fitchcraft

Visualize of Fitch-style proof diagrams, using custom syntax optimized for speedy writing and conciseness.

Introduction

I started writing this library after finding out plotting Fitch-style proof diagrams with existing tools was not possible, or at best unsatisfactory. As such plots were required for a course in introductory logic I was taking, I was facing a dilemma: learn LaTeX and use its existing libraries, or write one of my own. As I love Typst, I have chosen the latter course. This has allowed me to study some more on how visualization works in Typst, but more importantly, make my own proof-writing syntax for, well, *speedy writing and conciseness*.

Note

For the library to function properly, **all** elements of the entry file must be imported ending with the `: *` syntax, or you could import it as a module and access it when needed.

Proof Syntax

In *fitchcraft*, a proof is an array of "lines" of two kinds: formulas, and utility lines.

Formulas

Formulas are essentially arrays of length 3 of the following form: `(index, equation, rule)`. The index is the ordinal (or any other character) associated with a certain line in the proof; the equation is the "line itself"; and the rule is the rule used to derive that equation; all are of type `content`, and expectedly math equations.

Formula "Polymorphism"

More often than not, you'd want the indices to be determined automatically as `1`,`2`,`3` etc. Conventiently, Typst does offer such functionality, which I utilize here for simplification of the syntax. If an indexation scheme is defined, there is no need to explicitly enter the index at every line, although possible. In such cases, arrays of the form `(equation, rule)` will be evaluated as needed. Similarly, when writing a rule-less formula, it can be entered as `(equation,)`, `(equation)`, or just `equation`.

Utility Lines

Utility lines are constants inserted into the proof array in order to specify properties of the proof-structure, and not of the lines themselves. There are three utility lines: `open` which opens a subproof, `close` which closes a subproof, and `assume` which places a line under the last formula's equation, noting the end of a list of assumptions.

Example

```
(
  ($x -> y$),
  $x$
  assume,
  ($y$, $-> E 1,2$)
)
```

The `framing` Object

The `framing` object is the building-block of the "frame" of the proof. It consists of two main parts: a vertical line, and a horizontal line called the "assumption line", appearing when inserting an `assume` utility formula.

Assumption Modes

Assumption modes are ways of determining the lengths of assumption lines. At the moment, exist three: `fixed`, `widest`, and `dynamic-single`, with the fourth `dynamic` in the works. When using the `fixed` mode, all assumption lines are of a fixed length, determined in a way which will be demonstrated later. When using the `widest` mode, all assumption lines are of a fixed length, determined by the width of the (visually) widest equation in the proof. When using the `dynamic-single` mode, assumption lines are of variable lengths, determined by the width of the equation right above it.

The `proof` Function

Finally, the `proof` function visualizes the everything discussed so far. It takes

- a set of as parameters which determine the style of the framing objects as a dictionary of the form `(height, thickness, stroke, assume-length, assume-thickness, assume-stroke)` with the lengths (and thicknesses) and strokes being native Typst types.
- an assumption mode (defaults to `fixed`).
- an indexation schema (defaults to `"1"`).
- the array consisting the proof (as a positional argument).

And that's it! For examples see the examples document.