

Game-Theoryst

Connor T. Wiegand

Contents

Overview	2
Importing	2
Color	3
Cell Customization	3
Padding	3
Automatic Cell Sizing	4
Semantic Game Styling	4
Underlining	4
Mixed Strategies	5
Iterated Deletion (Elimination) of Dominated Strategies	5
Debugging	7

Overview

Full Example

```
#nfg(
  players: ([A\ Joe], [Bas Pro]),
  s1: ([$x$], [a]),
  s2: ("x", "aaaa", [$a$]),
  pad: ("x": 12pt, "y": 10pt),
  eliminations: ("s11", "s21", "s22"),
  ejust: (
    s11: (x: (0pt, 36pt), y: (-3pt, -3.5pt)),
    s22: (x: (-10pt, -12pt), y: (-10pt, 10pt)),
    s21: (x: (-3pt, -9pt), y: (-10pt, 10pt)),
  ),
  mixings: (hmix: ($p$, $1-p$), vmix: ($q$, $r$, $1-q-r$)),
  custom-fills: (hp: maroon, vp: navy, hm: purple, vm: fuchsia, he: gray, ve:
gray),
  [$0,vul(1000000000)$], [$0,1$], [$0,0$],
  [$hul(1),1$], [$0, -1$], table.cell(fill: yellow.lighten(30%),
[$hful(0),vful(0)$])
)
```

		Bas Pro		
		$[q]$ x	$[r]$ aaaa	$[1 - q - r]$ a
A Joe	$[p]$ x	0, <u>1000000000</u>	0, 1	0, 0
	$[1 - p]$ a	<u>1</u> , 1	0, -1	<u>0</u> , <u>0</u>

Importing

Simply insert the following into your Typst code: (Coming soon)

```
#import "@preview/game-theoryst:0.1.0": *
```

This imports the `nfg()` function as well as the underlining methods. If you want to tweak the helper functions for generating an `nfg`, import them explicitly through the `utils/` directory. The package's repository is located at <https://github.com/connortwiegand/game-theoryst>.

Example

The main function to make strategic (or **normal**) form games is `nfg`. For a basic 2x2 game, you can do

```
#nfg(
  players: ("Jack", "Diane"),
  s1: ($C$, $D$),
  s2: ($C$, $D$),
  [$10,10$], [$2,20$],
  [$20,2$], [$5,5$],
)
```

		Diane	
		C	D
Jack	C	10, 10	2, 20
	D	20, 2	5, 5

All of `<players>`, `<s1>`, and `<s2>` have defaults for convenience sake. Payoffs (table entries) are provided via unnamed arguments after all other payoff matrix options have been set.

Color

By default, player names, mixed-strategy parameters (called *mixings*), and elimination lines are shown in color. These colors can be turned off at the method-level by passing `bw: true`, or at the document level by running the state helper-function `#colorless()`.

`nfg` accepts custom colors for all of the aforementioned parameters by passing a dictionary of colors to the `custom-fills` arg. The keys for this dictionary are as follows (`<defaults>`):

- `hp` – “horizontal player” (**red**)
- `vp` – “vertical player” (**blue**)
- `hm` – “hor. mixing” (**#e64173**)
- `vm` – “ver. mixing” (**eastern**)
- `he` – “hor. elimination” line (**orange**)
- `ve` – “ver. elimination” line (**olive**)

Cell Customization

Since the payoffs are implemented as argument sinks (`.args`) which are passed directly to Typst’s `#table()`, underlining of non-math can be accomplished via the standard `#underline()` command. Similarly, any of the payoff cells can be customized by using `table.cell()` directly. For instance, `table.cell(fill: yellow.lighten(30%), [$1,1$])` can be used to highlight a specific cell.

Padding

There are edge cases where the default padding may be off. These can be mended by passing the optional `pad` argument to `nfg()`. This should represent how much **additional** padding you want. The `pad` arg. is interpreted as follows:

- If a `length` is provided, it assumes you want that much length added to all cell walls
- If an array of the form `(L1, L2)` is provided, it assumes you want padding a horizontal (`x`) padding of `L1` and a vertical padding (`y`) of `L2`

- If a `dictionary` is provided, it operates identically to that of the array, but you must specify the `x / y` keys yourself

Automatic Cell Sizing

Cells are automatically sized to equal heights/widths according to the longest/tallest content. If you want to avoid this behavior, pass `lazy-cells: true` to `nfg`. This behavior can be combined with the custom `padding` argument.

Semantic Game Styling

Underlining

The package imports a small set of underlining utility functions.

The primary functions for underlining are

- `hul()` – *Horizontal Underline*
- `vul()` – *Vertical Underline*
- `bul()` – *Black Underline*

These can be wrapped around values in math-mode (`$. . $`) within the payoff matrix. The underlines for `hul` and `vul` are colored by default according to the default colors for names, but they accept an optional `col` parameter for changing the color of the underline. `bul()` produces a black underline.

```
#nfg(
  players: ("Jack", "Diane"),
  s2: ($x$, $y$, $z$),
  s1: ($a$, $b$),
  [$hul(0), $vul(0)$], [$1,1$], [$2,2$],
  [$3,3$], [$4,4$], [$5,5$],
)
```

		Diane		
		<i>x</i>	<i>y</i>	<i>z</i>
Jack	<i>a</i>	<u>0</u> , <u>0</u>	1, 1	2, 2
	<i>b</i>	3, 3	4, 4	5, 5

By default, these commands leave the numbers themselves black, but boldface them. *Full Color* versions of `hul` and `vul`, which color the numbers and under-lines identically, are available via `hful()` and `vful()`. Like their counterparts, they accept an optional `col` command for the color.

Both of the colors can be modified individually via the general `cul()` command, which takes in content (`cont`), an underline color (`ucl`), and the color for the text value (`tccl`). For instance,

```
#let new-ul(cont, col: olive, tccl: fuchsia) = { cul(cont, col, tccl) }
```

will define a new command which underlines in olive and sets the text (math) color to fuchsia.

Mixed Strategies

You can optionally mark mixed strategies that a player will in a `nfg` using the `mixing` argument. This can be a dictionary with `hmix` and `vmix` keys, or an array, interpreted as a dictionary with the aforementioned keys in the `(hmix, vmix)` order. The values/entries here should be arrays which mimic `s1` and `s2` in size, with some parameter denoting the proportion of time the relevant player uses that strategy. If you would like to omit a strategy from this markup, pass `[]` in it's place.

For example, in a 2x3 game, the following dictionary would add mixing parameters to both of player 1's strategies and player 2's first and third strategies:

```
(hmix: (p$, 1-p$), vmix: (q$, [], 1-q$))
```

Example

```
#nfg(
  players: ("Chet", "North"),
  s1: (F$, G$, H$),
  s2: (X$, Y$),
  mixings: (
    hmix: (p$, 1-p$),
    vmix: (q$, [], 1-q$)),
  [7,3$, 2,4$,
  [5,2$, 6,1$,
  [6,1$, 5,4$
  )
```

		North	
		$[q]$ X	$[1-q]$ Y
Chet	$[p]$ F	7, 3	2, 4
	G	5, 2	6, 1
	$[1-p]$ H	6, 1	5, 4

Iterated Deletion (Elimination) of Dominated Strategies

You can use the `pinit` package to cross out lines, semantically eliminating strategies. `pinit` comes pre-imported with `gametheoryst` by default.

```
#let directions = (N$, S$, E$, W$)
#let elements = (W$, E$, F$, A$)
#let domd = ("s12", "s13", "s14", "s21", "s22", "s23")
```

```
#nfg(
  players: ("A", "B"),
  s1: directions,
  s2: elements,
  eliminations: domd,
  ejust:(
    "s12": (x: (0pt, 10pt), y: (-3pt, -3pt)),
    "s13": (x: (0pt, 10pt), y: (-3pt, -3pt)),
    "s14": (x: (0pt, 10pt), y: (-3pt, -3pt)),
    "s21": (x: (-6pt, -8pt), y: (3pt, 8pt)),
    "s22": (x: (-4pt, -8pt), y: (3pt, 8pt)),
    "s23": (x: (-4pt, -8pt), y: (3pt, 8pt)),
  ),
  [$6,4$], [$7,3$], [$5,5$], [$6,6$],
  [$7,3$], [$2,7$], [$4,6$], [$5,5$],
  [$8,2$], [$6,4$], [$3,7$], [$2,8$],
  [$3,7$], [$5,5$], [$4,6$], [$5,5$],
)
```

B

	W	E	F	A
N	6,4	7,3	5,5	6,6
S	7,3	2,7	4,6	5,5
E	8,2	6,4	3,7	2,8
W	3,7	5,5	4,6	5,5

A

You can tell `nfg` which strategies to eliminate with the `eliminations` argument and the corresponding `ejust` helper-argument. The `eliminations` argument is simply an array of strings of the form "`s<i><j>`", where `<i>` is the player – 1 or 2 – and `<j>` is player `i`'s `<j>`th strategy (in left-to-right / top-to-bottom order *starting from 1*). These strategy strings represent the rows/columns which you want to eliminate. For instance, ("`s12`", "`s21`") denotes an elimination of player 1's second strategy as well as player 2's first strategy.

Due to context dependence, the lines typically need manual adjustments, which can be done via the `ejust` arg. `ejust` needs to be a dictionary with keys of matching those strings present in `eliminations` (`s11`, `s21`, etc.). The values of one of these dictionary entries is itself a dictionary: one with `x` and `y` keys. Each of these keys needs an array consisting of 2 lengths, corresponding to the starting/ending `dx/dy` adjustments from `pinit-line`.

Example

Here is the previous game with no `ejust` options included. As you can see, even similar lines may need different adjustments.

B

	W	E	F	A
N	6,4	7,3	5,5	6,6
S	7,3	2,7	4,6	5,5
E	8,2	6,4	3,7	2,8
W	3,7	5,5	4,6	5,5

A

For example, one such `ejust` argument could be `("s12": (x: (5pt, -5pt), y: (-10pt, 3pt)))`. This says to adjust the “s12” elimination line by `5pt` in the x direction and `-10pt` in the y direction for the starting (strategy-) side of the line, and adjust by `-5pt` in x and `3pt` in y on the ending (payoff-) side of the line.

Pin modifications

If you would like to modify the pins and/or lines in any way, there are easy-to-follow naming conventions for the pins. The name of every pin is

`<prefix-id>-s<i><j>-<start/end suffix>`.

Prefix ID

To prevent confusion of `pinit-lines` across games within a document, there is a counter called `_nfg-counter` which steps every time `nfg` is called. The counter begins from 0. By default, this counter is used to define the start of every pin; if the value of `_nfg-counter` is `<c>`, then the start of the pin label is “nfg-”.

You may change the starting prefix of the elims within a game by passing a `string` to the `gid` argument of `nfg`. *No additional - will be added.* Note that the `_nfg-counter` will still increment for every game in the document.

Strategy

The strategy portion of the pin name is identical to the strategy being crossed out; see above for explanation.

Start/End Suffix

For the suffix, the rule is as follows:

- The starting pin – e.g., strategy-side pin – is labelled with the suffix `--start`
- The ending pin – e.g., payoff-side pin – is labelled with the suffix `--end`

For example, consider an elimination line in the 1st game within a document which eliminates player 2’s 1st strategy. This `pinit-line` would connect the pins `nfg0-s21--start` with `nfg0-s22--end`.

Debugging

If you want to see all of the lines for the table, including the ones for a players, strategies, and mixings, set the following at the top of your document.

```
#set table.cell(stroke: (thickness: auto))
```

Note that cells are always present for mixings, they just have 0 width/height when no mixings of a specific variety are provided.