Fitch-style natural deduction macros

Peter Selinger
University of Ottawa

Version 0.5, February 8, 2005

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 `\input{fitch.sty}` near the beginning 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 number* and a *formula*. Some lines also carry a *justification*. Moreover, each line is either a *hypothesis* or a *derived formula*. Generally, derived formulas carry a justification, whereas hypotheses do not; however, the macros do not enforce this.

Derivations are typeset inside the `nd` environment, which must be used in math mode. 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.

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

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

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`.

2 Details

2.1 Guards

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

1		$\exists x \forall y. P(x,y)$	
2	v u	$\forall y. P(u,y)$	
3		$P(u,v)$	$\forall$E, 2
4		$\exists x. P(x,v)$	$\exists$I, 3
5		$\exists x. P(x,v)$	$\exists$E, 1, 2–5
6		$\forall y \exists x. P(x,y)$	$\forall$I, 2–5

```
\[
\begin{nd}
  \hypo {1} {\exists x\forall y.P(x,y)}
  \open[v]
  \open[u]
  \hypo {2} {\forall y.P(u,y)}
  \have {3} {P(u,v)}                      \Ae{2}
  \have {4} {\exists x.P(x,v)}            \Ei{3}
  \close
  \have {5} {\exists x.P(x,v)}            \Ee{1,2-5}
  \close
  \have {6} {\forall y\exists x.P(x,y)}  \Ai{2-5}
\end{nd}
\]
```

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

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.

2.2 Generic justifications

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. A comma is automatically inserted between the name and the reference list, unless the reference list is empty.

2.3 Label and reference list details

Labels may not contain commas, periods, semicolons, hyphens, parenthesis, or spaces. In a reference list, spaces are ignored (even within a label!), whereas commas, periods, semicolons, parenthesis, 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 an error message of the form:

```
! Undefined natdeduction reference: lab17.
```

2.4 Referencing line numbers in the text

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.

2.5 Scope

The commands `\hypo`, `\have`, `\open`, `\close`, `\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 2.4, and the command `\nddim` and the dimension `\ndindent` described in Section 2.9.

2.6 Breaking it across pages

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. Here is an example:

1	$P \vee Q$	
2	$\neg Q$	
3	$\quad P$	
4	$\quad P$	R, 3
5	$\quad Q$	
6	$\quad \neg Q$	R, 2

Derivations can be interrupted and resumed at any point.

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}
\end{nd}
$

Derivations can be interrupted and
resumed at any point.

$
\begin{ndresume}
  \have {7} {\bot}     \ne{aa,6}
  \have {8} {P}        \be{7}
  \close
  \have {9} {P}        \oe{1,3-4,aa-8}
\end{ndresume}
$
```

2.7 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	$P \vee Q$	
2	$\quad P$	
$\vdots$	$\quad \vdots$	
$n-1$	$\quad A \wedge B$	
n	$\quad Q$	
$\vdots$	$\quad \vdots$	
m	$\quad A \wedge B$	
$m+1$	$A \wedge B$	$\vee$E, 1, 2–$(n-1)$, n–m
$\vdots$	$\vdots$	
100	A	$\wedge$E, $m+1$

```
\[
\begin{nd}
  \hypo          {1} {P\vee Q}
  \open
  \hypo          {2} {P}
  \have [\vdots] {3} {\vdots}
  \have [n][-1]  {4} {A\wedge B}
  \close
  \open
  \hypo          {5} {Q}
  \have [\vdots] {6} {\vdots}
  \have [m]      {7} {A\wedge B}
  \close
  \have          {8} {A\wedge B}  \oe{1,2-(4),5-7}
  \have [\vdots] {9} {\vdots}
  \have [][100] {10} {A}          \ae{8}
\end{nd}
\]
```

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.

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

```
\[
\begin{nd}
  \hypo{1}  {A\wedge B}
  \hypo{2}  {A\wedge B\rightarrow{} \\ C\wedge D}
  \have{3}  {C\wedge D}                        \ie{1,2}
  \have{4}  {A\wedge B\wedge{} \\ C\wedge D}  \ai{1,3}
\end{nd}
\]
```

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
	$\quad C \wedge D$	

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

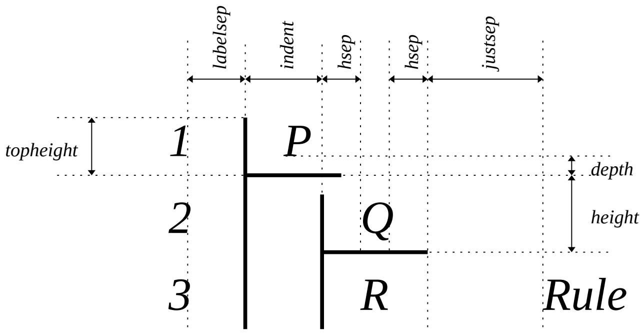
This latter style gives slightly more flexibility in the placement of justifications, since each line and continuation line can have its own justification and its own guard (via the `\guard` command). It also allows a derivation to be interrupted between a line and its continuation, as discussed in Section 2.6.

2.9 Customizing dimensions

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. This can be achieved 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. The meaning of the first seven parameters is shown in the following illustrations; *linethickness* is simply the thickness of the lines.



The default dimensions are:

`\nddim{4.5ex}{3.5ex}{1.5ex}{1em}{1.6em}{.5em}{2.5em}{.2mm}`.

In addition, there is a dimension called `\ndindent`, controls the amount of extra indentation used on continuation lines as discussed in Section 2.8. It can be changed and is `1ex` by default.

2.10 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.

3 Copyright and license

The following notice applies to the `fitch.sty` macros.