The ‘`lawofforms`’ package

Thomas Schmidt

January 12, 2020

1 Introduction

This package is an implementation to enable the clean and easy drawing of Laws of Form[2] formulas. This document describes the usage of the implemented features, as well as how to extend the package to support new and customised features. This package uses the `arrayjobx` package[1] to store the encoded formulas in an array we called `\orderarray`, through which we then iterate and draw the formulas using the Tikz package[3].

2 Command reference

`\lof` : Create a new cross at the current location.

Syntax: `\lof{`*`content`*`}{`*`id`*`}`

Example: `\lof{}{1}`

Remark: By “content,” we mean further, nested forms, since for those nesting purposes it is required to differentiate between nested forms and written text.

`\lofc` : Creates a new cross with text at the current location.

Syntax: `\lof{`*`content`*`}{`*`text`*`}{`*`id`*`}`

Example: `\lof{}{f}{1}`

`\re` : Creates a reentry, originating from the given `id`, at the current location.

Syntax: `\re{`*`id`*`}`

Example: `\re{1}`

`\ren` : Creates a reentry with text, originating from the given `id`, at the current location.

Syntax: `\ren{`*`text`*`}{`*`id`*`}`

Example: `\ren{Problem}{1}`

Remark: This currently does not support spacing for additional reentries going below this one.

`\lawofforms` : Creates a single line equation math environment, in which the above commands can be used.

Syntax: `\begin{lawofforms}` *`content`* `\end{lawofforms}`

Example:

`\begin{lawofforms} f = \lofc{\re{1}}{a}{1} \end{lawofforms}`

`\lofinline` : Creates an inline math environment in which the above commands can be used.

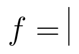
Syntax: `\begin{lofinline}` *`content`* `\end{lofinline}`

Example:

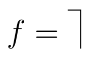
`\begin{lofinline} f = \lofc{\re{1}}{a}{1}\end{lofinline}`

Remark: This also creates a box with appropriate height around the formula, so that it fits with surrounding text.

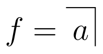
3 Examples



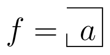
```
\begin{lawofforms}
  f = \lof{}{1}
\end{lawofforms}
```



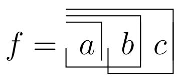
```
  \begin{lawofforms}
  f = \lofc{}{}{1}
  \end{lawofforms}

```



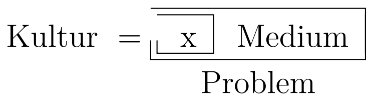
```
\begin{lawofforms}
f = \lofc{}{a}{1}
\end{lawofforms}
```



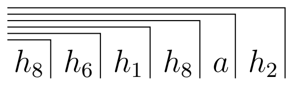
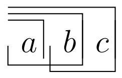
```
\begin{lawofforms}
f = \lofc{\re{1}}{a}{1}
\end{lawofforms}
```



```
\begin{lawofforms}
f = \lofc{\lofc{\lofc{\re{2}}{a}{1}\re{3}}{b}{2}}{c}{3}
\end{lawofforms}
```



```
\begin{lawofforms}
\text{ Kultur } = \lofc{\ren{Problem}{2}\lofc{\re{1}}{\text{ x }}{1}}{\text{ Medium }}{2}
\end{lawofforms}
```

This is the start of an example text $h_6 =$  this is the middle of an example text $f =$  this is the end of an example text.

```
This is the start of an example text \begin{lofinline}
h_6 = \lofc{\lofc{\lofc{\lofc{\lofc{\lofc{}{h_8}{1}}{h_6}{2}}{h_1}{3}}{h_8}{4}}{a}{5}}{h_2}{6}
\end{lofinline} this is the middle of an example text \begin{lofinline}
f = \lofc{\lofc{\lofc{\re{2}}{a}{1}\re{3}}{b}{2}}{c}{3}
\end{lofinline} this is the end of an example text.
```

4 Extending features

To extend this package with additional features, there are several key places in the code where certain features can be added or changed.

The first place to start are the definitions of the actual commands that are used to encode the desired forms, i.e. `\lof`, `\lofc`, `\re`, and `\ren`, and their respective helper functions (see below) that enter corresponding elements into the `\orderarray`. To create a fundamental new feature, one will need to first create the command which will be used in the `\lawofform` environments. This command will then have to set the appropriate `\orderarray` entries, with the following commands:

`\enterOAL` : Creates an entry for the start of a cross in the `\orderarray`.

Syntax: `\enterOAL{`*id*`}`

`\enterOAR` : Creates an entry for the end of a cross in the `\orderarray`.

Syntax: `\enterOAR{`*id*`}`

`\enterOAT`	: Creates an entry for the text under a cross in the `\orderarray`. Syntax: `\enterOAT{id}`
`\enterOAE`	: Creates an entry for a reentry in the `\orderarray`. Syntax: `\enterOAE{id}`
`\enterOAN`	: Creates an entry for a reentry with text in the `\orderarray`. Syntax: `\enterOAN{id}`

It is likely that a new feature will require a new type of entry, and therefore an additional `\enterOA` command, in the `\orderarray`. To then implement the new feature, the `\formdraw` command will need to be extended.

Firstly, the `\formdraw` command iterates through the `\orderarray` and creates Tikz nodes according to the type of entry in the `\orderarray`. This loop will need to be extended to account for the new type of `\orderarray` entry and implement the desired behaviour.

If the desired behaviour requires a new type of Tikz node (the types are corresponding to the orderarray entries except for the text), the loop in which we draw the crosses will then have to be extended to account for the new type of node. Note that the loop itself does not iterate over the nodes themselves, but over the indices, i.e. `id`'s, that were set by hand. In the loop, we draw the cross from the left node of the current index (i.e. `l<id>`), to the right node of the current index (i.e. `r<id>`), and then down to the bottom of the line. We then check if there are reentry or reentry-with-text nodes for the current index (i.e. `e<id>` or `n<id>`), and then draw the respective path. Additional checks for new types of nodes can be simply implemented with the `\ifnodedefined` helper command. Nodes that have been drawn need to be deleted with the `\aeundefinenode` command at the end of the iteration, so as to free up the name-space for the next use.

If you do not wish to implement a new drawing feature, but simply need to create a new environment in which to use our package, for example a multi-line math environment, you can simply do that by using the `\newenvironment` command, and specifying your desired formatting and environment. Note that it is assumed that our commands are used in a math environment, and that you will have to call the `\formdraw` command at the end of your environment, or whenever you want your form to be drawn. For example in the case of a multi-line math environment, it is likely that you want to call the `\formdraw` command at the end of every line.

References

[1] Jiang, Z. Arrayjobx. https://ctan.org/pkg/arrayjobx.

[2] Spencer-Brown, G. *Laws of Form.* 1969.

[3] Tantau, T. Tikz. https://www.ctan.org/pkg/pgf.