objects, or use `transform canvas`.

12.4.2 `\setxtshowarrows`: controlling `\xtShowArrows`

The macro `\setxtshowarrows` controls the color and shape of arrow tips. You can even change the position of arrow tips by specifying the distance (`1.2pt` by default) from the endpoint as the last parenthesis optional argument, like `(5pt)`.

```
% \setxtshowarrows
% syntax:
  \setxtshowarrows[<arrow opt>]{<arrow shape>}(<arrow end shorten dim>)
% defaults: []{>}(1.2pt)
```

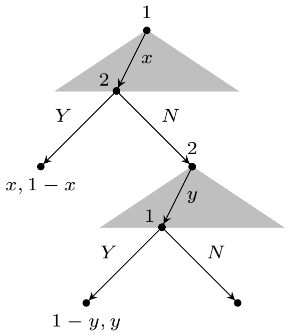
```
% Example: \xtShowArrows, \xtHideArrows
\begin{istgame}
\setxtshowarrows{<|o|>}[blue,thick](5pt)
\xtShowArrows
\istroot(0)+15mm..30mm+  \istb[blue] \istb[red] \endist
\xtHideArrows
\istroot(1)(0-1)  \istb{a} \istb \endist
\xtHideArrows* % endpoints remain
\istroot(2)(0-2)  \istb{b} \istb \endist
\xtShowArrows[oval node]
\istroot(3)(1-2)  \istb{c} \istb \endist
\end{istgame}
```



12.4.3 Arrows and continua of branches: examples (Not for most users)

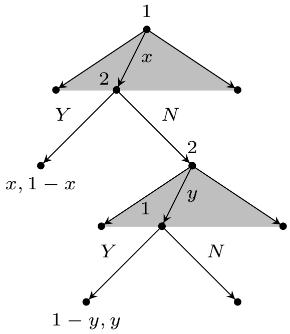
There is nothing special in using arrow tips in game trees having continua of branches.

```
\begin{istgame}[font=\scriptsize]
\xtdistance{10mm}{20mm}
\xtShowArrows
\istrootcntm(0)(0,0){1}  \istb{x}[r]  \istbm  \endist
\istroot(1)(0-1)<[label distance=-3pt]120>{2}
  \istb{Y}[al]{x,1-x}    \istb{N}[ar]         \endist
\istrootcntm(2)(1-2){2}  \istb{y}[r]  \istbm  \endist
\istroot(3)(2-1)<[label distance=-3pt]120>{1}
  \istb[]{Y}[al]{1-y,y}  \istb{N}[ar]         \endist
\end{istgame}
```



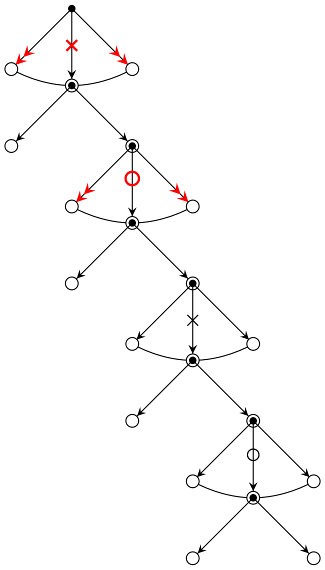
It is not impossible to put arrow tips on the outermost branches of a continuum. To do that, you can use `\cntmistb(*)` and `\cntmAistb(*)` together with the middle arrow tip styles and `\setxtarrowtips`.

```
\begin{istgame}[font=\scriptsize]
\xtdistance{10mm}{20mm}
\xtShowArrows
\setxtarrowtips<.97>
\cntmistb*[->-,draw]
\istrootcntm(0)(0,0){1}  \istb{x}[r]  \istbm  \endist
\istroot(1)(0-1)<[label distance=-3pt]120>{2}
  \istb{Y}[al]{x,1-x}    \istb{N}[ar]         \endist
\cntmistb*[->-,draw]
\istrootcntm(2)(1-2){2}  \istb{y}[r]  \istbm  \endist
\istroot(3)(2-1)<120>{1}
  \istb[]{Y}[al]{1-y,y}  \istb{N}[ar]         \endist
\end{istgame}
```



Here is one more example.

```
\begin{istgame}
\cntmdistance*{8mm}{16mm}
\xtShowArrows
\xtShowEndPoints*[oval node]
\setxtarrowtips<.93>{>>}[red,thick]
\cntmAistb[->-]
\istrootcntmA(1)          \istbA[-x-] \endist
\istroot(A1)(1-1)  \istb  \istb  \endist
\cntmAistb[->-]
\istrootcntmA(2)(A1-2)    \istbA[-o-] \endist
\istroot(A2)(2-1)  \istb  \istb  \endist
\setxtarrowtips<.93>
\cntmAistb[->-]
\istrootcntmA(3)(A2-2)    \istbA[-x-] \endist
\istroot(A3)(3-1)  \istb  \istb  \endist
\cntmAistb[->-]
\istrootcntmA(4)(A3-2)    \istbA[-o-] \endist
\istroot(A4)(4-1)  \istb  \istb  \endist
\end{istgame}
```



13 Supplementary macros

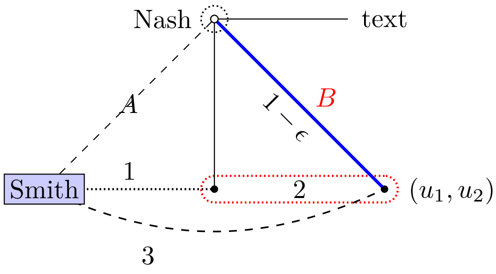
13.1 Supplementary macros to important labels

The istgame package also provides some supplementary macros to important labels: players, action labels, and payoffs. The supplementary macros work outside of simple trees.

`\xtOwner` puts the owner of a node (or a player), in text mode
`\xtOwner*` puts an owner in the input mode as set by `\setistmathTF(*)`
  syntax: `\xtOwner(<coor>){<owner>}[<pos>,<node opt>]`
  defaults: `(<m>){}[above]`
  (abbreviations for `<pos>` available)
`\xtActionLabel` puts an action label to a branch, in math mode
`\xtActionLabel*` puts an owner in the input mode as set by `\setistmathTF(*)`
  syntax: `\xtActionLabel[<opt>](<from>)(<to>){<action>}[<pos>,<node opt>]`
  defaults: `[-,draw=none](<m>)(<m>){}[black,text depth=.25ex]`
  (abbreviations for `<pos>` available)
`\xtPayoff` puts payoffs, in math mode
`\xtPayoff*` puts payoffs in the input mode as set by `\setistmathTF(*)`
  syntax: `\xtPayoff(<coor>){<payoff>}[<pos>,<node opt>]`
  defaults: `(<m>){}[inner sep=0pt,outer sep=4pt,text depth=.25ex]`
  (abbreviations for `<pos>` available)
`\xtInfosetOwner` puts the owner of an information set, in text mode
`\xtInfosetOwner*` puts an information set owner in the input mode as set by `\setistmathTF(*)`
  syntax: `\xtInfosetOwner(<from>)(<to>){<owner>}[<pos>,<node opt>]`
  defaults: `(<m>)(<m>){}[]`
  (abbreviations for `<pos>` available)
`\xtNode` puts ‘something,’ say an owner, into a node
  syntax: `\xtNode[<opt>](<coor>)[<node style>,<node opt>]{<node text>}`
  defaults: `[-](<m>)[solid node]{}`
`\xtNode*` puts ‘something’ into a plain node
  syntax: `\xtNode*(<coor>)[<node style>,<node opt>]{<node text>}`
  defaults: `(<m>)[plain node]{}`

The supplementary macros depend on the coordinates defined in the sequence of `\istroot`–`\istb`–`\endist` and print their corresponding objects on or around the specified coordinates. So, in the above supplementary macros, all the coordinates such as `(<from>)`, `(<to>)`, and `(<coor>)` are mandatory arguments, denoted by `<m>`.

```
\begin{istgame}[scale=1.5]
\istroot(0)[chance node]
  \istb[dashed]{A}  \istb*
  \istb*[blue,very thick]{B}[right,xshift=2pt,yshift=2pt,red]  \endist
\xtInfosetO(0)(0)
\xtInfoset[thick](0-1)(0-2){1}
\xtInfosetO[thick,red,rounded corners=.5em](0-2)(0-3){2}
\xtInfoset[dashed,bend right=25](0-1)(0-3)
%-----------------
\xtActionLabel(0)(0-3){1-\epsilon}[below,sloped] % action label in math mode
\xtInfosetOwner(0-1)(0-3){3}[xshift=-25pt,yshift=-25pt]
\xtOwner(0){Nash}[xshift=-5pt,left]
\xtPayoff(0-3){(u_1,u_2)}[right,xshift=5pt] % payoffs in math mode
\xtNode[dotted](0-1)[box node,fill=blue!20]{Smith}
%-----------------
\istroot[east](a)(0)[chance node]  \istb \endist
\xtNode*(a-1){text}
\end{istgame}
```



To specify the `[<pos>]` option for the above supplementary macros (other than `\xtNode`), you can use the abbreviations `[l]`, `[r]`, `[a]`, `[b]`, `[al]`, `[ar]`, `[bl]`, and `[br]`. Each of these abbreviations works only when used alone without any other option keys in the option list. For example, the option `[b]` or `[below]` or `[below,xshift=5pt]` works, but `[b,xshift=5pt]` does not, producing a compile error.

13.2 More supplementary macros

13.2.1 `\xtTimeLineH` and `\xtTimeLineV`

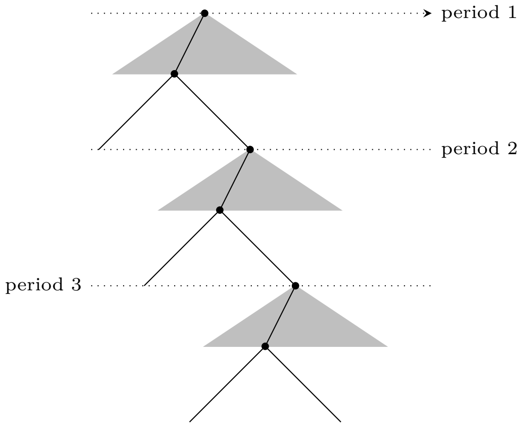
With the macro `\xtTimeLineH`, you can draw a horizontal time-line.

```
% \xtTimeLineH (same logic for \xtTimeLineV)
% syntax:  %% if tree swapped, the following means -<from> -<to>
  \xtTimeLineH[<opt>](<coor>){<from>}{<to>}{<label>}[<pos>,<opt>]
% defaults:
  [-,dotted,shorten <=0pt,shorten >=0pt](<m>){}{}[right]
```

For example, `\xtTimeLineH(coor){-1.5}{3}` draws a dotted (by default) horizontal time-line going through `(coor)` from `-1.5cm` to `3cm` with a label on the `right` (by default). Recall that, by default, the root of a tree is located at `(0,0)`. The two numbers determine the x-coordinates of a time-line. To change the position of the label, you can use the abbreviations, like `[a]` or `[ar]`.

The swap version `\xtTimeLineH'` simply puts the label on the other end of a time-line, `left` by default. That's the only difference to the normal version, `\xtTimeLineH`.

```
% Example: \xtTimeLineH('): not swapped
\begin{istgame}[font=\scriptsize]
\setistgrowdirection{south} % default
\xtdistance{10mm}{20mm}
\istrootcntm(1)        \istb \istbm \endist
\istroot(A1)(1-1)      \istb \istb  \endist
\istrootcntm(2)(A1-2)  \istb \istbm \endist
\istroot(A2)(2-1)      \istb \istb  \endist
\istrootcntm(3)(A2-2)  \istb \istbm \endist
\istroot(A3)(3-1)      \istb \istb  \endist
\xtTimeLineH[->](1){-1.5}{3}{period 1}
\xtTimeLineH(2){-1.5}{3}{period 2}
\xtTimeLineH'(3){-1.5}{3}{period 3}
\end{istgame}
```

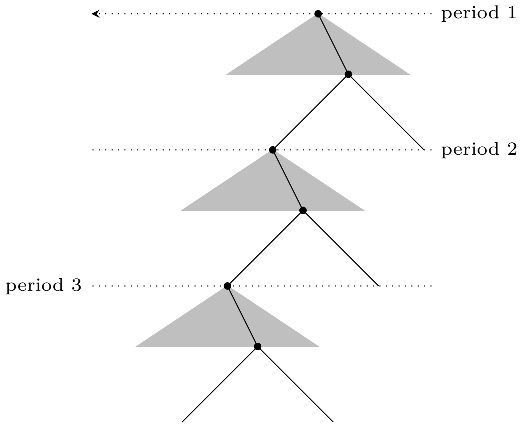


Remark: The direction to which a time-line goes depends on whether a tree is swapped or not.

- If branches are arranged `counterclockwise` (by default), a time-line goes *from left to right* or *from below to above*. As to `\xtTimeLineH`, we take this as a standard case: *left to right*.
- If a tree is *swapped*, so the branches are arranged `clockwise`, `\xtTimeLineH` draws a line going to the opposite direction: *from right to left* (for trees growing southwards). So, if a tree is swapped by `\setistgrowdirection'`, for example, `\xtTimeLine(coor){-1.5}{3}` draws a time-line going the to the opposite direction: from `1.5cm` to `-3cm`.

The following codes are the same as the above, except that the tree is swapped.

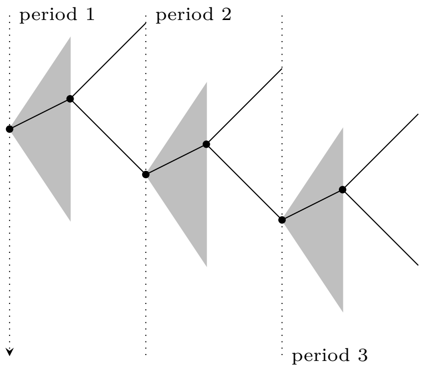
```
% Example: \xtTimeLineH: swapped
\begin{istgame}[font=\scriptsize]
\setistgrowdirection'{south} % clockwise
\xtdistance{10mm}{20mm}
\istrootcntm(1)       \istb \istbm \endist
\istroot(A1)(1-1)     \istb \istb  \endist
\istrootcntm(2)(A1-2) \istb \istbm \endist
\istroot(A2)(2-1)     \istb \istb  \endist
\istrootcntm(3)(A2-2) \istb \istbm \endist
\istroot(A3)(3-1)     \istb \istb  \endist
\xtTimeLineH[->](1){-1.5}{3}{period 1}
\xtTimeLineH(2){-1.5}{3}{period 2}
\xtTimeLineH'(3){-1.5}{3}{period 3}
\end{istgame}
```



With the macro `\xtTimeLineV`, you can draw a vertical time-line with a label on the top right, by default. As to `\xtTimeLineV`, we take a time-line going *from above to below* as a standard case, because it is natural. So, if a tree grows eastwards, the *swapped tree* (arranging branches *from above to below*) is taken as a standard case for `\xtTimeLineV`.

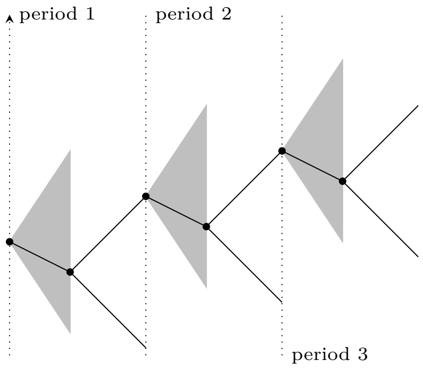
The swap version `\xtTimeLineV'` simply puts the label on the other side of a time-line. That's the only difference to the normal version, `\xtTimeLineV`.

```
% Example: \xtTimeLineV('): swapped
\begin{istgame}[font=\scriptsize]
\setistgrowdirection'{east} % clockwise
\xtdistance{10mm}{20mm}
\istrootcntm(1)       \istb \istbm \endist
\istroot(A1)(1-1)     \istb \istb  \endist
\istrootcntm(2)(A1-2) \istb \istbm \endist
\istroot(A2)(2-1)     \istb \istb  \endist
\istrootcntm(3)(A2-2) \istb \istbm \endist
\istroot(A3)(3-1)     \istb \istb  \endist
\xtTimeLineV[->](1){1.5}{-3}{period 1}
\xtTimeLineV(2){1.5}{-3}{period 2}
\xtTimeLineV'(3){1.5}{-3}{period 3}
\end{istgame}
```



The following codes are the same as the above, except that the tree is *not* swapped.

```
\begin{istgame}[font=\scriptsize]
\setistgrowdirection{east} % counterclockwise
\xtdistance{10mm}{20mm}
\istrootcntm(1)       \istb \istbm \endist
\istroot(A1)(1-1)     \istb \istb  \endist
\istrootcntm(2)(A1-2) \istb \istbm \endist
\istroot(A2)(2-1)     \istb \istb  \endist
\istrootcntm(3)(A2-2) \istb \istbm \endist
\istroot(A3)(3-1)     \istb \istb  \endist
\xtTimeLineV[->](1){1.5}{-3}{period 1}
\xtTimeLineV(2){1.5}{-3}{period 2}
\xtTimeLineV'(3){1.5}{-3}{period 3}
\end{istgame}
```



In the example above, since the tree is not swapped, `\xtTimeLineV(coor){1.5}{-3}` draws a vertical time-line going to the opposite direction: from `-1.5cm` to `3cm`.

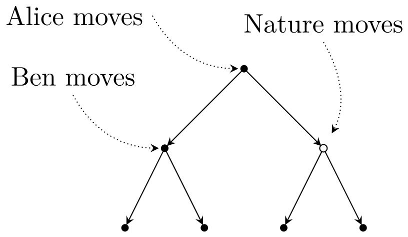
13.2.2 `\xtCommentTo` and `\xtCommentFrom`

With the macro `\xtCommentTo`, you can attach a comment to a node from a coordinate *relative* to the node with a densely dotted arrow line (by default). Note that with `\xtCommentTo` you need to specify a *target absolute* coordinate followed by a *start* ***relative*** coordinate.

```
% \xtCommentTo
% syntax:
  \xtCommentTo[<line opt>](<absolute to-coor>)(<relative from-coor>){<comment>}
              [<pos>,<node opt>](<end-shorten dim>)
% defaults:
  [densely dotted,->,>=stealth,shorten >=1pt](<m>)(<m>){}[above](1pt)
```

The last parenthesis option if for shortening arrow end. You can use the abbreviations of the directional words, but not with other options in the option list.

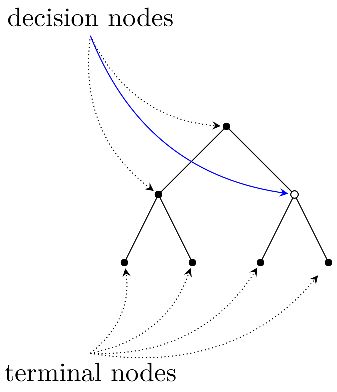
```
% Example: \xtCommentTo
\begin{istgame}[->,scale=.7]
\xtShowEndPoints
\istroot(0)+15mm..30mm+       \istb \istb \endist
\istroot(1)(0-1)              \istb \istb \endist
\istroot(2)(0-2)[chance node] \istb \istb \endist
\xtCommentTo[bend right](0)(150:2){Alice moves}[l]
\xtCommentTo[bend right](1)(150:2){Ben moves}
\xtCommentTo[bend left](2)(0,2){Nature moves}[a](5pt)
\end{istgame}
```



The macro `\xtCommentFrom` enables you to attach a comment to one or more nodes from an *absolute* coordinate with a densely dotted arrow line. Note that with `\xtCommentFrom` you need to specify a *start absolute* coordinate followed by a *target absolute* coordinate. The last parenthesis option if for shortening arrow end. You can use the abbreviations of the directional words.

```
% \xtCommentFrom
% syntax:
  \xtCommentFrom[<line opt>](<from>)(<to>){<comment>}[<pos>,<node opt>]<arrow end
    shorten>
% defaults:
  [densely dotted,->,>=stealth,shorten >=1pt](<m>)(<m>){}[above](1pt)
```

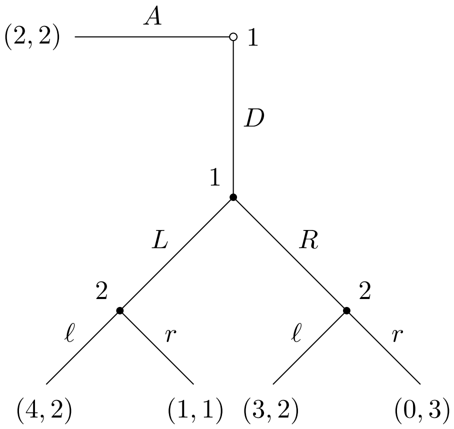
```
% Example: \xtCommentFrom
\begin{istgame}[scale=.6]
\xtShowEndPoints
\istroot(0)+15mm..30mm+
  \istb \istb \endist
\istroot(1)(0-1)
  \istb \istb \endist
\istroot(2)(0-2)[chance node]
  \istb \istb \endist
\xtCommentFrom[bend right](-3,2)(0){decision nodes}
\xtCommentFrom[bend right](-3,2)(1)
\xtCommentFrom[bend right,solid,blue](-3,2)(2)
\xtCommentFrom[bend right](-3,-5)(1-1)
               {terminal nodes}[b]
\xtCommentFrom[bend right](-3,-5)(1-2)
\xtCommentFrom[bend right](-3,-5)(2-1)
\xtCommentFrom[bend right](-3,-5)(2-2)(5pt)
\end{istgame}
```



14 Representing subgames

Here is an example of a whole game tree.

```
\begin{istgame}
\xtdistance{15mm}{30mm}
\istroot[-135](0)[initial node]<0>{1}
  \istb{A}[a]{(2,2)}[l]           \istb{D}[r]  \endist
\istroot(1)(0-2)<135>{1}
                   \istb{L}[al]  \istb{R}[ar]  \endist
\xtdistance{10mm}{20mm}
\istroot(2)(1-1)<135>{2}
  \istb{\ell}[al]{(4,2)}  \istb{r}[ar]{(1,1)}  \endist
\istroot(3)(1-2)<45>{2}
  \istb{\ell}[al]{(3,2)}  \istb{r}[ar]{(0,3)}  \endist
\end{istgame}
```



14.1 `\xtSubgameBox` (experimental)

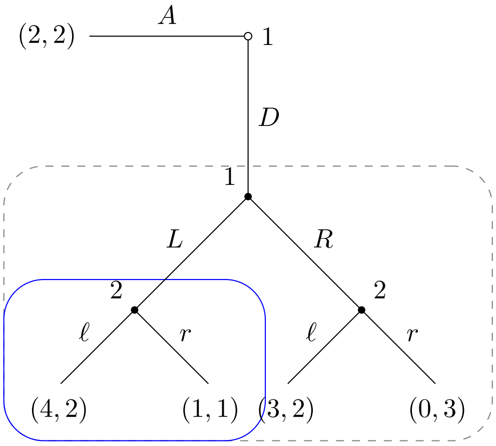
14.1.1 `\xtSubgameBox`

`\xtSubgameBox` is for indicating a `subgame` by a box (or a rectangle) with rounded corners.

```
% \xtSubgameBox(*)
% syntax:
  \xtSubgame(*)(<subroot coor>){(<coor1>) (<coor2>) ... }[<opt>]
% options:
  *: filled box
  (<subroot coor>): the subroot of a subgame (mandatory)
  {(coor1) (coor2) ...}: coordinates of terminal nodes (mandatory)
  [<opt>]: color, line style
% defaults:
  (<m>){}[rectangle,dashed,inner sep=20pt,rounded corners=15pt,black!50]
  *(<m>){}[rectangle,inner sep=20pt,rounded corners=15pt,red!20]
```

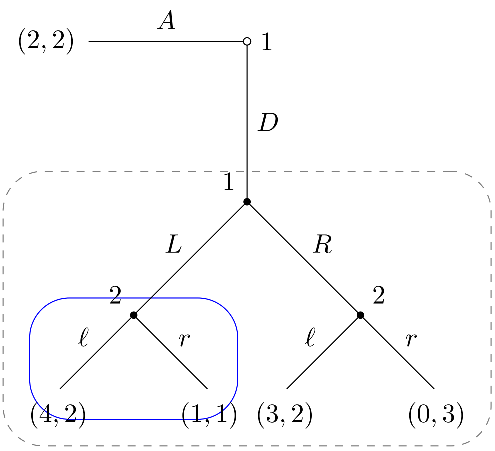
The subgame box embraces the subroot indicated in parentheses and terminal nodes specified within curly braces. The coordinates of terminal nodes are listed in curly braces without delimiters (spaces are allowed). The default options for the subgame box are `dashed`, `inner sep=20pt`, `rounded corners=15pt`, and `black!50`.

```
% subgame: rectangle
\begin{istgame}
\xtdistance{15mm}{30mm}
\istroot[-135](0)[initial node]<0>{1}
  \istb{A}[a]{(2,2)}[l]           \istb{D}[r]  \endist
\istroot(1)(0-2)<135>{1}
                   \istb{L}[al]  \istb{R}[ar]  \endist
\xtdistance{10mm}{20mm}
\istroot(2)(1-1)<135>{2}
  \istb{\ell}[al]{(4,2)}  \istb{r}[ar]{(1,1)}  \endist
\istroot(3)(1-2)<45>{2}
  \istb{\ell}[al]{(3,2)}  \istb{r}[ar]{(0,3)}  \endist
\xtSubgameBox(1){(2-1)(3-2)}
\xtSubgameBox(2){(2-1)(2-2)}[solid,blue]
\end{istgame}
```



You can change the size of the box in two ways. First, you can change it by using `inner sep`, `inner xsep`, or `inner ysep`. Secondly, you can shift the subroot, like `([xshift=5pt]1)` when the subroot is `(1)`. Of course, you can do the both at the same time.

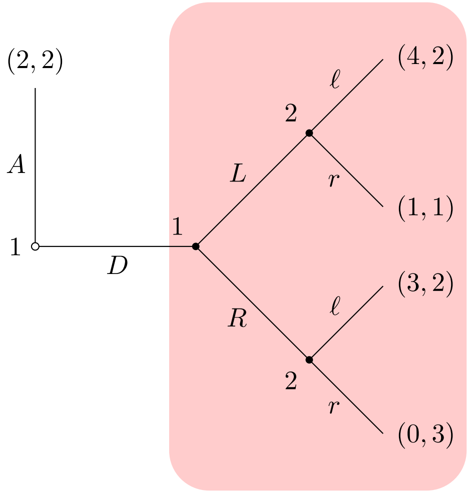
```
\begin{istgame}
\xtdistance{15mm}{30mm}
\istroot[-135](0)[initial node]<0>{1}
           \istb{A}[a]{(2,2)}[l]  \istb{D}[r]  \endist
\istroot(1)(0-2)<135>{1}
                   \istb{L}[al]   \istb{R}[ar]  \endist
\xtdistance{10mm}{20mm}
\istroot(2)(1-1)<135>{2}
  \istb{\ell}[al]{(4,2)}  \istb{r}[ar]{(1,1)}  \endist
\istroot(3)(1-2)<45>{2}
  \istb{\ell}[al]{(3,2)}  \istb{r}[ar]{(0,3)}  \endist
\xtSubgameBox(1){(2-1)(3-2)}
\xtSubgameBox([yshift=5pt]2){(2-1)(2-2)}%
    [solid,blue,inner sep=10pt]
\end{istgame}
```



14.1.2 `\xtSubgameBox*`

The starred version `\xtSubgameBox*` produces a subgame box with background color (by default `red!20`). In the following example, where the tree grows east, the size of the filled subgame box is adjusted by manipulating `inner sep` and `xshift`.

```
% subgame: rectangle*
\begin{istgame}
\setistgrowdirection'{east}
\xtdistance{15mm}{30mm}
\istroot[45](0)[initial node]<180>{1}
          \istb{A}[l]{(2,2)}[a]  \istb{D}[b]   \endist
\istroot(1)(0-2)<135>{1}
                   \istb{L}[al]   \istb{R}[bl]  \endist
\xtdistance{10mm}{20mm}
\istroot(2)(1-1)<135>{2}
  \istb{\ell}[al]{(4,2)}  \istb{r}[bl]{(1,1)}  \endist
\istroot(3)(1-2)<-135>{2}
  \istb{\ell}[al]{(3,2)}  \istb{r}[bl]{(0,3)}  \endist
\xtSubgameBox*([xshift=20pt]1){(2-1)(3-2)}%
    [inner xsep=30pt]
\end{istgame}
```



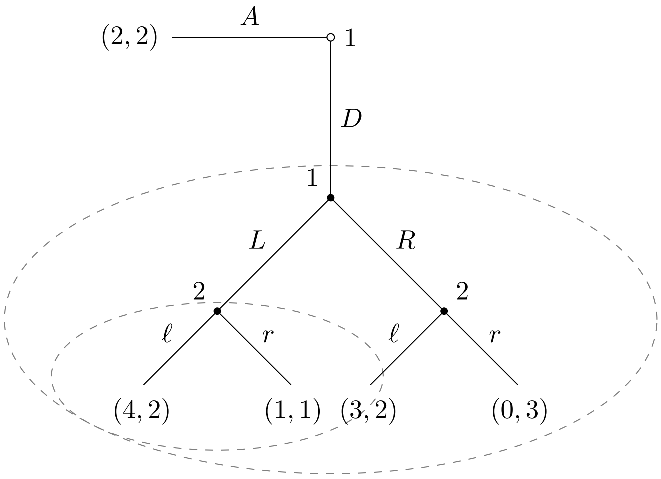
14.2 `\xtSubgameOval` and `\xtSubgameOval*` (experimental)

`\xtSubgameOval` and `\xtSubgameOval*` are also provided.

```
% \xtSubgameOval(*)
% syntax:
  \xtSubgameOval(*)(<subroot coor>){(<coor1>) (<coor2>) ... }[<opt>]
% options:
  *: filled box
  (<subroot coor>): the subroot of a subgame (mandatory)
  {(coor1) (coor2) ...}: coordinates of terminal nodes (mandatory)
  [<opt>]: color, line style
% defaults:
  (<m>){}[ellipse,dashed,inner sep=20pt,rounded corners=15pt,black!50]
  *(<m>){}[ellipse,inner sep=20pt,rounded corners=15pt,red!20]
```

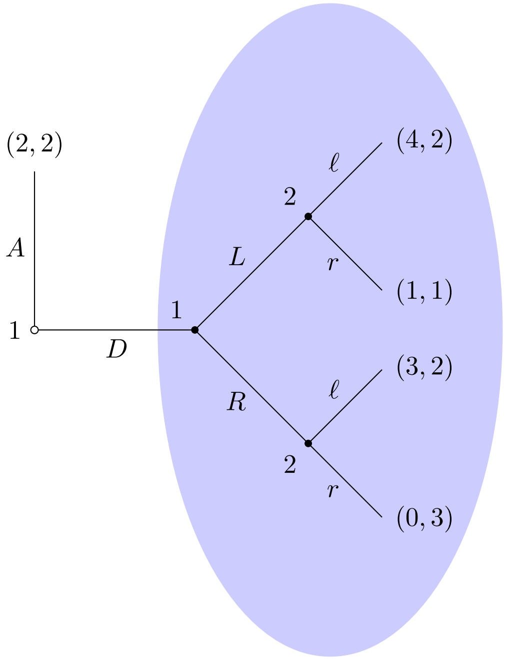
`\xtSubgameOval(*)` works just like `\xtSubgameBox(*)`, except for one thing: an oval instead of a rectangle. `\xtSubgameOval` draws an oval (or an ellipse) to represent a subgame, and `\xtSubgameOval*` draws a filled oval with `red!20` by default.

```
% subgame: ellipse
\begin{istgame}
\xtdistance{15mm}{30mm}
\istroot[-135](0)[initial node]<0>{1}
  \istb{A}[a]{(2,2)}[l]
  \istb{D}[r]
  \endist
\istroot(1)(0-2)<135>{1}
  \istb{L}[al]
  \istb{R}[ar]
  \endist
\xtdistance{10mm}{20mm}
\istroot(2)(1-1)<135>{2}
  \istb{\ell}[al]{(4,2)}
  \istb{r}[ar]{(1,1)}
  \endist
\istroot(3)(1-2)<45>{2}
  \istb{\ell}[al]{(3,2)}
  \istb{r}[ar]{(0,3)}
  \endist
\xtSubgameOval(1){(2-1)(3-2)}
\xtSubgameOval(2){(2-1)(2-2)}
\end{istgame}
```



The way of changing the size of the subgame oval is the same as with `\xtSubgameBox`: first, changing inner sep, and secondly, shifting the subroot. In the example below, the options `inner xsep=25pt` and `([xshift=30pt]1)` are used to change the size of the subgame oval. Its color is changed by `blue!20`.

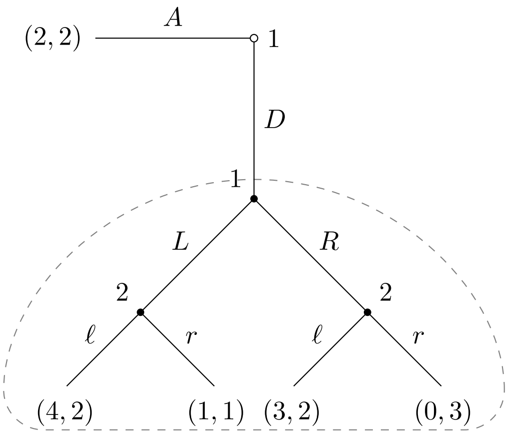
```
% subgame: ellipse*
\begin{istgame}
\setistgrowdirection'{east}
\xtdistance{15mm}{30mm}
\istroot[45](0)[initial node]<180>{1}
  \istb{A}[l]{(2,2)}[a]
  \istb{D}[b]
  \endist
\istroot(1)(0-2)<135>{1}
  \istb{L}[al]
  \istb{R}[bl]
  \endist
\xtdistance{10mm}{20mm}
\istroot(2)(1-1)<135>{2}
  \istb{\ell}[al]{(4,2)}
  \istb{r}[bl]{(1,1)}
  \endist
\istroot(3)(1-2)<-135>{2}
  \istb{\ell}[al]{(3,2)}
  \istb{r}[bl]{(0,3)}
  \endist
\xtSubgameOval*([xshift=30pt]1){(2-1)(3-2)}%
    [inner xsep=25pt,blue!20]
\end{istgame}
```



14.3 `\setxtsubgamelayer`

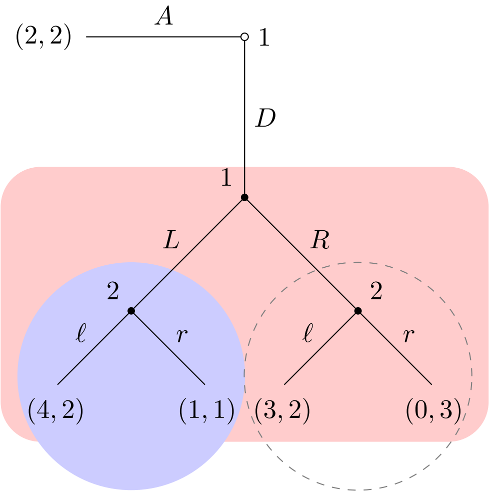
You can change the shape of a subgame representation to other shapes such as `semicircle` and `circle` or even `star`. You will need to change the size of the shape by using `inner sep` and by shifting the subroot to get the result you want.

```
% subgame: half circle
\begin{istgame}
\xtdistance{15mm}{30mm}
\istroot[-135](0)[initial node]<0>{1}
  \istb{A}[a]{(2,2)}[l]
  \istb{D}[r]    \endist
\istroot(1)(0-2)<135>{1}
  \istb{L}[al]
  \istb{R}[ar]  \endist
\xtdistance{10mm}{20mm}
\istroot(2)(1-1)<135>{2}
  \istb{\ell}[al]{(4,2)}
  \istb{r}[ar]{(1,1)}
  \endist
\istroot(3)(1-2)<45>{2}
  \istb{\ell}[al]{(3,2)}
  \istb{r}[ar]{(0,3)}
  \endist
\xtSubgameBox([yshift=-2cm]1){(2-1)(3-2)}%
    [semicircle,inner sep=15pt]
\end{istgame}
```



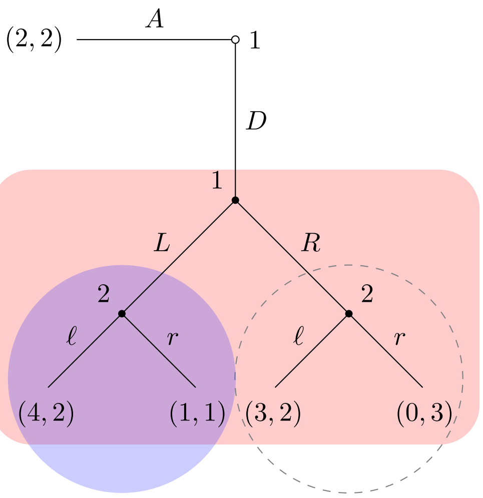
Here are more examples showing how to draw a rectangle and circles to represent subgames.

```
% subgame: rectangle*
\begin{istgame}
\xtdistance{15mm}{30mm}
\istroot[-135](0)[initial node]<0>{1}
  \istb{A}[a]{(2,2)}[l]
  \istb{D}[r]    \endist
\istroot(1)(0-2)<135>{1}
  \istb{L}[al]
  \istb{R}[ar]  \endist
\xtdistance{10mm}{20mm}
\istroot(2)(1-1)<135>{2}
  \istb{\ell}[al]{(4,2)}
  \istb{r}[ar]{(1,1)}
  \endist
\istroot(3)(1-2)<45>{2}
  \istb{\ell}[al]{(3,2)}
  \istb{r}[ar]{(0,3)}
  \endist
\xtSubgameBox*(1){(2-1)(3-2)}
\xtSubgameOval*(2){(2-1)(2-2)}%
  [circle,blue!20,inner xsep=9pt]
\xtSubgameOval(3){(3-1)(3-2)}%
  [circle,inner xsep=9pt]
\end{istgame}
```



You can use `\setxtsubgamelayer` to change the layer of a subgame from `background` to `behind`, `main`, `above`, or `foreground`, in that order. In order to go back to the default layer, you can just declare `\setxtsubgamelayer` or do `\setxtsubgamelayer{}`.

```
% subgame: \setxtsubgamelayer
\begin{istgame}
\xtdistance{15mm}{30mm}
\istroot[-135](0)[initial node]<0>{1}
  \istb{A}[a]{(2,2)}[l]
  \istb{D}[r]   \endist
\istroot(1)(0-2)<135>{1}
  \istb{L}[al]
  \istb{R}[ar]  \endist
\xtdistance{10mm}{20mm}
\istroot(2)(1-1)<135>{2}
  \istb{\ell}[al]{(4,2)}
  \istb{r}[ar]{(1,1)}
  \endist
\istroot(3)(1-2)<45>{2}
  \istb{\ell}[al]{(3,2)}
  \istb{r}[ar]{(0,3)}
  \endist
\setxtsubgamelayer{behind}
\xtSubgameOval*(2){(2-1)(2-2)}%
  [circle,blue!80,inner xsep=9pt,opacity=.25]
\setxtsubgamelayer % restore defaults
\xtSubgameBox*(1){(2-1)(3-2)}
\xtSubgameOval(3){(3-1)(3-2)}%
  [circle,inner xsep=9pt]
\end{istgame}
```

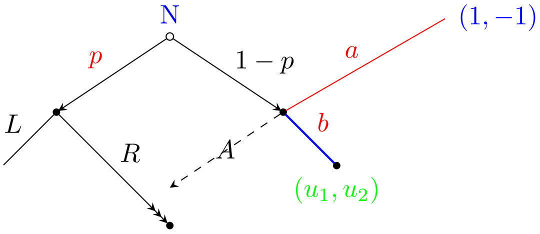


15 Miscellany

15.1 Various branch types, directions, and lengths

Here is an example of how to deal with various line types, directions, and lengths of branches. You can do this using `\istbA` (see Section 5.5, for more details).

```
\begin{istgame}
\istroot(0)[chance node]<[blue]>{N}+10mm..30mm+
  \istb[->,shorten >=.8pt]{p}[above left,red]
  \istb[->,shorten >=.8pt]{1-p}[ar]  \endist
\istroot(1)(0-1)+15mm..30mm+
  \istb<level distance=7mm,sibling distance=14mm>{L}[al]
  \istb*[->>>]{R}[ar]  \endist
\istroot(2)(0-2)+10mm..15mm+
  \istb[->,dashed]{A}
  \istbA(2.5)<grow=30>[red]{a}[al]{(1,-1)}[[blue]right]
  \istb*<grow=-45>[blue,text=red,thick]{b}[ar]{(u_1,u_2)}[[green]-90]  \endist
\end{istgame}
```

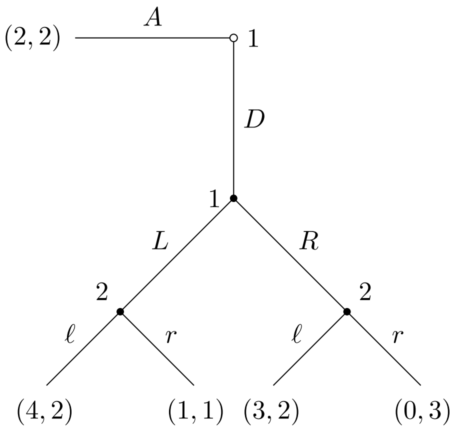


15.2 Code reuse

15.2.1 Drawing subgames

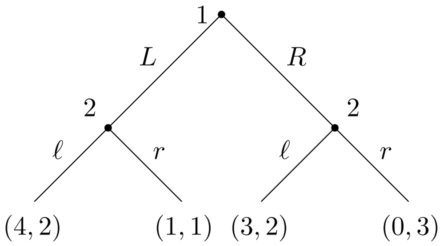
Suppose that you have the following tree as a whole game.

```
% Example: a whole game
\begin{istgame}
\xtdistance{15mm}{30mm}
\istroot[-135](0)[initial node]<0>{1}
  \istb{A}[a]{(2,2)}[l]
  \istb{D}[r]
  \endist
\istroot(1)(0-2)<left>{1}
  \istb{L}[al]
  \istb{R}[ar]
  \endist
\xtdistance{10mm}{20mm}
\istroot(2)(1-1)<135>{2}
  \istb{\ell}[al]{(4,2)}
  \istb{r}[ar]{(1,1)}
  \endist
\istroot(3)(1-2)<45>{2}
  \istb{\ell}[al]{(3,2)}
  \istb{r}[ar]{(0,3)}
  \endist
\end{istgame}
```



It is easy to draw a subgame if you use the existing codes for a whole game. To get a subgame, just comment out the codes for the rest of the subgame.

```
% Example: a subgame
\begin{istgame}
\xtdistance{15mm}{30mm}
%\istroot[-135](0)[initial node]<0>{1}
%  \istb{A}[a]{(2,2)}[l]
%  \istb{D}[r]
%\endist
\istroot(1)(0-2)<left>{1}
  \istb{L}[al]
  \istb{R}[ar]
  \endist
\xtdistance{10mm}{20mm}
\istroot(2)(1-1)<135>{2}
  \istb{\ell}[al]{(4,2)}
  \istb{r}[ar]{(1,1)}
  \endist
\istroot(3)(1-2)<45>{2}
  \istb{\ell}[al]{(3,2)}
  \istb{r}[ar]{(0,3)}
  \endist
\end{istgame}
```



Remark: If you copy and paste the previous codes alone and try to complie to get a subgame, you will get an error message:

```
! Package pgf Error: No shape named 0-2 is known.
```

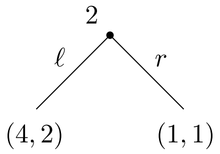
This is because the first active `\istroot` refers to the undefined coordinate `(0-2)`. In this case you need to change the line as:

```
\istroot(1)(0,0)<left>{1}
```

However, if you have the codes for a whole game before the codes with some lines commented out for a subgame, you will not get the error message when you compile the file (because the coordinate `(0-2)` is already defined in the whole game). Therefore, most of the time you will not get the error message by commenting out some lines to get a subgame.

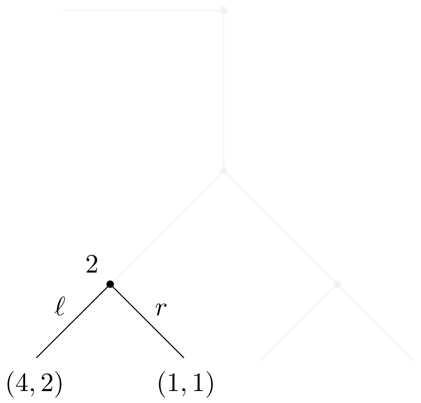
A smaller subgame is also easy to get by commenting out the rest of the subgame you want.

```
% Example: a smaller subgame
\begin{istgame}
%\xtdistance{15mm}{30mm}
%\istroot[-135](0)[initial node]<0>{1}
%  \istb{A}[a]{(2,2)}[l]
%  \istb{D}[r]
%\endist
%\istroot(1)(0-2)<left>{1}
%  \istb{L}[al]
%  \istb{R}[ar]
%\endist
\xtdistance{10mm}{20mm}
\istroot(2)(1-1)<135>{2}
  \istb{\ell}[al]{(4,2)}
  \istb{r}[ar]{(1,1)}
  \endist
%\istroot(3)(1-2)<45>{2}
%  \istb{\ell}[al]{(3,2)}
%  \istb{r}[ar]{(0,3)}
%\endist
\end{istgame}
```



It is not possible to make a subgame stay put where it appears in a whole game. (Replace `[black!5]` by `[white]` to make the rest of the game disappear in the example below.)

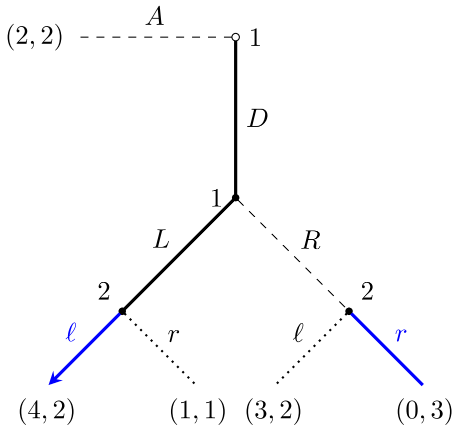
```
% Example: a subgame stays put
\begin{istgame}
\setistDecisionNodeStyle[black!5]
\xtdistance{15mm}{30mm}
\istroot[-135](0)%[initial node]<0>{1}
  \istb[black!5]%{A}[a]{(2,2)}[l]
  \istb[black!5]%{D}[r]
  \endist
\istroot(1)(0-2)%<left>{1}
  \istb[black!5]%{L}[al]
  \istb[black!5]%{R}[ar]
  \endist
\setistDecisionNodeStyle
\xtdistance{10mm}{20mm}
\istroot(2)(1-1)<135>{2}
  \istb{\ell}[al]{(4,2)}
  \istb{r}[ar]{(1,1)}
  \endist
\setistDecisionNodeStyle[black!5]
\istroot(3)(1-2)%<45>{2}
  \istb[black!5]%{\ell}[al]{(3,2)}
  \istb[black!5]%{r}[ar]{(0,3)}
  \endist
\end{istgame}
```



15.2.2 Backward induction

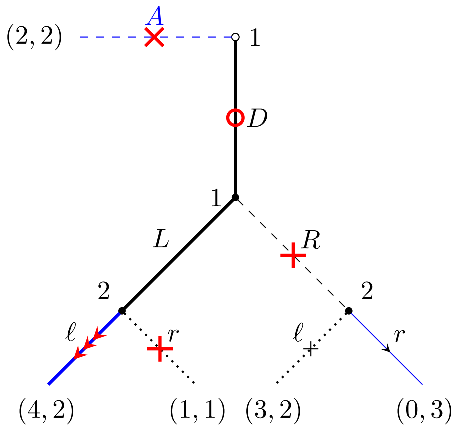
It is also easy to illustrate the procedure of backward induction, when you analyze an extensive game.

```
\begin{istgame}[>=stealth]
\xtdistance{15mm}{30mm}
\istroot[-135](0)[initial node]<0>{1}
  \istb[dashed]{A}[a]{(2,2)}[l]
  \istb[very thick]{D}[r]
\endist
\istroot(1)(0-2)<left>{1}
  \istb[very thick]{L}[al]
  \istb<dashed>{R}[ar]
\endist
\xtdistance{10mm}{20mm}
\istroot(2)(1-1)<135>{2}
  \istb[->,very thick,blue]{\ell}[al]{(4,2)}
  \istb[thick,dotted]{r}[ar]{(1,1)}
\endist
\istroot(3)(1-2)<45>{2}
  \istb[thick,dotted]{\ell}[al]{(3,2)}
  \istb[very thick,blue]{r}[ar]{(0,3)}
\endist
\end{istgame}
```



By using `\istb` with options for the middle arrow tips, you can effectively express the backward induction procedure. The command `\setxtarrowtips` controls the style of arrow tips and just declaring `\setxtarrowtips` followed by nothing restores defaults for the middle arrow tips styles. See Section 12, for more details. change the color and the style of a line representing a branch and an arrow tip to it.

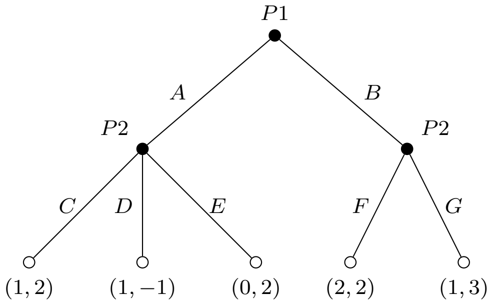
```
% Example: backward induction
\begin{istgame}
\setxtarrowtips[red,very thick,shorten
    >=-2pt,shorten <=-2pt]
\xtdistance{15mm}{30mm}
\istroot[-135](0)[initial node]<0>{1}
  \istb[-x-,dashed,blue]{A}[a]{(2,2)}[l]
  \istb[-o-,very thick]{D}[r]
  \endist
\istroot(1)(0-2)<left>{1}
  \istb[very thick]{L}[al]
  \istb[dashed,-x-]{R}[ar]
  \endist
\xtdistance{10mm}{20mm}
\istroot(2)(1-1)<135>{2}
  \istb[->>>-,very
    thick,draw=blue]{\ell}[al]{(4,2)}
  \istb[-x-,thick,dotted]{r}[ar]{(1,1)}
  \endist
\setxtarrowtips % restore defaults
\istroot(3)(1-2)<45>{2}
  \istb[-x-,thick,dotted]{\ell}[al]{(3,2)}
  \istb[->-,draw=blue]{r}[ar]{(0,3)}
  \endist
\end{istgame}
```



15.2.3 Code reusability

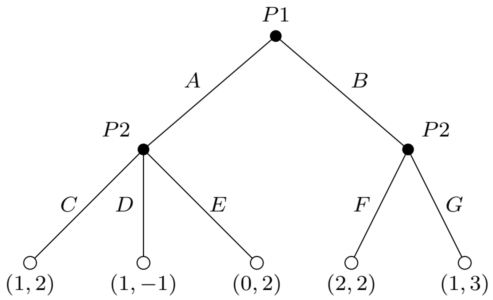
Chen(2013)'s work is good to understand how drawing a game tree with Ti*k*Z works.

```
% Chen (2013)
\begin{tikzpicture}[scale=1,font=\footnotesize]
\tikzstyle{solid node}=[circle,draw,inner sep=1.5,fill=black]
\tikzstyle{hollow node}=[circle,draw,inner sep=1.5]
\tikzstyle{level 1}=[level distance=15mm,sibling distance=3.5cm]
\tikzstyle{level 2}=[level distance=15mm,sibling distance=1.5cm]
\tikzstyle{level 3}=[level distance=15mm,sibling distance=1cm]
\node(0)[solid node,label=above:{$P1$}]{}
child{node[solid node,label=above left:{$P2$}]{}
child{node[hollow node,label=below:{$(1,2)$}]{} edge from parent node[left]{$C$}}
child{node[hollow node,label=below:{$(1,-1)$}]{} edge from parent node[left]{$D$}}
child{node[hollow node,label=below:{$(0,2)$}]{} edge from parent node[right]{$E$}}
edge from parent node[left,xshift=-5]{$A$}
}
child{node[solid node,label=above right:{$P2$}]{}
child{node[hollow node,label=below:{$(2,2)$}]{} edge from parent node[left]{$F$}}
child{node[hollow node,label=below:{$(1,3)$}]{} edge from parent node[right]{$G$}}
edge from parent node[right,xshift=5]{$B$}
};
\end{tikzpicture}
```



You can get a quite similar (actually the same) result by using the istgame package.

```
% istgame
\begin{istgame}[scale=1,font=\footnotesize]
\setistDecisionNodeStyle{4pt}
\xtdistance{15mm}{3.5cm}
\istroot(0){$P1$}
  \istb{A}[al]
  \istb{B}[ar]
  \endist
\xtShowEndPoints[oval node]
\xtdistance{15mm}{1.5cm}
\istroot(1)(0-1)<135>{$P2$}
  \istb{C}[l]{(1,2)}
  \istb{D}[l]{(1,-1)}
  \istb{E}[r]{(0,2)}
  \endist
\istroot(2)(0-2)<45>{$P2$}
  \istb{F}[l]{(2,2)}
  \istb{G}[r]{(1,3)}
  \endist
\end{istgame}
```



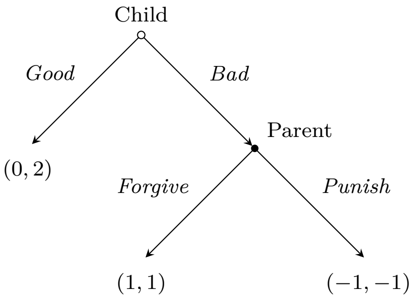
The istgame package enhances simplicity and readability, and hence it is easy to reuse codes. You can easily read and modify game tree codes even if you go over them after a while.

16 Game tree examples

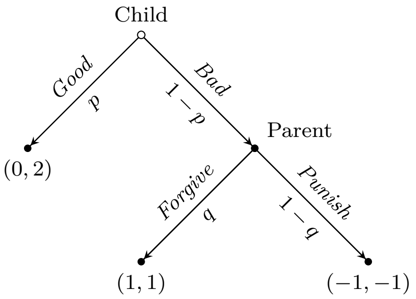
This section provides some examples of extensive games. Before we start, let us change the default font size of the `istgame` environment by stating `\setistgamefontsize{\footnotesize}` outside of the environment. (Right here!)

16.1 Simple examples

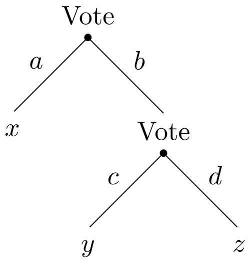
```
\begin{istgame}[->,shorten >=1.3pt]
\setistmathTF*001
\xtdistance{15mm}{30mm}
\istroot(0)[initial node]{Child}
  \istb{Good}[above left]{(0,2)}
  \istb{Bad}[above right]
  \endist
\istroot(1)(0-2)<30>{Parent}
  \istb{Forgive}[above left]{(1,1)}
  \istb{Punish}[above right]{(-1,-1)}
  \endist
\end{istgame}
```



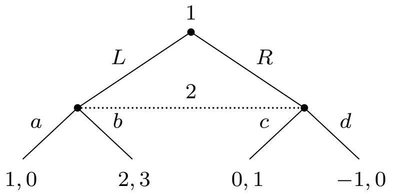
```
% dual sloped action labels (\istB)
\begin{istgame}
\setistmathTF*001 % action labels in italics
\xtShowArrows
\xtdistance{15mm}{30mm}
\istroot(0)[initial node]{Child}
  \istB{Good}[above,sloped]
       {$p$}[below, sloped]{(0,2)}
  \istB{Bad}[above,sloped]
       {$1-p$}[below, sloped]
  \endist
\istroot(1)(0-2)<30>{Parent}
  \istB{Forgive}[above,sloped]
       {$q$}[below, sloped]{(1,1)}
  \istB{Punish}[above,sloped]
       {$1-q$}[below, sloped]{(-1,-1)}
  \endist
\end{istgame}
```



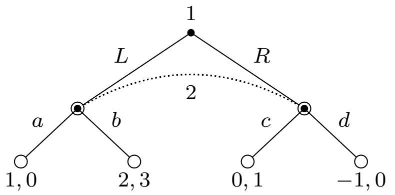
```
% IGT 218.1 (Osborne, 2004b)
\begin{istgame}[font=\normalsize]
\xtdistance{10mm}{20mm}
  \istroot(0){Vote}
  \istb{a}[al]{x}
  \istb{b}[ar]
  \endist
\istroot(1)([yshift=-1.5em]0-2){Vote}
  \istb{c}[al]{y}
  \istb{d}[ar]{z}
  \endist
\end{istgame}
```



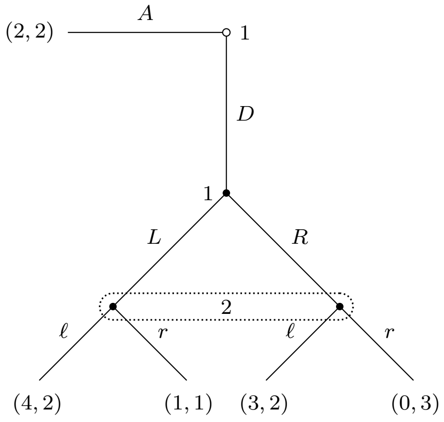
```
% information set
\begin{istgame}
\xtdistance{10mm}{30mm}
\istroot(0){1}
  \istb{L}[al]   \istb{R}[ar]
  \endist
\xtdistance{7mm}{15mm}
\istroot(1a)(0-1)
  \istb{a}[al]{1,0}
  \istb{b}[ar]{2,3}
  \endist
\istroot(1b)(0-2)
  \istb{c}[al]{0,1}   \istb{d}[ar]{-1,0}
  \endist
\xtInfoset(1a)(1b){2}
\end{istgame}
```



```
% information set
\begin{istgame}
\xtShowEndPoints[oval node]
\xtdistance{10mm}{30mm}
\istroot(0){1}
  \istb{L}[al]   \istb{R}[ar]
  \endist
\xtdistance{7mm}{15mm}
\istroot(1a)(0-1)
  \istb{a}[al]{1,0}   \istb{b}[ar]{2,3}
  \endist
\istroot(1b)(0-2)
  \istb{c}[al]{0,1}   \istb{d}[ar]{-1,0}
  \endist
\xtInfoset[bend left=30](1a)(1b){2}
\end{istgame}
```

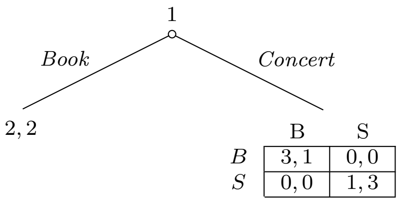


```
% (oval) information set
\begin{istgame}
\xtdistance{15mm}{30mm}
\istroot[-135](0)[initial node]<0>{1}
  \istb{A}[a]{(2,2)}[l]  \istb{D}[r]
  \endist
\istroot(1)(0-2)<left>{1}
  \istb{L}[al]    \istb{R}[ar]
  \endist
\xtInfosetO(1-1)(1-2){2}
\xtdistance{10mm}{20mm}
\istroot(2)(1-1)
  \istb{\ell}[al]{(4,2)}  \istb{r}[ar]{(1,1)}
  \endist
\istroot(3)(1-2)
  \istb{\ell}[al]{(3,2)}  \istb{r}[ar]{(0,3)}
  \endist
\end{istgame}
```



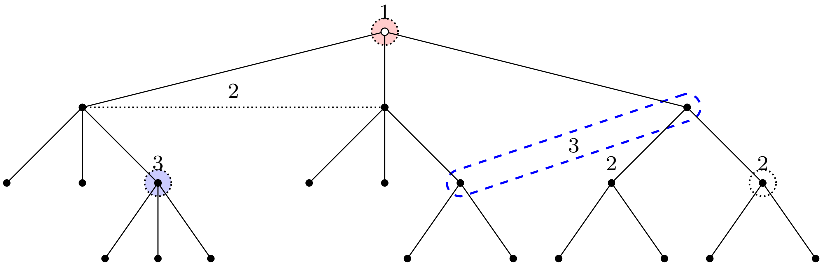
16.2 A game tree with a strategic game

```
%\usepackage{tabu}
%\NewExpandableDocumentCommand\xcol{mO{c|}m}
%  {\multicolumn{#1}{#2}{\ensuremath{#3}}}
\def\strategicgame{
\begin{tabular}  {c|c|c|} % to 3cm
\xcol1[c]{~} & \xcol1[c]{$B$} & \xcol1[c]{$S$}
    \\\cline{2-3}
$B$   & $3,1$     & $0,0$    \\\cline{2-3}
$S$   & $0,0$     & $1,3$    \\\cline{2-3}
\end{tabular}
}
  \begin{istgame}
\setistmathTF*001
\xtdistance{10mm}{40mm}
\istroot(0)[initial node]{1}
  \istb{Book}[al]{2,2}
  \istb{Concert}[ar]
  \endist
\xtPayoff(0-2){\strategicgame}[below,xshift=-7pt]
\end{istgame}
```

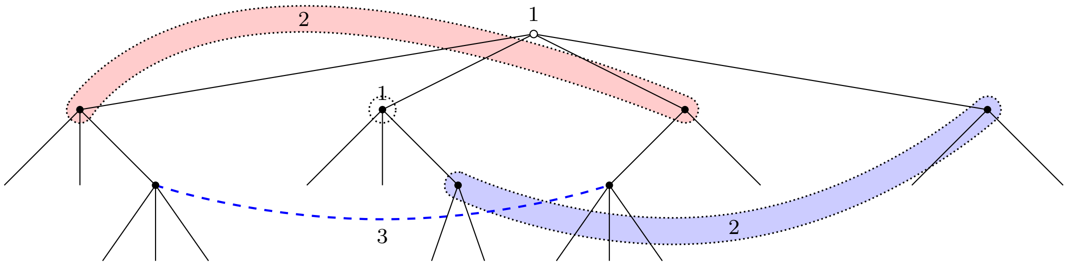


16.3 Larger game trees with information sets

```
\begin{istgame}
\xtShowEndPoints
\xtdistance{10mm}{40mm}
\istroot(0)[initial node]{1}  \istb  \istb  \istb  \endist
\xtdistance{10mm}{10mm}
\istroot(1)(0-1)              \istb  \istb  \istb  \endist
\istroot(2)(0-2)              \istb  \istb  \istb  \endist
\xtdistance{10mm}{20mm}
\istroot(3)(0-3)              \istb  \istb  \endist
\xtdistance{10mm}{7mm}
\istroot(a)(1-3){3}           \istb  \istb  \istb  \endist
\xtdistance{10mm}{14mm}
\istroot(b)(2-3)              \istb  \istb  \endist
\istroot(c)(3-1){2}           \istb  \istb  \endist
\istroot(d)(3-2){2}           \istb  \istb  \endist
\xtInfosetO[fill=red!20](0)(0)
\xtInfoset(1)(2){2}
\xtCInfosetO[fill=blue!20](a)(a)
\setxtinfosetstyle{blue,thick,dashed}
\xtInfosetO(b)(3){3}
\setxtinfosetstyle % restore defaults
\xtCInfosetO(d)(d)
\end{istgame}
```

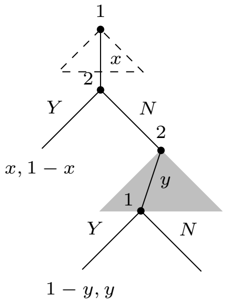


```
\begin{istgame}[scale=1]
\useasboundingbox (-7,-3) rectangle (7,.5);
\xtdistance{10mm}{40mm}
\istroot(0)[initial node]{1}  \istb  \istb  \istb  \istb \endist
\xtdistance{10mm}{10mm}
\istroot(1)(0-1)     \istb  \istb  \istb  \endist
\istroot(2)(0-2)  \istb  \istb  \istb  \endist
\xtdistance{10mm}{20mm}
\istroot(3)(0-3)  \istb  \istb  \endist
\istroot(4)(0-4)  \istb  \istb  \endist
\xtdistance{10mm}{7mm}
\istroot(a)(1-3)  \istb  \istb  \istb  \endist
\istroot(b)(2-3)  \istb  \istb  \endist
\istroot(c)(3-1)  \istb  \istb  \istb  \endist
\xtCInfosetO[fill=red!20](1)!.4!(3){2}
\xtInfosetO(2)(2){1}[above]
\setxtinfosetlayer{behind}
\xtCInfoset[blue,thick,dashed](a)(c)<.85>{3}[below]
\setxtinfosetlayer % restore default layer (behnd)
\xtCInfosetO[fill=blue!20](b)(4)<.7>{2}
\end{istgame}
```

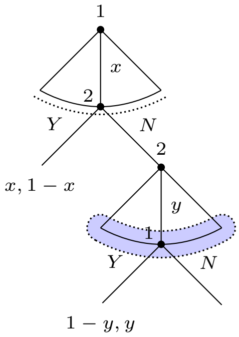


16.4 A continuum of branches

```
\begin{istgame}[font=\scriptsize]
\cntmdistance*{8mm}{16mm}
\cntmpreset*[dashed]{.7} % white triangle
\istrootcntm(0)(0,0){1}  \istb{x}[r]  \endist
\istroot(1)(0-1)<[label distance=-3pt]120>{2}
  \istb{Y}[al]{x,1-x}    \istb{N}[ar] \endist
\cntmpreset % restore defaults
\istrootcntm(2)(1-2){2}  \istb{y}[r]  \istbm  \endist
\istroot(3)(2-1)<[label distance=-3pt]120>{1}
  \istb[]{Y}[al]{1-y,y}  \istb{N}[ar] \endist
\end{istgame}
```

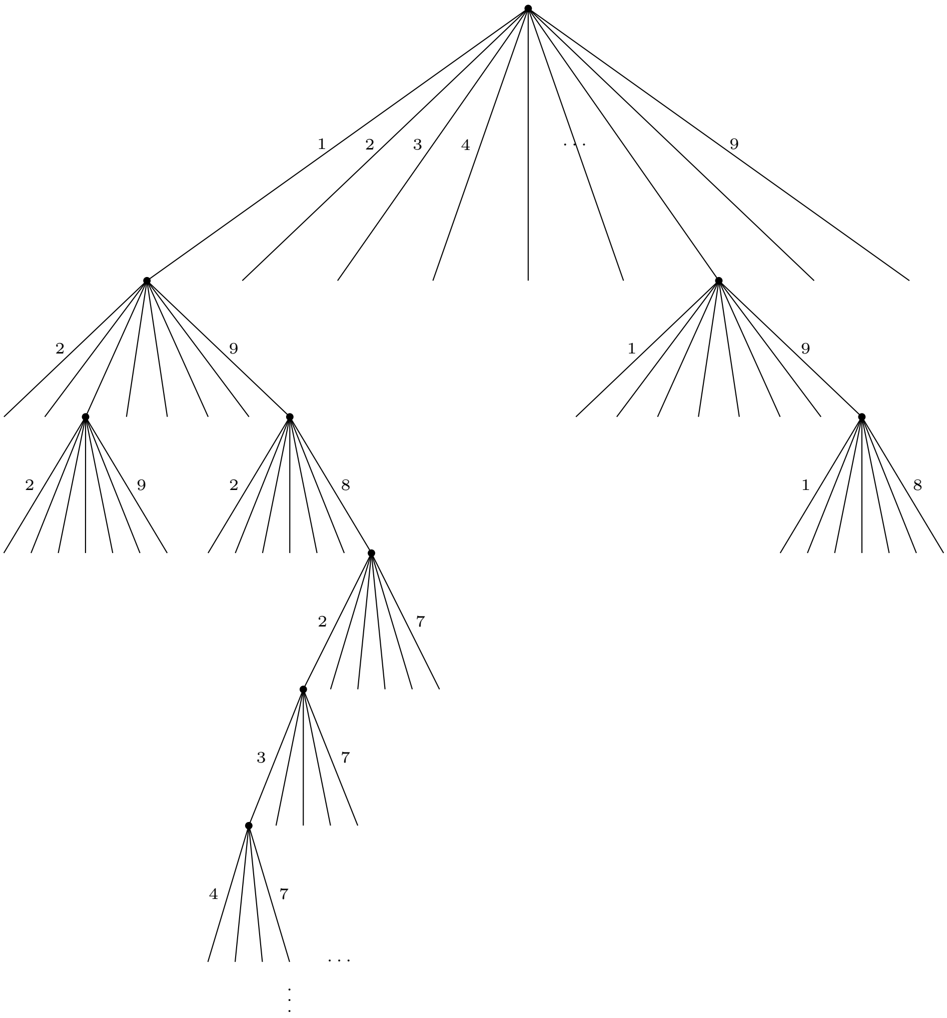


```
\begin{istgame}[font=\scriptsize]
\cntmdistance*{8mm}{16mm}
\istrootcntmA(0)(0,0){1} \istbA{x}[r]  \endist
\cntmAInfoset(0)
\istroot(1)(0-1)<[label distance=-3pt]120>{2}
  \istb{Y}[al]{x,1-x}    \istb{N}[ar]  \endist
\istrootcntmA(2)(1-2){2} \istbA{y}[r]  \endist
\cntmAInfosetO[fill=blue!20](2)
\istroot(3)(2-1)<[label distance=-3pt]120>{1}
  \istb[]{Y}[al]{1-y,y}  \istb{N}[ar]  \endist
\end{istgame}
```



16.5 Tic-tac-toe (sketch)

```
% tic-tac-toe (sketch)
\begin{istgame}[scale=1.8,font=\tiny]
\xtdistance{20mm}{7mm}
\istroot(0) \istb \istb \istb \istb \istb \istb{\dots} \istb \istb \istb{9}[r]
    \endist
\foreach \x in {1,...,4}
{\xtActionLabel(0)(0-\x){\x}[l]}
\xtdistance{10mm}{3mm}
\istroot(a1)(0-1) \istb{2}[l] \istb \istb \istb \istb \istb \istb \istb{9}[r]
    \endist
\istroot(a7)(0-7) \istb{1}[l] \istb \istb \istb \istb \istb \istb \istb{9}[r]
    \endist
\xtdistance{10mm}{2mm}
\istroot(A)(a1-3) \istb{2}[l] \istb \istb \istb \istb \istb \istb{9}[r] \endist
\istroot(B)(a1-8) \istb{2}[l] \istb \istb \istb \istb \istb \istb{8}[r] \endist
\istroot(C)(a7-8) \istb{1}[l] \istb \istb \istb \istb \istb \istb{8}[r] \endist
\istroot(Bx)(B-7) \istb{2}[l] \istb \istb \istb \istb \istb{7}[r] \endist
\istroot(By)(Bx-1) \istb{3}[l] \istb \istb \istb \istb{7}[r]  \endist
\istroot(Bz)(By-1) \istb{4}[l] \istb \istb \istb{7}[r]  \endist
\xtPayoff(Bz-4){\vdots}    \xtPayoff(Bz-4){\cdots}[right,xshift=10pt]
\end{istgame}
```



16.6 Selten's horse

```
% Selten's horse: IGT 331.2 (Osborne, 2004b)
\begin{istgame}
\xtdistance{8mm}{16mm}
\istroot[-45](0)[initial node]{1}
  \istb{D}[r]
  \istb<grow=east,level distance=30mm>{C}[a]
  \endist
\istroot(1)(0-1)+10mm..20mm+
  \istb{L}[al]{3,3,2}
  \istb{R}[ar]{0,0,0}
  \endist
\istroot[-45](a)(0-2){2}
  \istb{d}[r]
  \istb<grow=0,level
    distance=20mm>{c}[a]{1,1,1}[r]
  \endist
\istroot(a1)(a-1)+10mm..20mm+
  \istb{L}[al]{4,4,0}
  \istb{R}[ar]{0,0,1}
  \endist
\xtInfoset(1)(a1){3}
\end{istgame}
\captionof{figure}{IGT 331.2}
```

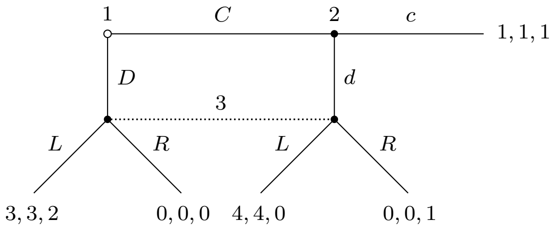
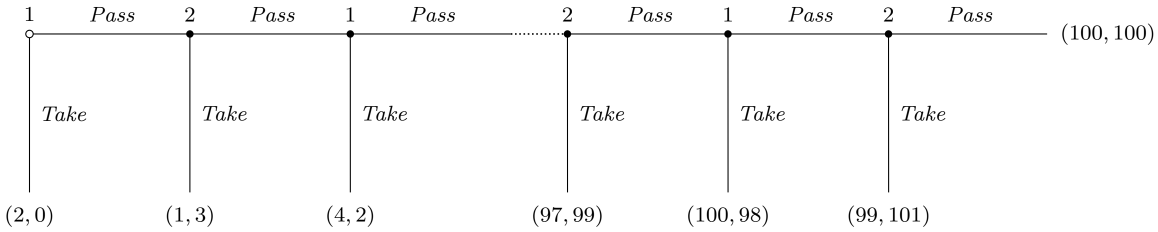


Figure 2: IGT 331.2

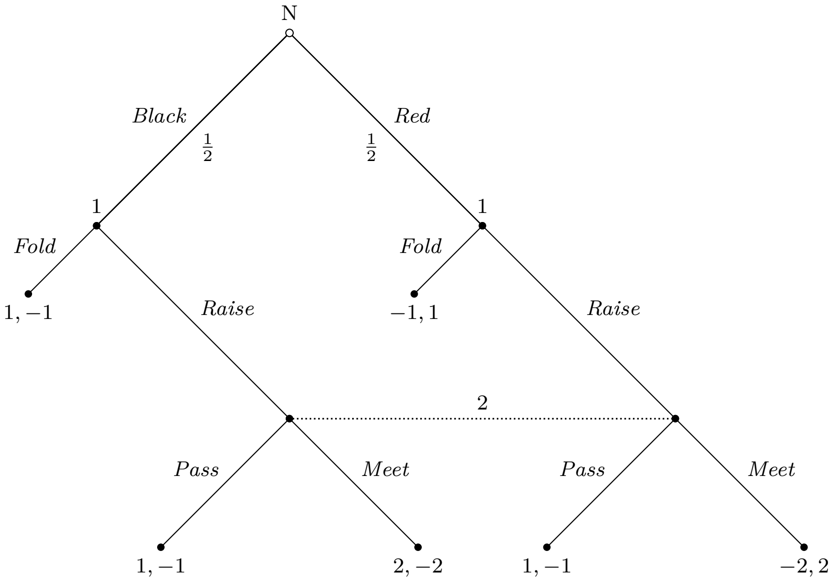
16.7 Centipede game

```
% centipede
\begin{istgame}[scale=1.5]
\setistmathTF*001
\setistgrowdirection{south east}
\xtdistance{10mm}{20mm}
\istroot(0)[initial node]{1}
  \istb{Take}[r]{(2,0)}[b]  \istb{Pass}[a]  \endist
\istroot(1)(0-2){2}
  \istb{Take}[r]{(1,3)}[b]  \istb{Pass}[a]  \endist
\istroot(2)(1-2){1}
  \istb{Take}[r]{(4,2)}[b]  \istb{Pass}[a]  \endist
\xtInfoset(2-2)([xshift=5mm]2-2)
%-------------
\istroot(3)([xshift=5mm]2-2){2}
  \istb{Take}[r]{(97,99)}[b]  \istb{Pass}[a]  \endist
\istroot(4)(3-2){1}
  \istb{Take}[r]{(100,98)}[b]  \istb{Pass}[a]  \endist
\istroot(5)(4-2){2}
  \istb{Take}[r]{(99,101)}[b]  \istb{Pass}[a]{(100,100)}[r]  \endist
\end{istgame}
```



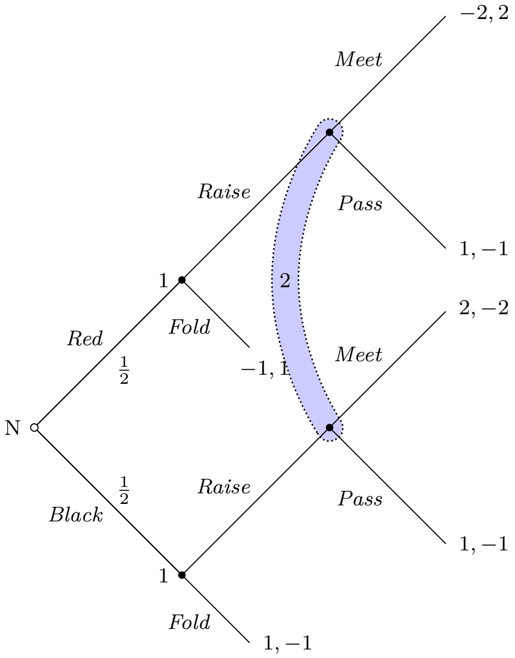
16.8 Poker game

```
% poker: growing south
\begin{istgame}[scale=1.7]
\setistmathTF*001
\xtShowEndPoints
\xtdistance{15mm}{30mm}
\istroot(0)[chance node]{N}
  \istB{Black}[al]{$\frac12$}[br]
  \istB{Red}[ar]{$\frac12$}[bl]
  \endist
\xtdistance{15mm}{30mm}
\istroot(1-1)(0-1){1}
  \istbA(.5)<grow=-135>{Fold}[al]{1,-1}
  \istb{Raise}[ar]
  \endist
\xtdistance{10mm}{20mm}
\istroot(1)(1-1-2)
  \istb{Pass}[al]{1,-1}
  \istb{Meet}[ar]{2,-2}
  \endist
\xtdistance{15mm}{30mm}
\istroot(1-2)(0-2){1}
  \istbA(.5)<grow=-135>{Fold}[al]{-1,1}
  \istb{Raise}[ar]
  \endist
\xtdistance{10mm}{20mm}
\istroot(2)(1-2-2){}
  \istb{Pass}[al]{1,-1}
  \istb{Meet}[ar]{-2,2}
  \endist
\xtInfoset(1)(2){2}
\end{istgame}
```

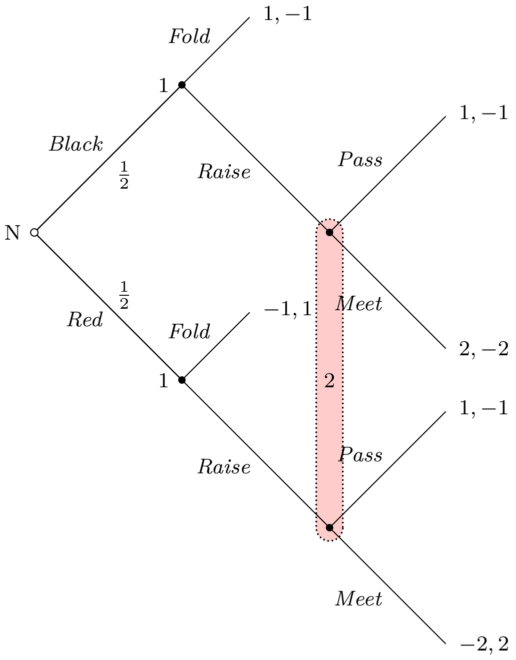


16.9 Poker game: growing to the right

```
% poker: growing east -- counterclockwise
\begin{istgame}[scale=1.3]
\setistmathTF*001
\setistgrowdirection{0}   % default grow-direction is 'east' from now on
\xtdistance{15mm}{30mm}
\istroot(0)[chance node]<left>{N}
  \istB{Black}[bl]{$\frac12$}[ar]
  \istB{Red}[al]{$ \frac12$}[br]
  \endist
\xtdistance{15mm}{30mm}
\istroot(1-1)(0-1)<left>{1}
  \istb<grow=-45,level distance=10mm>{Fold}[bl]{1,-1}
  \istb{Raise}[al]
  \endist
\xtdistance{12mm}{24mm}
\istroot(1)(1-1-2)
  \istb{Pass}[bl]{1,-1}
  \istb{Meet}[al]{2,-2}
  \endist
\xtdistance{15mm}{30mm}
\istroot(1-2)(0-2)<left>{1}
  \istb<grow=-45,level distance=10mm>{Fold}[bl]{-1,1}[[xshift=5pt]below]
  \istb{Raise}[al]
  \endist
\xtdistance{12mm}{24mm}
\istroot(2)(1-2-2)
  \istb{Pass}[bl]{1,-1}
  \istb{Meet}[al]{-2,2}
  \endist
\xtCInfoset0[fill=blue!20](1-1-2)(1-2-2){2}
\end{istgame}
```

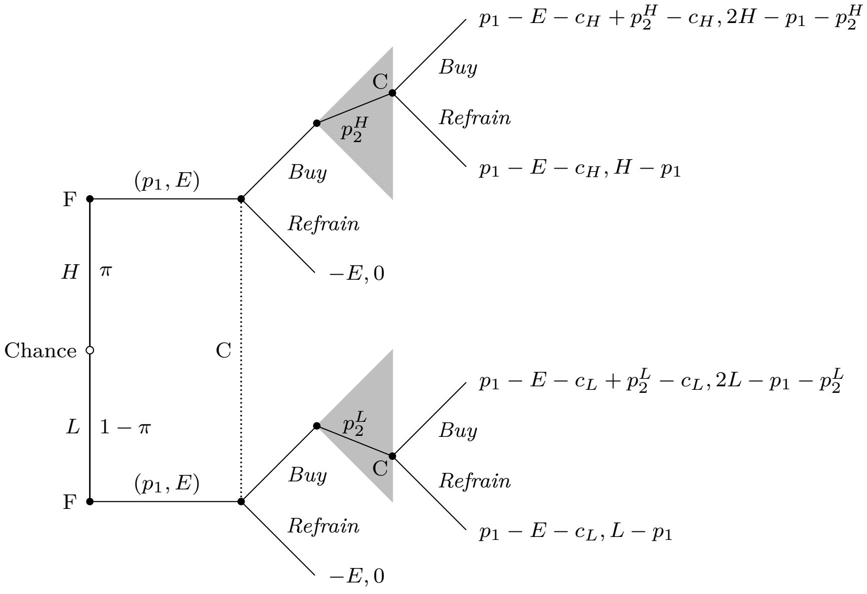


```
% poker: growing east -- clockwise (swap version)
\begin{istgame}[scale=1.3]
\setistmathTF*001
\setistgrowdirection'{0}   % \setistgrowdirection'
\xtdistance{15mm}{30mm}
\istroot(0)[chance node]<left>{N}
  \istB{Black}[al]{$\frac12$}[br]
  \istB{Red}[bl]{$\frac12$}[ar]
  \endist
\xtdistance{15mm}{30mm}
\istroot(1-1)(0-1)<left>{1}
  \istb<grow=45,level distance=10mm>{Fold}[al]{1,-1}
  \istb{Raise}[bl]
  \endist
\xtdistance{12mm}{24mm}
\istroot(1)(1-1-2)
  \istb{Pass}[al]{1,-1}
  \istb{Meet}[bl]{2,-2}
  \endist
\xtdistance{15mm}{30mm}
\istroot(1-2)(0-2)<left>{1}
  \istb<grow=45,level distance=10mm>{Fold}[al]{-1,1}
  \istb{Raise}[bl]
  \endist
\xtdistance{12mm}{24mm}
\istroot(2)(1-2-2)
  \istb{Pass}[al]{1,-1}
  \istb{Meet}[bl]{-2,2}
  \endist
\xtInfosetO[fill=red!20](1-1-2)(1-2-2){2}
\end{istgame}
```

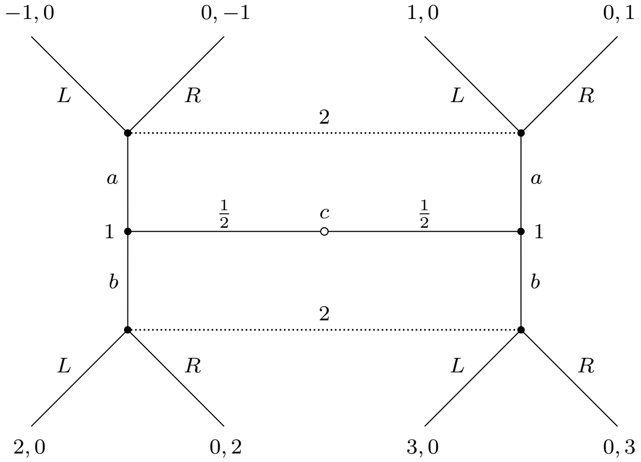


16.10 Signaling games

```
% signaling game: IGT 337.1 (Osborne, 2004b)
\begin{istgame}
\setistgrowdirection'{right}
% game start: choice of chance
\xtdistance{20mm}{20mm}
\istroot(0)[chance node]<180>{Chance}
  \istB<grow=north>{H}[l]{\pi}[r]  \istB<grow=south>{L}[l]{1-\pi}[r]  \endist
\istroot(H0)(0-1)<180>{F}
  \istb{(p_1,E)}[a]  \endist
\istroot(L0)(0-2)<180>{F}
  \istb{(p_1,E)}[a]  \endist
\setistmathTF*001
\cntmdistance*{10mm}{20mm}{8mm}
% subtree after H is chosen
\istroot(H)(H0-1)
  \istb{Buy}[br]  \istb{Refrain}[ar]{-E,0}  \endist
\istrootcntm(H1)(H-1)
  \istb{$p_2^H$}[b]  \istbm  \endist
\istroot(CH)(H1-1)<[label distance=-4pt]135>{C}
  \istb{Buy}[br]{p_1-E-c_H+p_2^H-c_H, 2H-p_1-p_2^H}
  \istb{Refrain}[ar]{p_1-E-c_H, H-p_1}
  \endist
% subtree after L is chosen
\istroot(L)(L0-1)
  \istb{Buy}[br]  \istb{Refrain}[ar]{-E,0}  \endist
\istrootcntm(L1)(L-1)
  \istbm  \istb{$p_2^L$}[a]  \endist
\istroot(CL)(L1-2)<[label distance=-4pt]-135>{C}
  \istb{Buy}[br]{p_1-E-c_L+p_2^L-c_L, 2L-p_1-p_2^L}
  \istb{Refrain}[ar]{p_1-E-c_L, L-p_1}
  \endist
\xtInfoset(H)(L){C}[left]
\end{istgame}
```



```
% signaling game
\begin{istgame}[scale=1.3]
\xtdistance{20mm}{20mm}
\istroot(0)[chance node]{$c$}
  \istb<grow=left>{\frac12}[a]
  \istb<grow=right>{\frac12}[a]
  \endist
\xtdistance{10mm}{20mm}
\istroot(1)(0-1)<180>{1}
  \istb<grow=north>{a}[l]
  \istb<grow=south>{b}[l]
  \endist
\istroot(2)(0-2)<0>{1}
  \istb<grow=north>{a}[r]
  \istb<grow=south>{b}[r]
  \endist
\istroot'[north](a1)(1-1)
  \istb{L}[bl]{-1,0}
  \istb{R}[br]{0,-1}
  \endist
\istroot(b1)(1-2)
  \istb{L}[al]{2,0}
  \istb{R}[ar]{0,2}
  \endist
\istroot(a2)(2-2)
  \istb{L}[al]{3,0}
  \istb{R}[ar]{0,3}
  \endist
\istroot'[north](b2)(2-1)
  \istb{L}[bl]{1,0}
  \istb{R}[br]{0,1}
  \endist
\xtInfoset(a1)(b2){2}
\xtInfoset(b1)(a2){2}
\end{istgame}
```



Version history

- v2.0(2019/01/27) to be upload to CTAN
  - changed the title of the package to “Draw Game Trees with Ti*k*Z”
  - package document done
- v2.0rc (2019/01/27)
  - introduced `\setxtinfosetstyle`
    - introduced `infoset style`, a new style for information sets
    - modified all definitions of information sets, accordingly
  - redefined macros related to layer, to easily restore default values
  - modified the cross arrow tip options
  - modified defaults of `\setxtarrowtips` not to depend on branch styles initially
  - code refinement
- v2.0beta1 (2019/01/26)
  - introduced `\xtshowXpoints` and `\xtshowXcircles` (for developer only, not documented)
- v1.0 (2017/09/04)

- introduced `\cntmlevdist` and `\cntmsibdist`
- introduced `\cntmdistance`
- redefined related marcos
- package document done

Acknowledgement

The basic idea of the istgame package came from Osborne's egameps package and Chen (2013). The update of the package istgame to `version 2.0` is partly motivated by the questions and discussions on the subject of `game tree` at `https://tex.stackexchange.com/`, where Alan Munn is thanked for his kind interest in the istgame package. Special thanks go to Kangsoo Kim of KTUG for his great help in using expl3 to simplify the appearance and the usage of the macros, and to resolve the issues on TEX's expansion.

References

Chen, H. K. (2013), "Drawing Game Trees with Ti*k*Z," `http://www.sfu.ca/~haiyunc/notes/Game_Trees_with_TikZ.pdf`.

Cho, I-S. (2017), "istgame: Drawing Game Trees with Ti*k*Z," in Korean TeX Society, *TEX, Beyond the World of Typesetting*, Seoul: Kyungmoonsa, 203–237.

Osborne, M. J. (2004a), "Manual for egameps.sty," Version 1.1, `http://texdoc.net/texmf-dist/doc/latex/egameps/egameps.pdf`

____________ (2004b), *An Introduction to Game Theory*, New York: Oxford.

Tantau, T. (2015), "Ti*k*Z and PGF: Manual for version 3.0.1a," `http://sourceforge.net/projects/pgf`.

Index