Drawing Games and Diagrams in LaTeX *

Sarah Bouchat (with additions by Michael Masterson)

September 19, 2018

This guide emphasizes a series of ways to draw game matrices and decision trees—both of which will be useful for 835 and 836—as well as providing some basic building blocks for drawing using LaTeX. This should be considered a very rudimentary introductory guide—there are many more things to learn about drawing and figures in LaTeX that are omitted here.

1 Strategic Form Games

Writing strategic form games using the `sgamevar` package creates diagrams that looks a lot like normal tabular environments, but it has the advantage of automatically tailoring the dimensions and labels to the alignment you want, rather than forcing you to do it by hand. There are a couple of things that distinguish it from a typical table environment:

- Use the "figure" float instead of the "table" float.
- The `\>` replaces `&` as the divider between cells. Note that in the `sgame` package, you use `&` still.
- The initial operators are different: rather than `\begin{tabular}{l|cc}...\end{tabular}` it is:
  `\begin{game}{rows}{columns}[player 1 label][player 2 label][figure label]...\end{game}`

		2		
		Left	Center	Right
1	Top	3,2	4,4	2,1
	Bottom	1,1	5,7	4, 3

Most stylistic commands also work inside games. Using circles to indicate best responses is problematic because the circles tend to cut off some of the numbers. You can also use highlighting.

*This handout borrows heavily from previous handouts authored by Dave Ohls (2010) and 2013 materials from Richard Loeza.

		2				
		Rock	Paper	Scissors	Flamethrower	Water Canon
	Rock	1,1	0,2	2,0	2,0	0,2
	Paper	2,0	1,1	0,2	0,2	2,0
1	Scissors	0,2	2,0	1,1	2,0	0,2
	Flamethrower	0,2	2,0	0,2	1,1	2,0
	Water Canon	2,0	0,2	2,0	0,2	①,①

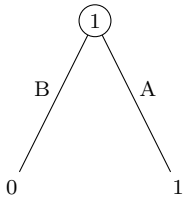
Ammunition

		Jim		
		A	B	C
Bob	X	8,6	6,8	0,0
	Y	6,8	8,6	0,0
	Z	0,0	0,0	5,5

2 Extensive Form Games

Previous versions of this workshop used the `egamesps` package to create extensive form games. That package makes creating extensive games simpler, but it is very prone to errors and much less flexible than using the tikz approach. Be aware that using egamesps may require you to use a different compilation engine for your LaTeX document (e.g., XeTeX).

Instead of dealing with these issues, you can create extensive form games using the `tikzpicture` environment, from the tikz package. The tikz package is designed for drawing in general, and is very flexible. The package can accommodate nearly any shape or arrangement you desire. The basic syntax of the coding language involves drawing a series of objects (lines, geometric shapes, text), specifying their location on a coordinate scale, and specifying the properties you want them to have (size, style, color, shading, special features).



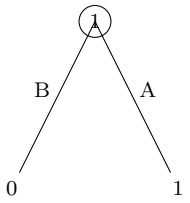
What the commands are doing:

- `[scale=1]`Scales the figure with the dimensions given (in centimeters). Increase (or decrease) the scale to expand (or shrink) the figure.
- `\draw (0,2) -- (1,0);`: Draws a straight line between coordinate points (0,2) and (1,0).
- `\draw (.7,1.1) node{\scriptsize{A}};` Creates a node with the small text A at

the coordinates (.7,1.1).

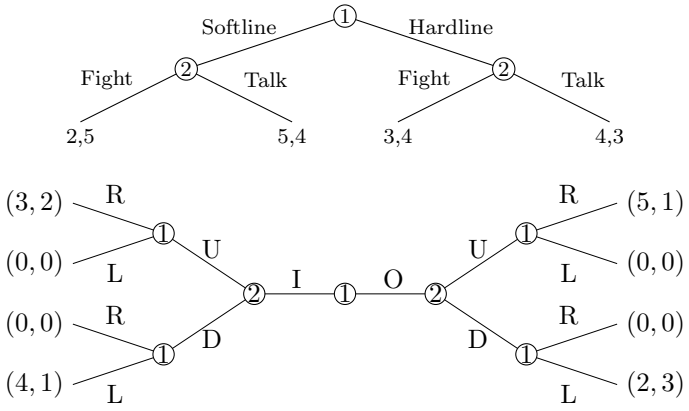
- `\filldraw[fill=white](0,2) circle (6pt) node{\scriptsize{1}};` Creates a circle, 6 points in diameter and fills the inside white. It then creates a small 1 in the middle of the circle. Note that this will cover up anything under the filled circle.

With tikz, the order in which objects are rendered is important. A filled circle will go over anything at the same coordinates if it is listed after those things. Note how the above figure looks when the ordering is changed.



Note that every line in a tikzpicture ends in a semicolon. A common error is forgetting to put one in.

With these basic tools, you can make games in a variety of shapes.



3 Make Custom Diagrams with Tikz

You can make your own custom diagrams with Tikz. Often these are used to make diagrams of causal processes, but you could potentially use it for anything.

3.1 Defining Custom Shapes

First you need to define the shapes you wil use in the preamble. In this document's preamble I define the shapes as follows:

```
\tikzstyle{startstop} = [rectangle, rounded corners, minimum width=1in, minimum height=0.5in,text centered, draw=black, fill=red!30]
\tikzstyle{dom} = [rectangle, rounded corners, minimum width=1in, minimum height=0.5in,text centered, draw=black, fill=red!30]
\tikzstyle{io} = [diamond, minimum width=0.25in, minimum height=2.3in, text centered, draw=black, fill=blue!30]
```

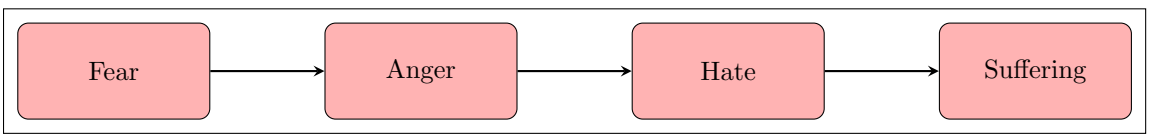
```
\tikzstyle{arrow} = [thick,->,>=stealth]
\tikzstyle{negarrow} = [thick,->,>=stealth, draw=red, fill=red]
\tikzstyle{assocarrow} = [thick,->,>=stealth, draw=green, fill=green]
```

The bracers after '\tikzstyle' define the name of the shape. In the square brackets, the shape of the shape, its size, color, and orientation can be defined.

3.2 Making Diagrams

Figure 1 shows the causal process of the dark side according to Yoda using only the startstop shape and the arrow shape.

Figure 1: The Darkside Causal Process Diagram



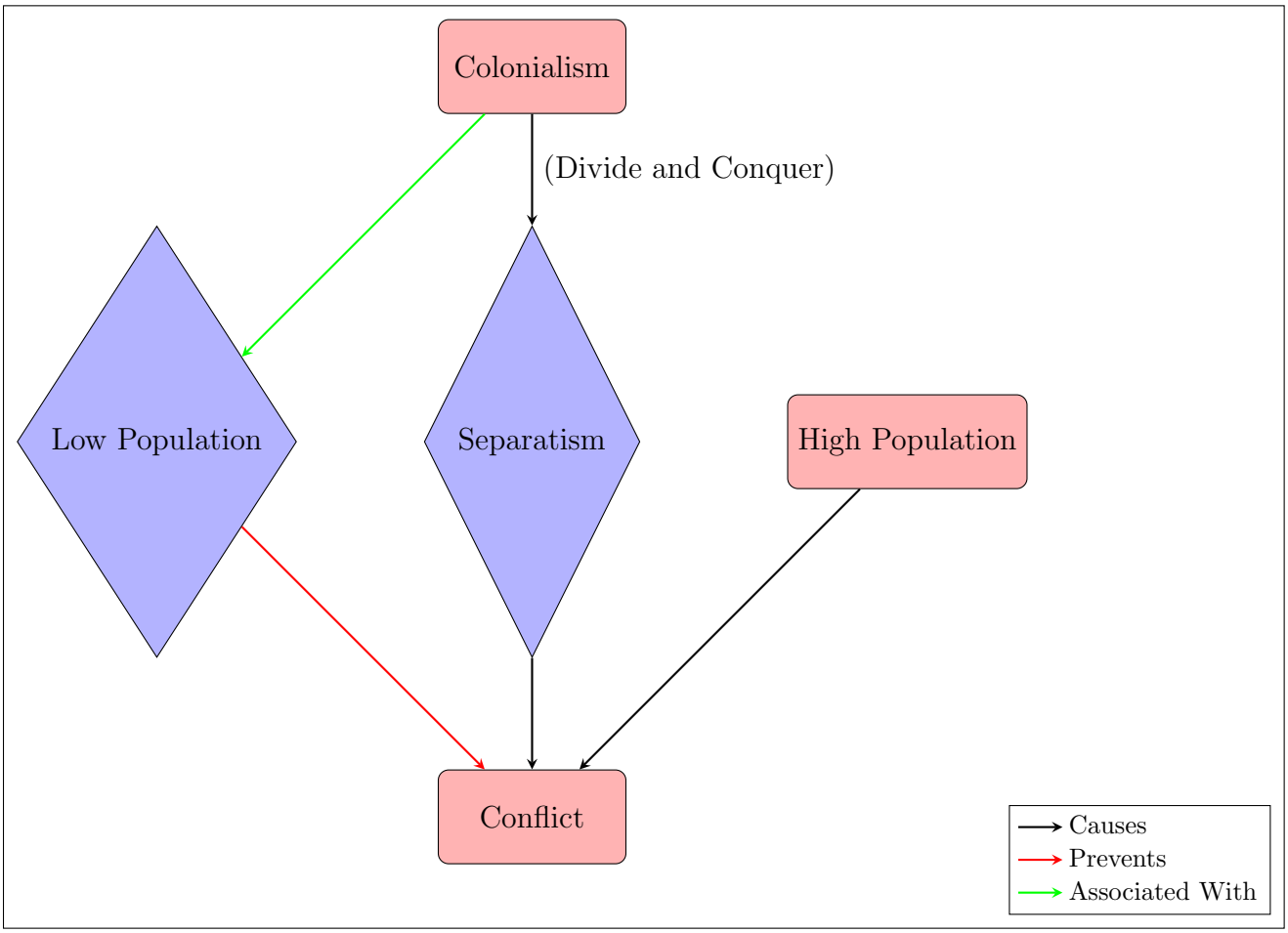
Here is the code used to make Figure 1:

```
\begin{centering}
\begin{figure}
\caption{\Large The Darkside Causal Process Diagram}
\label{dside}
        \begin{tikzpicture}[framed, node distance=1.6in]
        \begin{footnotesize}
         \node (start) [startstop, anchor = west] {Fear};
\node (anger) [startstop, right of=start] {Anger};
\node (hate) [startstop, right of = anger] {Hate};
\node (suffering) [startstop, right of=hate] {Suffering};
\draw [arrow] (start) --  (anger);
\draw [arrow] (anger) -- (hate);
\draw [arrow] (hate) -- (suffering);
%
        \end{footnotesize}
        \end{tikzpicture}
\end{figure}
\end{centering}
```

In the call to tikzpicture, I define the distance between nodes as well as whether the picture is framed. To create a node, use \node and follow it with its name. In the square brackets, specify the name of the node you defined in the preamble that you wish to use as well as the relative location of the node. You can add text to the node inside bracers at the end of the line.

You can also add text above arrows, use multiple shapes in a single diagram, and create a legend. See figure 2 below:

Figure 2: Theorized Relationships



To add text next to an arrow just add 'node[relative location]' and text you want to add after the

'--'

and before the second shape the arrow connects with. For example:

```
\draw [arrow] (start) --node[right] {(Divide  and Conquer)} (in2);
```

To add a legend, add a list of legend entries that you want to be the labels of the legend, specify the size and font of the legend, and lastly add the tikz shaped you want the legend to label. The legend should be within the tikz picture environment. The code to make the legend in Figure 2 is the following:

```
\begin{customlegend}[legend cell align=left,
legend entries={
Causes,
Prevents,
Associated With,
},
legend style={at={(10,-10)},font=\footnotesize}] % <= to define position and font legend
% the following are the "images" and numbers in the legend
\addlegendimage{arrow}
\addlegendimage{negarrow}
\addlegendimage{assocarrow}
\end{customlegend}
```