Documentation for the LaTeX package `NaturalDeduction`

Nicolas Fillion
Simon Fraser University

May 15, 2020

Contents

1 Introduction to the `NaturalDeduction` package

Experienced logic instructors know that the *mechanics* of doing logic is of *incontestable pedagogical value* in logic education. This pertains to what philosophers and educators often call *procedural knowledge*, or more simply "know-how." Teaching procedural knowledge effectively requires of instructors that instructors solve in the presence carefully chosen problems, in a way such that

- they go through each step of the procedure **in the order in which the thinking should be done**;
- the reasons behind why the thinking should be done in this order be explained.

The teaching of natural deduction in any decent course on formal logic will exactly meet those two standards. Normally, experiences instructors will solve examples on a blackboard, and explain the underlying thinking verbally. However, this is not always possible, especially in the context of online classes, in which case other modes of communication must be used. However, at the time of writing the first public release of this document (May 2020), there is no straightforward and satisfactory way of capturing the dynamical aspect of doing natural deduction in LaTeX. This document presents a solution to this pedagogical problem.

Specifically, this document describes the LaTeX package `NaturalDeduction.sty`, designed to produce Fitch-style natural deduction diagrams in an environment that supports TikZ additions (TikZ is the best graphical package for LaTeX) and Beamer overlay (Beamer is the best LaTeX class for slideshow presentations). The package is based on `TikZ` commands that can accomplish what the package `fitch.sty` accomplishes, but provides more readily customizable options to the user, and a more interactive experience to the reader/student. We will also see how animated deductions can be included in regular PDF notes.

2 How to use the `NaturalDeduction` package

A sample Fitch diagram with default options is included in figure 1. The procedure to generate and customize such a Fitch diagram is described below.

In this early development stage, the most straightforward way to use this package is to simply add the file `NaturalDeduction.sty` to your working directory containing you LaTeX document. In later stages, the package will be installable and usable as other packages included in the standard LaTeX distributions. To use the package in a LaTeX document, the user proceeds in the usual way by adding the following command to the preamble of the document (*i.e.*, between the lines `\documentclass{...}` and `\begin{document}`):

```
\usepackage{NaturalDeduction}
```

As usual, one can specify various options when the package is loaded. The syntax is the same as most packages:

```
\usepackage[option1=value1,option2=value2, etc]{
   NaturalDeduction}
```

The available options are described in section 3.

The construction of a Fitch diagram is based on the new environment `fitch` and on four simple commands. The environment is used in the usual way:

```
\begin{fitch}
... insert commands here ...
\end{fitch}
```

(1)	$R \vee (\neg P \wedge \neg Q)$	Prem
(2)	$P \vee Q$	Supp/CP
(3)	P	Case 1
(4)	$\neg R$	Supp/RA
(5)	$\neg P \wedge \neg Q$	DS 1,4
(6)	$\neg P$	$\wedge$Elim
(7)	$\bot$	$\bot$Int 3,6
(8)	R	RA 4-7
(9)	Q	Case 2
(10)	$\neg R$	Supp/RA
(11)	$\neg P \wedge \neg Q$	DS 1,10
(12)	$\neg Q$	$\wedge$Elim 11
(13)	$\bot$	$\bot$Int 9,12
(14)	R	RA 10-13
(15)	R	$\vee$Elim 2,3-8,9,14
(16)	$(P \vee Q) \rightarrow R$	CP 2-15

Figure 1: Sample Fitch diagram for propositional logic.

Each of the package options described in section 3 can be passed to individual instances of the `fitch` environment as follows:

```
\begin{fitch}[option1=value1,option2=value2, etc]
```

The commands inserted within the environment `fitch` include as many instances of the following commands as the user requires:

1. `\prem`
2. `\getline`
3. `\open`
4. `\close`
5. `\comment`

As the `fitch` environment is implemented in TikZ, users can also add any TikZ commands—which can be very predagogically beneficial. See section 5 for more details.

The `\open` and `\close` commands take no arguments; they are used to open and close subproofs, respectively prior to the first line of the subproof and after the last line of the subproof. The number of instances of the command `\open` and `\close` must be the same within each instance of the environment `fitch`. The actual content of the lines in the Fitch diagrams are provided by entering

```
\prem{line number}{logical formula}{justification}
```

and

```
\getline{line number}{logical formula}{justification}
```

By default, each of the three entries will be formatted in text mode, so the insertion of any mathematical symbol must be included explicitly by `$...$`. Also, by default, the line number will correspond to its vertical position in the deduction diagram, from top to bottom. The line number must be an integer.

The two commands are accomplishing essentially the same thing; the main difference, from the user point of view, is that a horizontal line is added by default after all the premises are entered to separate them from the rest of the deduction, whereas when a new subproof is opened a horizontal line is added under the very first statement of the subproof to separate it from the rest of the subproof. Section 3 describes an option to make the formula automatically formatted in math mode.

One last command is `\comment`, to add comments to the right of the justification of a line. It works as follows:

```
\comment[option]{n}{text}
```

This command will add a comment to line `n`, specifically, it will print '`text`'.

3 Environment options

Add sample alternative values.
Add global styles.

Two categories of options can be passed to the package when it is loaded in the preamble using the command `\usepackage[options]{NaturalDeduction}` or when instances of the environment are created using `\begin{fitch}[options]`, namely, *Boolean* and *String* options. Boolean options are such that they are simply attributed the values `true` or `false`, whereas string options can take any value that is an adequate variation of the default value. This section lists both categories of options, together with their default value and description. Let us begin with string options:

`JustPos` Specifies the horizontal position of the justification with respect to the left-hand side of the formula.
Default value: `3.5`

`JustSpace` Specifies the horizontal space between premise and supposition lines and justifications.
Default value: `0.25`

`LineSpacing` Specifies the vertical space between lines.
Default value: `0.7`

`IndentWidth` Specifies the horizontal width of indentation of subproofs.
Default value: `0.7`

`NumberSpacing` Specifies the horizontal space between the right-hand side of line numbers and the left-hand side of formulas in the main deduction scope.
Default value: `0.0`

`FontSize` Specifies the font size of the text in nodes.
Default value: `normalsize`

`Scale` Rescales all the dimensions of the Fitch diagram (but does not affect font size).
Default value: `1`

`CommentWidth` Determines the width of the comment node.
Default value: `5cm`

Now, let us turn to the Boolean options:

`PremLine` Specifies whether or not to include a line between the premises and other lines.
Default value: `true`

`SuppLine` Specifies whether or not to include a line between the supposition of a subproof and the rest of it.
Default value: `true`

`MainScopeLine` Specifies whether or not to include a scope line for the main scope that begins on line 1 and ends with the last line of the deduction.
Default value: `true`

`ScopeLines` Specifies whether or not to include a scope line for each subproof.
Default value: `true`

`NoStructure` Specifies whether or not all structure should be removed to generate a “flat” Fitch diagrams.
Default value: `false`

`MathModeFormula` Formats the formula given in `\prem` and `\getline` in math mode.
Default value: `false`

`Overlay` Includes overlay information. Only use in the Beamer class. See section 6.
Default value: `false`

More options will be implemented in the near future. There are also “global styles” defined:

`BergmannStyle` Imitates style of Fitch diagrams in the Logic Book.
Default value: `false`

`ArthurStyle` Imitates style of Fitch diagrams in Introduction to Logic, by Richar Arthur.
Default value: `false`

If you’d like to see more global styles, just send me an email and I’ll add it is possible.

4 Command options

Two options are currently available for `\prem` and `\getline`, which are inputed as follows:

```
\prem[option1=value1,option2=value2, etc]{line number}{
   logical formula}{justification}
\getline[option1=value1,option2=value2, etc]{line number
   }{logical formula}{justification}
```

The available options are the following:

`nonumber` Displays a line with no line number.
Default value: `false`

`number=value` Displays ‘`value`’ as line number.
Default value: `0` (returns the line number given in `\prem` or `\getline`, and can be replaced by any LaTeX commands)

Additional options are described in section 6. In the future, color options and other stylistic features will be available.

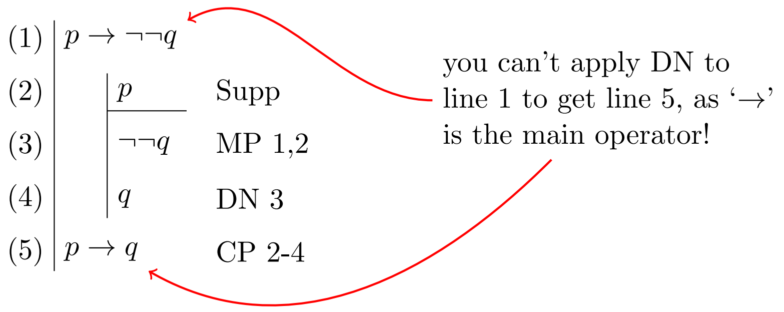

Figure 2: Example of TikZ additions

5 Nodes labeling for TikZ support

Since the environment `fitch` is implemented in TikZ, it is possible to add things to Fitch diagrams using TikZ code. For instance, consider the following:

```
\begin{fitch}[MathModeFormula,JustPos=2]
\getline{1}{p\to \neg\neg q}{}
\open
\getline{2}{p}{Supp}
\getline{3}{\neg\neg q}{MP 1,2}
\getline{4}{q}{DN 3}
\close
\getline{5}{p\to q}{CP 2-4}
% Additional TikZ code to add a comment and arrows.
\draw (5,-1.5) node[anchor=west,text width=1.75in] (
   comment) {you can't apply DN to line 1 to get line 5,
   as `$\to$' is the main operator!};
\draw (comment) edge[->,thick,red,out=180,in=30] (1_2.
   north east);
\draw (comment) edge[->,thick,red,out=225,in=-30] (5_2.
   south east);
\end{fitch}
```

See figure 2. The idea is that to each line generated by `\prem` and `\getline` there corresponds three nodes. Suppose the line number is `n`. The nodes are named as follows:

Line number node `n_1`

Formula node `n_2`

Justification node `n_3`

TikZ code using node names can be used accordingly.

6 Overlay in Beamer

At the moment, four basic overlay options are available. To activate them, simply ensure that

```
Overlay=true
```

is set and that the document is using the beamer class. If `Overlay=false` the overlay options described below will simply be ignored, and no error will result, whether or not the `beamer` class is loaded.

There are three options that can be passed for to the commands `\prem` and `\getline`:

oa number of the slides on which the number, formula, and justification of the line appears. Trumped by on, of, and oj (see below) if more details are needed.

on number of the slides on which the number of the line appears

of number of the slides on which the formula of the line appears

oj number of the slides on which the justification of the line appears

oc number of the slides on which the justification of the comment appears.

Note that, if there is a clash between one of `on`, `of`, or `oj` and `oa`, the first three are given priority.

Moveover, one option can be passed to the command `\close` by entering

```
\close[n]
```

for any natural number `n`. If `n` isn't specified, it is set to `1` by default. This number should be chosen so as to match the number of the slide at which the formula above the supposition line appears.

Here is a full example of a beamer frame with full overlay and commentary for students, the result of which is displayed in the animation in the next section:

```
\begin{fitch}[MathModeFormula,Overlay,JustPos=4.5cm,
   FontSize=small,CommentWidth=3.5cm,LineSpacing=.55]
\prem{1}{\forall x Rxx}{Prem}
\prem{2}{\forall x,y,z(Rxy\wedge Rxz \to Ryz)}{Prem}
␣␣\comment[oc=1]{2}{As always, first write down your
   premises, leave some blank space, and write down your
   conclusion.}
```

```
%Open a subproof
\open
\getline[oa=7]{3}{Rab\wedge Rbc}{Supp}
␣␣\comment[oc=7]{3}{As always, for CP, we write the
   antecedent of the subgoal on the first line of the
   subproof, and the consequence on the last line.}
\getline[oa=16]{4}{Rab}{$\wedge$Elim}
\getline[oa=17]{5}{Raa}{$\forall$Elim 1}
\getline[oa=18,of=15]{6}{Rab\wedge Raa}{$\wedge$Int 4,5}
\getline[oa=18,of=13]{7}{R\underline{\phantom{a}}b\wedge
   R\underline{\phantom{a}}a\to Rba}{$\forall$Elim$\times
    3$ 2}
\getline[oa=18,of=14]{7}{R\underline{a}b\wedge R\
   underline{a}a\to Rba}{$\forall$Elim$\times 3$ 2}
␣␣\comment[oc=14]{7}{Do not forget that you can
   instantiate two variables with $a$!}
\getline[oa=18,of=12]{8}{Rba}{MP 7,6}
\getline[oa=18,of=12]{9}{Rbc}{$\wedge$Elim 3}
␣␣\comment[oc=12]{9}{$Rbc$ is easy to get. But how will
   we obtain $Rba$? We'll \textbf{again} use the same
   trick we used below.}
\getline[oa=18,of=11]{10}{Rba\wedge Rbc}{$\wedge$Int 8,9}
␣␣\comment[oc=11]{10}{Let's try to deduce the antecedent
   to set up our MP.}
\getline[oa=18,of=9]{11}{R\underline{\phantom{b}}a\wedge
   R\underline{\phantom{b}}c\to Rac}{$\forall$Elim$\
   times3$ 2}
␣␣\comment[oc=8]{11}{Here's a good trick for such
   deductions: Make $Rac$ match the consequent of the
   second premise, to reach our sub-subgoal by MP.}
␣␣\comment[oc=9]{11}{That means we'll instantiate $y$
   with $a$ and $z$ with $c$ in line 2.}
\getline[oa=18,of=10]{11}{R\underline{b}a\wedge R\
   underline{b}c\to Rac}{$\forall$Elim$\times3$ 2}
␣␣\comment[oc=10]{11}{But what will we instantiate $x$
   with? We'll try to match information we've already got
   . $b$ would be a good guess.}
\getline[oa=18,of=7]{12}{Rac}{MP 11,10}
\close[7]
%Close the subproof
\getline[oa=18,oa=5]{13}{Rab\wedge Rbc\to Rac}{CP 3-12}
␣␣\comment[oc=4]{13}{As a result, what formula should be
```

```
    taken as subgoal?}
␣\comment[oc=5]{13}{Notice that the choice of $a,b,c$
    doesn't matter, as long as the two conditions on $\
    forall$Int (remember?) are met.}
␣\comment[oc=6]{13}{Now, this subgoal is a conditional.
    So our sub-subgoal is a CP.}
\getline{14}{(\forall x,y,z)(Rxy\wedge Ryz\to Rxz)}{$\
    forall$Int$\times3$ 14}
␣\comment[oc=2]{14}{Next, look at the conclusion. What
    is the main operator(s)? What strategy does it suggest
    ?}
␣\comment[oc=3]{14}{Since it is a sequence of three $\
    forall$s, we will reach 10 by using $\forall$Int$\
    times 3$.}
\end{fitch}
```

7 Embedding animated deductions in PDF notes

Embedding animated deductions in PDF notes becomes very easy with this package. The first think to do is to load the package `animate`. Thus, merely add the following line to your preamble:

```
\usepackage{animate}
```

The next step is to produce, in a separate file, a Beamer presentation with overlay and commentaries, as needed. In this case, my Beamer presentation is simply called `BeamerExample.pdf`. I can include an animated deduction in my PDF with just this one simple line of code:

```
\animategraphics[controls={play,step,stop},trim=0cm 0cm 0
    cm 0cm,poster=last]{1}{BeamerExample}{}{}
```

The result is in figure 3. To view the animation, it is essential to use a PDF viewer such as **Acrobat** or **X-Change**.

I hope the pedagogical virtues of including such animated deductions in notes is clear. LaTeX's leitmotiv is to to use all the appropriate resources that we have available to us when sharing electronic document, instead of simply using our computer as a glorified typewriting machine. This is exactly in the spirit of this request.

8 Implementation

```
%%
%% This is file ‘NaturalDeduction.sty’.
```

(1)	$\forall x Rxx$	Prem
(2)	$\forall x, y, z(Rxy \wedge Rxz \to Ryz)$	Prem
(3)	$Rab \wedge Rbc$	Supp
(4)	Rab	∧Elim
(5)	Raa	∀Elim 1
(6)	$Rab \wedge Raa$	∧Int 4,5
(7)	$R\underline{a}b \wedge R\underline{a}a \to Rba$	∀Elim×3 2
(8)	Rba	MP 7,6
(9)	Rbc	∧Elim 3
(10)	$Rba \wedge Rbc$	∧Int 8,9
(11)	$R\underline{b}a \wedge R\underline{b}c \to Rac$	∀Elim×3 2
(12)	Rac	MP 11,10
(13)	$Rab \wedge Rbc \to Rac$	CP 3-12
(14)	$(\forall x, y, z)(Rxy \wedge Ryz \to Rxz)$	∀Int×3 14

|◁ < ◁ ▷ > ▷|

Figure 3: An animated deduction to include in PDF notes

```
%%
%% Copyright (C) 2015-2020
%% by Nicolas Fillion <nfillion@sfu.ca>
%%
%% A flexible set of macros for Fitch-style natural deduction that supports overlay in Beamer and other TikZ-related options.
%% Author: Nicolas Fillion, Simon Fraser University
%% Created: July 20, 2015
%% Modified: May 13, 2020
%% Version: 1.3
%% Documentation: NaturalDeductionSTY-Documentation.tex
%% URL: http://www.nfillion.com
%%
%% LICENSE:
%% This program is free software that can be distributed and/or modified under the conditions
%% of the LaTeX Project Public License, either version 1.3c or the latest version, as the case
%% may be. The latest version of this license is
```

```
available at http://www.latex-project.org/lppl.txt
%% and version 1.3c or later is part of all distributions of LaTeX version 2005/12/01 or later.
%%
%% This work is "maintained" (as per the LPPL maintenance status) by Nicolas Fillion. Please report any bug or requests for
%% additional functionality by email.
%%
%% USAGE EXAMPLE WITHOUT OVERLAY:
%%
%% The following is a simple example illustrating the usage of this %% package.  For detailed
%% instructions and additional functionality, see the user guide, which can be found in the file
%% NaturalDeductionSTY-Documentation.tex.
%% \begin{fitch}
%% \prem{1}{\forall x Px}{Prem}
%% \prem{2}{\exists x\neg Px}{Prem}
%% \open
%% \getline{3}{\forall z (Pz\vee Qz)}{Supp/CP}
%% \open
%% \getline{4}{\neg R}{Supp/RA}
%% \getline{5}{\neg Pi}{$\exists$E 2}
%% \getline{6}{Pi}{$\forall$E 1}
%% \getline{7}{\bot}{$\bot$I 5,6}
%% \close
%% \getline{8}{R}{RA 4--7}
%% \close
%% \getline[number=changed number]{9}{\forall z (Pz\vee Qz)\to R}{CP 2--8}
%% \end{fitch}
%%
%% Package options with default values
%% use as \usepackage[option1,option2,etc]{NaturalDeduction} or as \begin{fitch}[option1,option2,etc.]
%% JustPos=3.5
%% JustSpace=.25
%% LineSpacing=0.7
%% IndentWidth=0.7
%% NumberSpacing=0
%% FontSize=normalsize
%% Scale=1
%% PremLine=true
%% SuppLine=true
%% MainScopeLine=true
%% ScopeLines=true
%% NoStructure=true
%% MathModeFormula=false
%%
%% Options for the commands \prem and \getline
%% number=x % to display a different string as line number
%% nonumber=false
%%
%% add beamer options on, of, oj for the uncover number of each node of a line
%%
%% Package Identification
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{NaturalDeduction}[2018/01/09 v1.3 Natural Deduction Package based on TikZ and supporting Beamer overlay]
%
%
%% Package Requirements
\RequirePackage{tikz}
\RequirePackage{kvoptions}
\RequirePackage{fmtcount}
%
%
%% Set key values family and prefix
\SetupKeyvalOptions{
␣␣family=ND,
␣␣prefix=ND@}␣␣␣
%Declare options
% 1. Package loading options
\DeclareStringOption[3.5]{JustPos} %optional setup option for justification position
\DeclareStringOption[.25]{JustSpace} %optional setup option for space between supposition lines and justification
\DeclareStringOption[.7]{LineSpacing} %optional setup
```

```
option for space between lines
\DeclareStringOption[.7]{IndentWidth} %optional setup for the width of indentation of subproofs
\DeclareStringOption[0]{NumberSpacing} %optional setup for the space between line numbers and main scope line
\DeclareStringOption[normalsize]{FontSize}
\DeclareStringOption[1]{Scale}
\DeclareStringOption[5cm]{CommentWidth}
%
\DeclareBoolOption[true]{PremLine}
\DeclareBoolOption[true]{SuppLine}
\DeclareBoolOption[true]{MainScopeLine}
\DeclareBoolOption[true]{ScopeLines}
\DeclareBoolOption[false]{NoStructure}
\DeclareBoolOption[false]{ArthurStyle}
\DeclareBoolOption[false]{BergmannStyle}
\DeclareBoolOption[false]{Overlay}
\DeclareBoolOption[false]{MathModeFormula}
% 2. Command options
\DeclareStringOption[0]{number} %optional line number parameter
\DeclareBoolOption[false]{nonumber}
\DeclareStringOption[1]{on}
\DeclareStringOption[1]{of}
\DeclareStringOption[1]{oj}
\DeclareStringOption[1]{oa}
\DeclareStringOption[1]{oc}
% Process Options
%\setkeys{ND}{option}%that would set defaults.␣
\ProcessKeyvalOptions{ND}


% Setup the consequences of the omnibus option NoStructure
\ifND@NoStructure
␣\setkeys{ND}{PremLine=false,SuppLine=false,MainScopeLine=false,ScopeLines=false,IndentWidth=0}
\fi

% Setup the consequences of the omnibus option NoStructure
\ifND@ArthurStyle
```

```
␣␣\setkeys{ND}{PremLine=false,SuppLine=false,MainScopeLine=false,ScopeLines=true}
\fi

\ifND@BergmannStyle
␣␣\setkeys{ND}{PremLine=true,SuppLine=true,MainScopeLine=true,ScopeLines=true}
\fi


% Define counters
\newcounter{leftindent} % a counter used throughout keeping track of the subproof number
\newcounter{numberoflines} %count how many lines (\line and \prem) are in the proof
\newcounter{numberofprems} % count how many premises are in the proof

% Define new environment
\newenvironment{fitch}[1][]
␣␣%At the beginning of the environment
␣␣{
␣␣\setkeys{ND}{#1}
␣␣\begin{tikzpicture}[font=\csname \ND@FontSize\endcsname,scale=\ND@Scale]
␣␣}
␣␣%At the end of the environment %%%%%%%%%%%%%%%%%%%%%
␣␣{
␣␣%If no line, add a token empty line to avoid error messages
␣␣\ifnum\thenumberoflines<1
␣␣␣␣\draw (0,0) node (emptynode) {};
␣␣␣␣\def\topline_2{emptynode}
␣␣␣␣\def\bottomline_2{emptynode}
␣␣\fi
␣␣%Store coordinate of top of diagram
␣␣\newdimen\topcoord
␣␣\pgfextracty{\topcoord}{\pgfpointanchor{\topline_2}{north}}
␣␣\newdimen\bottomcoord
␣␣\pgfextracty{\bottomcoord}{\pgfpointanchor{\bottomline_2}{south}}
```

```
  %Add a main scope line
  \ifnum\thenumberoflines>0
    \ifND@MainScopeLine
      \draw (0,\topcoord) edge (0,\bottomcoord);
    \fi
  \fi
  %Add a hypothesis line
  \ifnum\thenumberofprems>0
  \ifND@PremLine
    \ifcsdef{lastprem}{\draw ([yshift=-.5\baselineskip]\lastprem_2.west) edge ([xshift=-\ND@JustSpace cm, yshift=-.5\baselineskip]\lastprem_3.west);}
  \fi
  \else
  % \draw (0,0) node {there are no premises in this diagram!};
  \fi
  %Closing commands
  \end{tikzpicture}
  % Reset counters when the environment closes so latex doesn't remember it next time a fitch environment is opened
  \setcounter{leftindent}{0}
  \setcounter{numberoflines}{0}
  \setcounter{numberofprems}{0}
  }


% Define the commands \open and \close managing subproofs
\newcommand{\open}{%
  \addtocounter{leftindent}{1}
  }
\newcommand{\close}[1][]{%
  %Put code for the scope line here, with a conditional on the style Boolean
  \def\topnodenum{\csname firstline\roman{leftindent}\endcsname}
  \def\bottomnodenum{\csname lastline\roman{leftindent}\endcsname} 
  \ifND@ScopeLines
    \ifND@Overlay
      \def\temp{#1}\ifx\temp\empty % make sure the
```

```
argument of \close isn't empty to avoid error
␣␣␣␣\draw (\topnodenum_2.north west) edge (\bottomnodenum_2.south west);
␣␣␣␣\else␣␣␣
␣␣␣␣\uncover<#1->{\draw (\topnodenum_2.north west) edge (\bottomnodenum_2.south west);}
␣␣␣␣\fi
␣␣␣\else
␣␣␣\draw (\topnodenum_2.north west) edge (\bottomnodenum_2.south west);
␣␣␣\fi␣
␣␣\fi
␣␣%Adjust counter
␣␣\expandafter\let\csname firstline\roman{leftindent}\endcsname\undefined
␣␣\expandafter\let\csname lastline\roman{leftindent}\endcsname\undefined
␣␣\let\topnodenum\undefined
␣␣\let\bottomnodenum\undefined
␣␣\addtocounter{leftindent}{-1}␣
␣␣}
␣␣

%Dynamic macro definition for the \firstline and \lastline set of commands (one for each subpoof).
\newcommand\updatescopeinfo[2]{%
␣␣%Family of \firstlines
␣␣\ifcsdef{firstline#1}{
␣␣␣␣%command is defined
␣␣␣␣\ifnum#2<\csname firstline#1\endcsname
␣␣␣␣␣␣\expandafter\renewcommand\csname firstline#1\endcsname{#2}
␣␣␣␣\fi
␣␣␣␣}{
␣␣␣␣%command is undefined
␣␣␣␣\expandafter\newcommand\csname firstline#1\endcsname{#2}
␣␣␣␣}
␣␣%Family of \lastlines
␣␣\ifcsdef{lastline#1}{
␣␣␣␣%command is defined
␣␣␣␣\ifnum#2>\csname lastline#1\endcsname
```

```
        \expandafter\renewcommand\csname lastline#1\endcsname{#2}
      \fi
      }{
      %command is undefined
      \expandafter\newcommand\csname lastline#1\endcsname{#2}
      }
}


%Keep track of the line number of the first and last line to draw main scope line
\newcommand\storetopbottomcoord[1]{%
  \ifcsdef{topline}{
    \ifnum\topline>#1
      \def\topline{#1}
    \fi
    }{
    \def\topline{#1}
    }
  \ifcsdef{bottomline}{
    \ifnum\bottomline<#1
      \def\bottomline{#1}
    \fi
    }{
    \def\bottomline{#1}
    }
  } 


%The main line command
\DeclareRobustCommand{\getline}[4][]{%
  %Set the keys
  \setkeys{ND}{#1}
  %Process batch overlay information
  %Update scope line information
  \updatescopeinfo{\roman{leftindent}}{#2}
  %Update counters
  \addtocounter{numberoflines}{1}
  %Conditionally define the line number tag
  \if\ND@number0
```

```
␣\def\numbertag{#2}
␣\else
␣\def\numbertag{\ND@number}
␣\fi
␣% Ensure that on, of, oj are defined. First, with keyval, else oa, else 1.
␣\ifx\ND@on\relax
␣␣\ifx\ND@oa\relax
␣␣\def\ND@on{1}
␣␣\else
␣␣\def\ND@on{\ND@oa}
␣␣\fi
␣\else
␣\fi
␣%
␣\ifx\ND@of\relax
␣␣\ifx\ND@oa\relax
␣␣\def\ND@of{1}
␣␣\else
␣␣\def\ND@of{\ND@oa}
␣␣\fi
␣\else
␣\fi
␣%
␣\ifx\ND@oj\relax
␣␣\ifx\ND@oa\relax
␣␣\def\ND@oj{1}
␣␣\else
␣␣\def\ND@oj{\ND@oa}
␣␣\fi
␣\else
␣\fi
␣%Actually draw the line
␣%Number node
␣\ifND@Overlay
␣␣\uncover<\ND@on->{\draw (-\ND@NumberSpacing,-#2*\ND@LineSpacing) node[anchor=east] (#2_1) {\ifND@nonumber \else (\numbertag)\fi };}
␣\else
␣\draw (-\ND@NumberSpacing,-#2*\ND@LineSpacing) node[anchor=east] (#2_1) {\ifND@nonumber \else (\numbertag) \fi };
```

```
␣\fi
␣%Formula node
␣\ifND@Overlay
␣␣\uncover<\ND@of->{ \draw (\value{leftindent}*\ND@IndentWidth,-#2*\ND@LineSpacing) node[anchor=west] (#2_2) {
␣␣␣\ifND@MathModeFormula\ensuremath{#3}\else#3\fi
␣␣␣};}
␣\else
␣\draw (\value{leftindent} * \ND@IndentWidth, - #2 * \ND@LineSpacing) node[anchor=west] (#2_2) {
␣␣\ifND@MathModeFormula\ensuremath{#3}\else #3\fi
␣␣};
␣\fi
␣%Justification node
␣\ifND@Overlay
␣␣\uncover<\ND@oj->{␣\draw (\ND@JustPos,-#2*\ND@LineSpacing) node[anchor=west] (#2_3) {#4}; }
␣\else
␣\draw (\ND@JustPos,-#2*\ND@LineSpacing) node[anchor=west] (#2_3) {#4};
␣\fi
␣%Store y coordinate of top and bottom of diagram
␣\storetopbottomcoord{#2}
␣%Decide whether to add a hypothesis horizontal line
␣\ifND@SuppLine
␣\if\theleftindent0
␣\else
␣␣\ifnum\numexpr\csname firstline\roman{leftindent}\endcsname =#2
␣␣␣\ifND@Overlay
␣␣␣\uncover<\ND@of->{\draw ([yshift=-.5\baselineskip]#2_2.west) edge ([xshift=-\ND@JustSpace cm,yshift=-.5\baselineskip]#2_3.west);}
␣␣␣\else
␣␣␣\draw ([yshift=-.5\baselineskip]#2_2.west) edge ([xshift=-\ND@JustSpace cm,yshift=-.5\baselineskip]#2_3.west);
␣␣␣\fi
␣␣\fi
␣\fi
␣\fi
```

```
 %Keys to reset
  \def\ND@number{0}
  \let\ND@oa\relax
  \let\ND@on\relax
  \let\ND@of\relax
  \let\ND@oj\relax
 \ND@nonumberfalse
}
 

%The prem command
\DeclareRobustCommand{\prem}[4][]{%
 %Set the keys
 \let\ND@on\relax
 \let\ND@of\relax
 \let\ND@oj\relax
 \setkeys{ND}{#1}
 %Update scope line information
 \updatescopeinfo{\roman{leftindent}}{#2}
 % Update counters
 \addtocounter{numberoflines}{1}
 \addtocounter{numberofprems}{1}
 %Conditionally define the line number tag
 \if\ND@number0
 \def\numbertag{#2}
 \else
 \def\numbertag{\ND@number}
 \fi
 % Ensure that on, of, oj are defined. First, with keyval, else oa, else 1.
 \ifx\ND@on\relax
  \ifx\ND@oa\relax
  \def\ND@on{1}
  \else
  \def\ND@on{\ND@oa}
  \fi
 \else
 \fi
 %
 \ifx\ND@of\relax
  \ifx\ND@oa\relax
  \def\ND@of{1}
```

```
␣␣␣␣\else
␣␣␣␣\def\ND@of{\ND@oa}
␣␣␣␣\fi
␣␣\else
␣␣\fi
␣␣%
␣␣\ifx\ND@oj\relax
␣␣␣␣\ifx\ND@oa\relax
␣␣␣␣\def\ND@oj{1}
␣␣␣␣\else
␣␣␣␣\def\ND@oj{\ND@oa}
␣␣␣␣\fi
␣␣\else
␣␣\fi
␣␣%Actually draw the line
␣␣%Number node␣␣
␣␣\ifND@Overlay
␣␣␣␣\uncover<\ND@on->{\draw (-\ND@NumberSpacing,-#2*\ND@LineSpacing) node[anchor=east] (#2_1) {\ifND@nonumber \else (\numbertag)\fi };}
␣␣\else
␣␣\draw (-\ND@NumberSpacing,-#2*\ND@LineSpacing) node[anchor=east] (#2_1) {\ifND@nonumber \else (\numbertag)\fi };
␣␣\fi␣
␣␣%Formula node
␣␣\ifND@Overlay
␣␣␣␣\uncover<\ND@of->{␣␣\draw (\value{leftindent}*\ND@IndentWidth,-#2*\ND@LineSpacing) node[anchor=west] (#2_2) {
␣␣␣␣␣␣\ifND@MathModeFormula\ensuremath{#3}\else#3\fi
␣␣␣␣␣␣};}␣␣␣
␣␣\else
␣␣\draw (\value{leftindent} * \ND@IndentWidth, - #2 * \ND@LineSpacing) node[anchor=west] (#2_2) {
␣␣␣␣\ifND@MathModeFormula\ensuremath{#3}\else #3\fi
␣␣␣␣};
␣␣\fi␣␣
␣␣%Justification node
␣␣\ifND@Overlay
␣␣␣␣\uncover<\ND@oj->{␣␣\draw (\ND@JustPos,-#2*\ND@LineSpacing) node[anchor=west] (#2_3) {#4}; }␣␣␣␣
```

```
␣\else
␣\draw (\ND@JustPos,-#2*\ND@LineSpacing) node[anchor=west] (#2_3) {#4};
␣\fi
␣%Store y coordinate of top and bottom of diagram
␣\storetopbottomcoord{#2}
␣%Keep track of last premise for the premise line
␣\ifcsdef{lastprem}{
␣␣\ifnum\lastprem<#2
␣␣␣\def\lastprem{#2}
␣␣\fi
␣␣}{
␣␣\def\lastprem{#2}
␣␣}
␣%Keys to reset
␣ \def\ND@number{0}
␣ \let\ND@oa\relax
␣ \let\ND@on\relax
␣ \let\ND@of\relax
␣ \let\ND@oj\relax
␣ \ND@nonumberfalse
}


%The comment command
\DeclareRobustCommand{\comment}[3][]{%
␣%Set the keys
␣\let\ND@oc\relax
␣\setkeys{ND}{#1}
␣%Actually draw the comment
␣\ifND@Overlay
␣␣\@ifundefined{ND@oc}{
␣␣\uncover<\ND@oa>{␣␣\node[right of=#2_3,xshift=3mm,anchor=west,text width=\ND@CommentWidth] {#3}; }
␣}{
␣\uncover<\ND@oc>{␣␣\node[right of=#2_3,xshift=3mm,anchor=west,text width=\ND@CommentWidth] {#3}; }␣␣
␣␣}
␣\else
␣\node[right of=#2_3,xshift=3mm,anchor=west,text width=\ND@CommentWidth] {#3};
␣\fi
```

```
}

␣␣
\endinput
%%
%% End of file ‘NaturalDeduction.sty’
```