`fitch.sty`: Fitch-style natural deduction macros

Peter Selinger
Dalhousie University

Richard Zach
University of Calgary*

Version 1.0
December 17, 2023

1 Overview

This document describes how to use the `fitch.sty` macros for typesetting Fitch-style natural deduction derivations. To load the macros, simply put `\usepackage{fitch}` into the preamble of your LaTeX file.

Here is a natural deduction derivation, together with the code that produced it:

1	$P \vee Q$	
2	$\neg Q$	
3	$\quad P$	
4	$\quad P$	R, 3
5	$\quad Q$	
6	$\quad \neg Q$	R, 2
7	$\quad \bot$	$\neg$E, 5, 6
8	$\quad P$	$\bot$E, 7
9	P	$\vee$E, 1, 3–4, 5–8

```
$\begin{nd}
  \hypo {1} {P\vee Q}
  \hypo {2} {\neg Q}
  \open
  \hypo {3} {P}
  \have {4} {P}       \r{3}
  \close
  \open
  \hypo {aa} {Q}
  \have {6} {\neg Q} \r{2}
  \have {7} {\bot}   \ne{aa,6}
  \have {8} {P}       \be{7}
  \close
  \have {9} {P}       \oe{1,3-4,aa-8}
\end{nd}$
```

A derivation consists of *lines*, and each line contains a *line label* and a *formula*. Some lines also carry a *justification*. Moreover, each line is either a *hypothesis* or a *derived formula*. Usually, derived formulas carry a justification, whereas hypotheses do not; however, the macros do not enforce this.

Proofs set using `fitch.sty` will have lines numbered automatically in the output. It is possible to override the automatic numbering (see Section 3.6). You can refer to line numbers in the text using `\ndref` (see Section 3.2). Various dimensions, formatting of justifications and line references, and the shorthand macros used to produce rule names can be customized (see Section 4).

New in 1.0

Several new commands and customization options have been introduced in v1.0. It is mostly

*The current maintainer of this package is Richard Zach. The package repository is at https://github.com/OpenLogicProject/fitch/, where you can also report any issues with it.

backwards compatible with earlier versions, but see section 5. In particular, if you redefined any internal `fitch` commands, you will have to change `nd*` to `nd@`.

2 Usage

`nd` (*env.*) Derivations are typeset inside the `nd` environment. By default, the standard `array` environment
`nd` (*env.*) is used to do this, so the `nd` environment must must be used in math mode, i.e., it should be surrounded by `$...$` or `\[...\]`.

New in 1.0

The environment `fitchproof` typesets the proof on its own in text, not math mode. Proofs will be set flush left, with the default `\partopsep` spacing surrounding lists added. You do not have to insert `$` to switch to math mode for `fitchproof`—by default. However, it only works if you generate proofs using the `tabular` environment, e.g., by loading `fitch` with the `arrayenv=tabular` option.

`\hypo` `\have` The commands `\hypo` and `\have` are used to typeset one line of the derivation; `\hypo` is used for hypotheses, and `\have` for derived formulas. Both commands take a label and a formula as an argument. Note that the labels used to identify lines in the derivation need not be actual line numbers; for instance, in the above example, we used the label *aa* instead of 5. In the output, lines are automatically numbered consecutively. Labels may not contain any punctuation characters or spaces.

`\open` `\close` Subderivations are opened and closed with the commands `\open` and `\close`. Finally, the following commands are provided for annotating lines with justifications:

`\r`	reiteration	`\ni`	not introduction
`\ii`	implication introduction	`\ne`	not elimination
`\ie`	implication elimination	`\be`	bottom elimination
`\ai`	and introduction	`\nne`	double negation elimination
`\ae`	and elimination	`\Ai`	forall introduction
`\oi`	or introduction	`\Ae`	forall elimination
`\oe`	or elimination	`\Ei`	exists introduction
		`\Ee`	exists elimination

New in 1.0

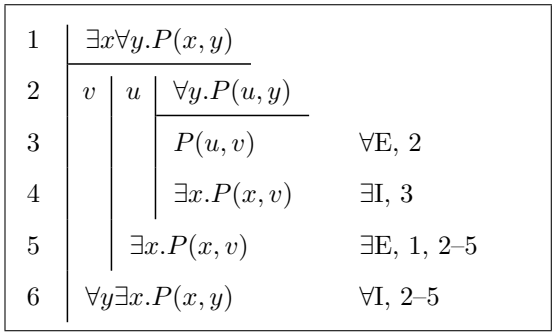
These commands and what they produce can be customized, see Section 4.

Each such command takes a *reference list* as an argument. A reference list is a string made from labels, commas, and hyphens, for instance `1,3a-3b,4a-4d`.

3 Details

3.1 Guards

Some natural deduction derivations with quantifiers use *guards*, as in the following example:




```
1 $
2 \begin{nd}
3   \hypo {1} {\exists x\forall y.P(x,y)}
4   \open[v]
5   \open[u]
6   \hypo {2} {\forall y.P(u,y)}
7   \have {3} {P(u,v)}                    \
        Ae{2}
8   \have {4} {\exists x.P(x,v)}          \
        Ei{3}
9   \close
10  \have {5} {\exists x.P(x,v)}          \
        Ee{1,2-5}
11  \close
12  \have {6} {\forall y\exists x.P(x,y)}
        \Ai{2-5}
13 \end{nd}
14 $
```

The guards v and u in line 2 were typeset by giving optional arguments to the `\open` commands of the respective subderivations.

`\guard` For most purposes, the above way of specifying guards is sufficient. However, there is another method, which allows a more flexible placement of guards: before any line, you can give the command `\guard{u}` to add a guard u to the top-level subderivation at that line, or `\guard[n]{u}` to add a guard to the nth enclosing subderivation at that line. Thus, the above example could have also been typeset by inserting the two commands `\guard{u}` and `\guard[2]{v}` just after the second `\open` statement.

3.2 Label and reference list details

Labels for lines given to the `\have` and `\hypo` commands need not be numeric, although the package will *output* them as consecutive numbers (see Section 3.6 for how to adjust the numbering). Labels, however, may not contain commas, periods, semicolons, hyphens, parentheses, or spaces. In a reference list, spaces are ignored (even within a label!), whereas commas, periods, semicolons, parentheses, and hyphens are copied to the output. All other characters are interpreted as part of a label. Attempting to reference a label which has not been previously defined by any `\hypo` or `\have` command produces a warning message of the form:

New in 0.6

```
! Package fitch Warning: Undefined line reference: lab17.
```

(In earlier versions, this resulted in an error, not a warning.)

`\ndref` Labels defined in an `nd` environment can be referenced in the text with the `\ndref` command. This command takes a reference list as an argument, and produces the corresponding output. However, it is only possible to reference labels *after* the corresponding derivation has been typeset. There is currently no convenient way of defining forward references. Also, if a label is used more than once, `\ndref` will always refer to the most recent time it was used.

3.3 Generic justifications

`\by` Non-standard justifications can be created with the `\by` command. This command takes two arguments: a name and a reference list. For instance, the command `\by{De Morgan}{lab3,lab4}` might produce the output "De Morgan, 3, 4". Note that the justification is typeset in text mode.

New in 1.0

In the default justification format, a comma is automatically inserted between the name and the reference list, unless the reference list is empty. The formatting of justifications can be changed, see Section 4. If the second argument (the reference list) is empty, only the first argument (without formatting or punctuation) is printed. If the first argument is empty, only the reference list is printed.

Since the `\by` command outputs its first argument without additional formatting when the second argument is empty, you can use `\by{...}{}` to produce arbitrary text in the justification. You can use the `\ndref` command here.

1	$A \Rightarrow B$	Premise
2	A	Premise
3	B	1, 2 (but *how?*)
4	$A \wedge B$	2, 3

```
$
\begin{nd}
  \hypo {a} {A \Rightarrow B}
    \by{Premise}{}
  \hypo {b} {A} \by{Premise}{}
  \have {c} {B}
    \by{\ndref{a,b}
      (but \emph{how?})}{}
  \have {d} {A \wedge B} \by{}{b,c}
\end{nd}
$
```

3.4 Scope

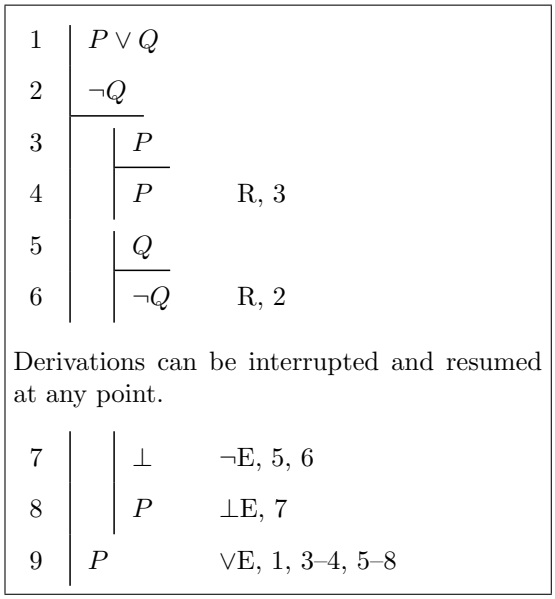
The commands `\hypo`, `\have`, `\open`, `\close`, `\by`, `\r`, `\ii`, and so forth are only available inside an `nd` environment. These commands may have a different meaning elsewhere in the same document. The only commands provided by the `fitch.sty` package which are visible outside an `nd` environment are the command `\ndref` described in Section 3.2, the commands `\ndrules`, `\ndjustformat`, `\ndrefformat`, and `\nddim`, and the dimension `\ndindent` described in Section 4.

3.5 Breaking it across pages

`ndresume` (*env.*)
`fitchproof*` (*env.*)

New in 1.0

The `nd` environment is derived from the LaTeX `array` environment, and thus it does not break across pages automatically. However, if a derivation is too long to fit on a single page, it is possible to split it manually into physically independent, but logically consecutive subparts. For this purpose, the `ndresume` environment is provided to continue a previously interrupted derivation. The environment `fitchproof*` works the same way, except typesets the proof just like the `fitchproof` environment (no need for math mode, flush left, spacing before and after). However, like `fitchproof`, it requires the `arrayenv=tabular` option. Here is an example:



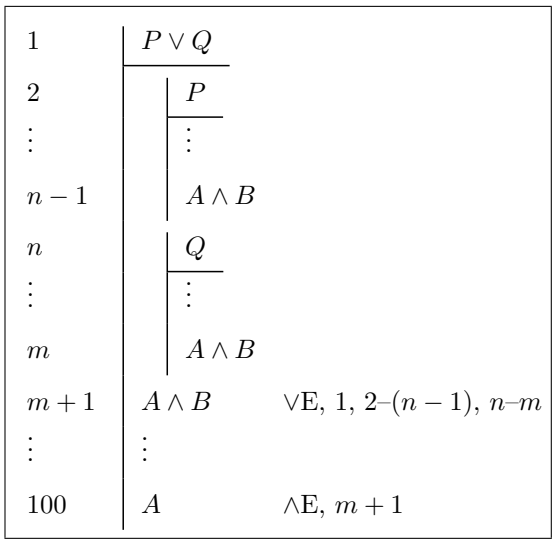

```
\begin{fitchproof}[arrayenv=tabular]
  \hypo {1} {P\vee Q}
  \hypo {2} {\neg Q}
  \open
  \hypo {3} {P}
  \have {4} {P}    \r{3}
  \close
  \open
  \hypo {aa} {Q}
  \have {6} {\neg Q} \r{2}
\end{fitchproof}
Derivations can be interrupted and
resumed at any point.
\begin{fitchproof*}[arrayenv=tabular]
  \have {7} {\bot} \ne{aa,6}
  \have {8} {P}   \be{7}
  \close
  \have {9} {P}   \oe{1,3-4,aa-8}
\end{fitchproof*}
```

New in 1.0

You can also have derivations break across pages automatically. In order to do this, you have to load the `longtable` package, and load `fitch` with the `arrayenv=longtable` option. Since the `longtable` environment works in text (not math) mode, you should then only use `fitchproof`, or the `nd` environment but *not* in text mode. Note that the `longtable` package does not work in 2-column mode or inside a `multicolumn` environment. You can always produce a proof inside a `multicolumn` environment by passing `arrayenv=tabular` as an option to the `nd` or `fitchproof` environment.

3.6 Custom line numbers

One often needs to write derivation schemas, rather than derivations. This often requires the use of symbolic constants such as n, $n+1$, etc, instead of actual line numbers. The `\have` and `\hypo` commands have an optional first argument which is a symbolic constant. For instance, `\have[n]` will cause the current line to be numbered with the symbolic constant n. Subsequent lines are automatically numbered $n+1$ etc. An initial offset can be given as a second optional argument, as in `\have[n][-1]`, which will cause the current line to be numbered $n-1$, the following line n, etc. In an explicit offset is given, the symbolic constant can also be absent: for instance, the command `\have[][7]` resets the current line number to 7. The following example illustrates this behavior:




```
1 $
2 \begin{nd}[justsep=1em]
3   \hypo         {1} {P\vee Q}
4   \open
5   \hypo         {2} {P}
6   \have [\vdots] {3} {\vdots}
7   \have [n][-1] {4} {A\wedge B}
8   \close
9   \open
10  \hypo         {5} {Q}
11  \have [\vdots] {6} {\vdots}
12  \have [m]     {7} {A\wedge B}
13  \close
14  \have         {8} {A\wedge B}
15     \oe{1,2-(4),5-7}
16  \have [\vdots] {9} {\vdots}
17  \have [][100] {10} {A} \ae{8}
18 \end{nd}
19 $
```

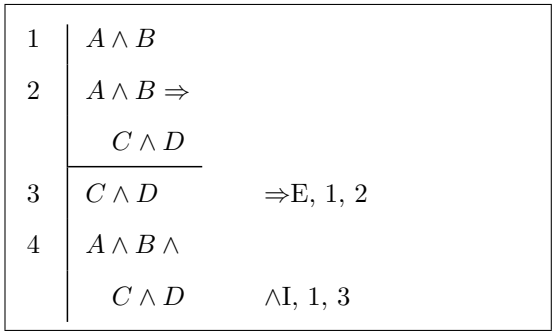
Note that in the justification for line $m+1$, parentheses had to be put around the label 4. There is currently no way of doing this automatically.

Exercise. How does one typeset an empty line number?

Solution. Since `\have[]` has a special meaning as explained above, we need to use `\have[~]` instead.

3.7 Continuation lines

Sometimes one has to typeset a very long formula that does not fit on a single line. As of version 0.5 of the `fitch.sty` macros, it is possible to break a formula into several lines using `\\` as a line separator. Continuation lines are automatically indented, as shown in the following example.




```
1 $
2 \begin{nd}
3   \hypo{1} {A\wedge B}
4   \hypo{2} {A\wedge B\Rightarrow{} \\
5             C\wedge D}
6   \have{3} {C\wedge D} \ie{1,2}
7   \have{4} {A\wedge B\wedge{} \\
8             C\wedge D} \ai{1,3}
9 \end{nd}
10 $
```

`\hypocont` `\havecont` Alternatively, the `\havecont` and `\hypocont` commands can be used to specify each continuation line separately, as the following example illustrates.

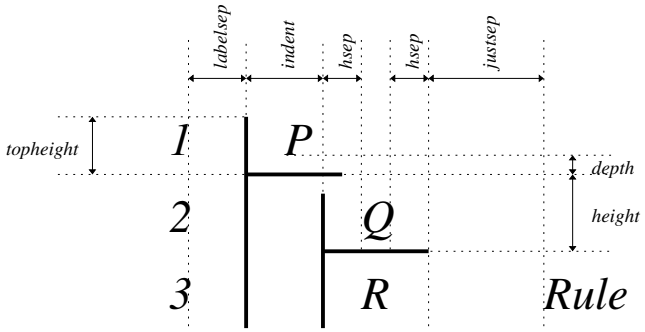
1	$A \wedge B$	
2	$A \wedge B \Rightarrow$	
	$\quad C \wedge D$	
3	$C \wedge D$	$\Rightarrow$E, 1, 2
4	$A \wedge B \wedge$	$\wedge$I, 1, 3
	$\quad C \wedge D$	

```
$
\begin{nd}
  \hypo{1} {A\wedge B}
  \hypo{2} {A\wedge B\Rightarrow{}}
  \hypocont {C\wedge D}
  \have{3} {C\wedge D}       \ie{1,2}
  \have{4} {A\wedge B\wedge{}} \ai{1,3}
  \havecont {C\wedge D}
\end{nd}
$
```

This latter style gives slightly more flexibility in the placement of justifications, since each line and continuation line can have its own justification and its own guard (via the `\guard` command). It also allows a derivation to be interrupted between a line and its continuation, as discussed in Section 3.5.

4 Customization

The relative sizes of the various elements of a natural deduction proof are preset to reasonable values depending on the size of the currently selected font. However, it will sometimes be necessary to customize these dimensions. The customizable dimensions are given in the following diagram.



In addition, ⟨*linethickness*⟩ determines the thickness of scope lines, and ⟨*cindent*⟩ the extra indentation of continuation lines (as discussed in Section 3.7). The default dimensions are:

⟨*height*⟩	4.5ex	⟨*topheight*⟩	3.5ex
⟨*depth*⟩	1.5ex	⟨*labelsep*⟩	1em
⟨*indent*⟩	1.6em	⟨*hsep*⟩	.5em
⟨*justsep*⟩	2.5em	⟨*linethickness*⟩	.2mm
⟨*cindent*⟩	1em		

New in 1.0

To change these default dimensions, pass a list of key-value pairs as package options, as optional arguments to the `nd` or `fitchproof` environment, or use the `\setkeys` command:

```
\usepackage[justsep=1em]{fitch}
\begin{nd}[rules=myrules]
\begin{fitchproof}[linethickness=1pt]
\setkeys{fitch}{hsep=1em,indent=1em}
```

In addition, the macros used to generate the table containing the proof, to format justifications, format line number references, and to initialize macros to produce justifications in the proof can be customized:

option	default
`rules=`⟨*macroname*⟩	`ndrules`
`justformat=`⟨*macroname*⟩	`ndjustformat`
`refformat=`⟨*macroname*⟩	`ndrefformat`
`arrayenv=`⟨*envname*⟩	`array`

For compatibility with earlier versions of `fitch.sty`, the default for ⟨*arrayenv*⟩ is `array`, i.e., the table containing the proof is generated using an `array` environment, and must therefore occur inside math mode. The `fitchproof` and `fitchproof*` environments assume that you use a text table command instead, such as `tabular`. Hence, you should *not* use `fitchproof` in math mode, and you must use the option `arrayenv=tabular` (when loading `fitch`, as an optional argument to `fitchproof`, or using `\setkeys{fitch}{arrayenv=tabular}`. Any other table environment that takes the same table format argument as `array` and `tabular` can be used here, e.g., the `longtable` environment from the `longtable` package (which must be loaded separately).

`\ndrules` The package defines the macros `\ndrules`, which defines the rule macros given at the end of Section 2, using

```
\def\ndrules{%
\def\ii{\by{$\Rightarrow$I}}%
\def\ie{\by{$\Rightarrow$E}}%
\def\Ai{\by{$\forall$I}}%
\def\Ae{\by{$\forall$E}}%
\def\Ei{\by{$\exists$I}}%
\def\Ee{\by{$\exists$E}}%
\def\ai{\by{$\wedge$I}}%
\def\ae{\by{$\wedge$E}}%
\def\ai{\by{$\wedge$I}}%
\def\ae{\by{$\wedge$E}}%
\def\oi{\by{$\vee$I}}%
\def\oe{\by{$\vee$E}}%
\def\ni{\by{$\neg$I}}%
\def\ne{\by{$\neg$E}}%
\def\be{\by{$\bot$E}}%
\def\nne{\by{$\neg\neg$E}}%
\def\r{\by{R}}}
```

`\ndjustformat` The macro `\ndjustformat` is defined as

```
\newcommand{\ndjustformat}[2]{#1, #2}
```

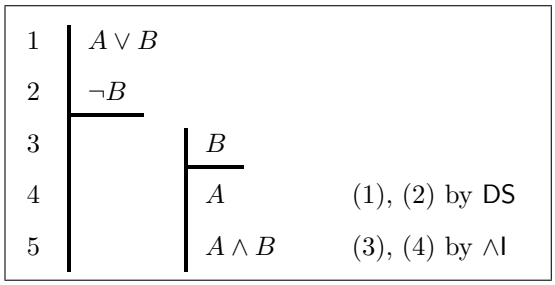
The first argument takes the rule name, the second the reference list. It is used to typeset the justification.

`\ndrefformat` The macro `\ndrefformat` is defined as

```
\newcommand{\ndrefformat}[1]{#1}
```

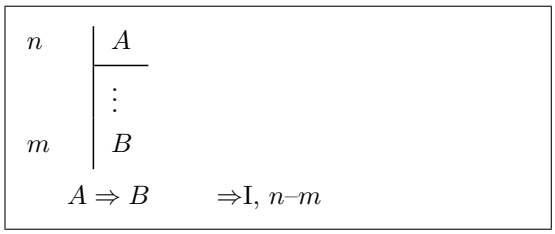
It is used to typeset the line numbers in justifications.

`\ndrules`, `\ndjustformat`, and `\ndrefformat` can be redefined using `\renewcommand`, or you can define your own commands to provide the rule names, the justification format, and line number format, and pass the names (without initial \) as an option to the `\usepackage` or individual `\nd` or `\fitchproof` commands.




```
\newcommand{\myjust}[2]
  {#2 by \textsf{#1}}
\newcommand{\myrules}{
  \ndrules % include standard rules
  \def\ds{\by{DS}}}
\renewcommand{\ndrefformat}[1]{(#1)}
$
\begin{nd}[rules=myrules,
  justformat=myjust,
  indent=1.5cm,
  linethickness=1.5pt,
  justsep=1cm]
  \hypo {1} {A\vee B}
  \hypo {2} {\neg B}
  \open
  \hypo {a} {B}
  \have {3} {A}        \ds{1,2}
  \have {b} {A \wedge B} \ai{a,3}
\end{nd}
$
```

The boolean option `outerline` can be set to false to suppress the leftmost scope line. This may be useful when printing inference rules, e.g.,




```
$
\begin{nd}[outerline=false,
  labelsep=0pt]
  \open
  \hypo [n]{1} {A}
  \have [~]{2} {\raisebox{-1ex}{\vdots
      }}
  \have [m]{3} {B}
  \close
  \have [~]{b} {A \Rightarrow B} \ii
      {1-3}
\end{nd}
$
```

5 Obsolete commands and backwards compatibility

`\nddim` The dimensions can also be changed with the `\nddim` command. The syntax of the command is as follows:

`\nddim{`⟨*height*⟩`}{`⟨*topheight*⟩`}{`⟨*depth*⟩`}{`⟨*labelsep*⟩`}`
`{`⟨*indent*⟩`}{`⟨*hsep*⟩`}{`⟨*justsep*⟩`}{`⟨*linethickness*⟩`}`,

where each of the eight parameters is a dimension. This still works, but using the key-value pair options is the preferred method.

New in 1.0

`\ndindent` In versions before v1.0, the recommended way to change the extra indentation used on continuation lines was to change dimension `\ndindent` directly using `\setlength`. As of v1.0, you should use the `cindent` option instead.

The original code "hid" the internal macros by naming them `\nd*...`. In v1.0 this has been changed to the standard `\nd@...`. Any low-level redefinition of `fitch` internals that uses `*` will break in v1.0.

6 Other comments

The goal was to design a flexible package which would not impose any constraints on the form of derivations, while making typesetting easy. With this package, it is in fact possible to typeset incomplete, ill-formed, or invalid derivations. Sometimes it is pedagogically necessary to do so.

There are no arbitrary limits on the size or nesting depth of a derivation, except for the obvious requirement of fitting horizontally on the printed page.

7 Copyright and license