

prooftrees

Version v0.9 (SVN Rev: 10522)

Clea F. Rees*

2024/10/23

Abstract

prooftrees is a L^AT_EX 2_ε package, based on **forest**, designed to support the typesetting of logical tableaux — ‘proof trees’ or ‘truth trees’ — in styles sometimes used in teaching introductory logic courses, especially those aimed at students without a strong background in mathematics. One textbook which uses proofs of this kind is Hodges (1991). Like **forest**, **prooftrees** supports memoize out-of-the-box.

Note that this package requires version 2.1 (2016/12/04) of forest (Živanović 2016). It will not work with versions prior to 2.1.

I would like to thank Živanović both for developing forest and for considerable patience in answering my questions, addressing my confusions and correcting my mistakes. The many remaining errors are, of course, entirely my own. This package’s deficiencies would be considerably greater and more numerous were it not for his assistance.

*Bug tracker: codeberg.org/cfr/prooftrees/issues | Code: codeberg.org/cfr/prooftrees | Mirror: github.com/cfr42/prooftrees

$S \leftrightarrow \neg T, T \leftrightarrow \neg R \mid_{\mathcal{L}} S \leftrightarrow R$		
1.	$S \leftrightarrow \neg T \checkmark$	pr.
2.	$T \leftrightarrow \neg R \checkmark$	pr.
3.	$\neg(S \leftrightarrow R) \checkmark$	\neg conc.
4.	S	$1 \leftrightarrow E$
5.	$\neg T$	$1 \leftrightarrow E$
6.	T	$2 \leftrightarrow E$
7.	$\neg R$	$2 \leftrightarrow E$
8.	$\neg S$	$3 \neg \leftrightarrow E; 5 \neg \neg E$
9.	R	$3 \neg \leftrightarrow E$
10.	\otimes	$7 \neg \neg E$

$(\exists x)((\forall y)(Py \Rightarrow (x = y)) \cdot Px) \mid_{\mathcal{L}_1} (\exists x)(\forall y)(Py \Leftrightarrow (x = y))$		
1.	$(\exists x)((\forall y)(Py \Rightarrow (x = y)) \cdot Px) \checkmark d$	pr.
2.	$\sim(\exists x)(\forall y)(Py \Leftrightarrow (x = y)) \setminus d$	\neg conc.
3.	$(\forall y)(Py \Rightarrow (d = y)) \cdot Pd \checkmark$	1 $\exists E$
4.	$(\forall y)(Py \Rightarrow (d = y)) \setminus c$	3 $\cdot E$
5.	Pd	3 $\cdot E$
6.	$\sim(\forall y)(Py \Leftrightarrow (d = y)) \checkmark c$	2 $\sim \exists E$
7.	$\sim(Pc \Leftrightarrow (d = c)) \checkmark$	6 $\sim \forall E$
8.	$d \neq c$	7 $\sim \Leftrightarrow E$
9.	$d = c$	7 $\sim \Leftrightarrow E$
10.	$Pc \Rightarrow (d = c) \checkmark$	5, 9 =
11.	\otimes	4 $\forall E$
12.	$d = c$	11 $\Rightarrow E$
13.	\otimes	9, 12 =

Contents

1	Raison d'être	2
2	Assumptions & Limitations	6
3	Typesetting a Proof Tree	6
4	Loading the Package	15
5	Invocation	15
6	Proof Tree Anatomy	15
7	Options	16
7.1	Global Options	16
7.2	Local Options	21
8	Macros	24
9	Memoization	25
10	Compatibility	26
11	Version History	26
11.1	0.9	26
11.2	0.8	26
11.3	0.7	26
11.4	0.6	26
11.5	0.5	26
11.6	0.41	26
11.7	0.4	27
11.8	0.3	27
A	Implementation	29

1 Raison d'être

Suppose that we wish to typeset a typical proof tree demonstrating the following entailment

$$\{P \vee (Q \vee \neg R), P \rightarrow \neg R, Q \rightarrow \neg R\} \vdash \neg R$$

We start by typesetting the tree using `forest`'s default settings (box 1) and find our solution has several advantages: the proof is specified concisely and the code reflects the structure of the tree. It is relatively straightforward to specify a proof using `forest`'s bracket notation, and the spacing of nodes and branches is automatically calculated.

Despite this, the results are not quite what we might have hoped for in a proof tree. The assumptions should certainly be grouped more closely together and no edges (lines) should be drawn between them because these are not steps in the proof — they do not represent inferences. Preferably, edges should start from a common point in the case of branching inferences, rather than there being a gap.

Moreover, proof trees are often compacted so that *non-branching* inferences are grouped together, like assumptions, without explicitly drawn edges. Although explicit edges to represent non-branching inferences are useful when introducing students to proof trees, more complex proofs grow unwieldy and the more compact presentation becomes essential.

Furthermore, it is useful to have the option of *annotating* proof trees by numbering the lines of the proof on the left and entering the justification for each line on the right.

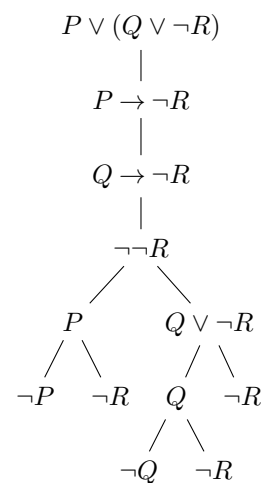
`forest` is a powerful and flexible package capable of all this and, indeed, a good deal more. It is not enormously difficult to customise particular trees to meet most of our desiderata. However, it is difficult to get things perfectly aligned even in simple cases, requires the insertion of ‘phantom’ nodes and management of several sub-trees in parallel (one for line numbers, one for the proof and one for the justifications). The process requires a good deal of manual intervention, trial-and-error and hard-coding of things it would be better to have $\text{\LaTeX 2}_{\varepsilon}$ manage for us, such as keeping count of lines and line references.

`prooftrees` aims to make it as easy to specify proof trees as it was to specify our initial tree using `forest`’s default settings. The package supports a small number of options which can be configured to customise the output. The code for a `prooftrees` proof tree is shown in box 2, together with the output obtained using the default settings.

More extensive configuration can be achieved by utilising `forest` (Živanović 2016) and/or `TikZ` (Tantau 2015) directly. A sample of supported proof tree styles are shown in box 3. The package is *not* intended for the typesetting of proof trees which differ significantly in structure.

1 forest: default settings

```
\begin{forest}
  [$P \vee (Q \vee \neg R)$
    [$P \text{ \texttt{\textbackslash}if \texttt{\textbackslash}not R$}
      [$Q \text{ \texttt{\textbackslash}if \texttt{\textbackslash}not R$}
        [$\neg \neg R$
          [$P$
            [$\neg P$]
            [$\neg R$]
          ]
        ]
      ]
    ]
    [$Q \vee \neg R$
      [$Q$
        [$\neg Q$]
        [$\neg R$]
      ]
    ]
    [$\neg R$]
  ]
]
\end{forest}
```

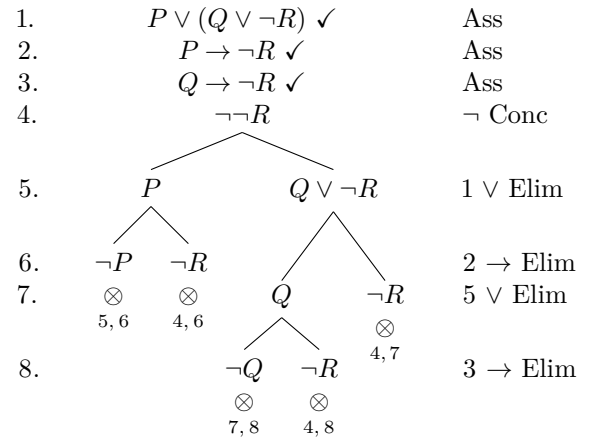


2 prooftrees: default settings

```

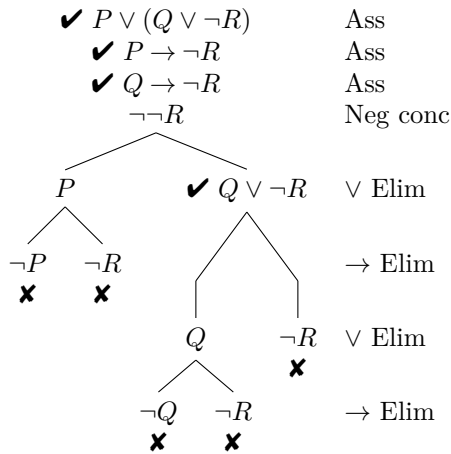
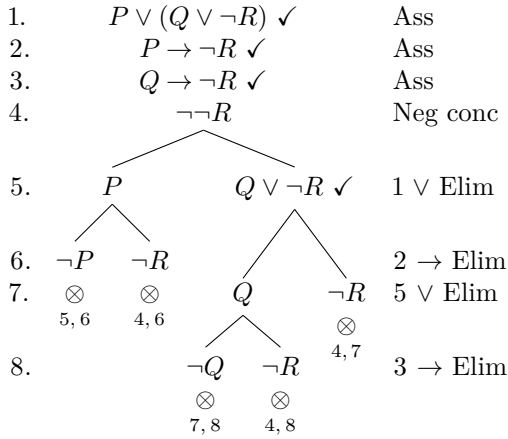
\begin{tableau}
{
  to prove={\{P \vee (Q \vee \lnot R), P \lif
\lnot R, Q \lif \lnot R\} \sststyle{}{} \lnot
R}
}
[P \vee (Q \vee \lnot R), just=Ass, checked
[P \lif \lnot R, just=Ass, checked
[Q \lif \lnot R, just=Ass, checked,
name=last premise
[\lnot \lnot R, just={\$ \lnot \$ Conc},
name=not conc
[P, just={\$ \vee \$ Elim: !uuuu}
[\lnot P, close={: !u, !c}]
[\lnot R, close={: not conc, !c},
just={\$ \lif \$ Elim: !uuuu}]]
[Q \vee \lnot R
[Q, move by=1
[\lnot Q, close={: !u, !c}]
[\lnot R, close={: not conc, !c},
just={\$ \lif \$ Elim: last premise}]]
[\lnot R, close={: not conc, !c},
move by=1, just={\$ \vee \$ Elim: !u}]]]]]]
\end{tableau}

```

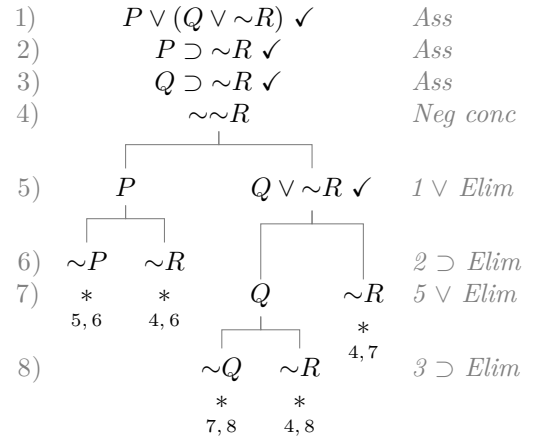
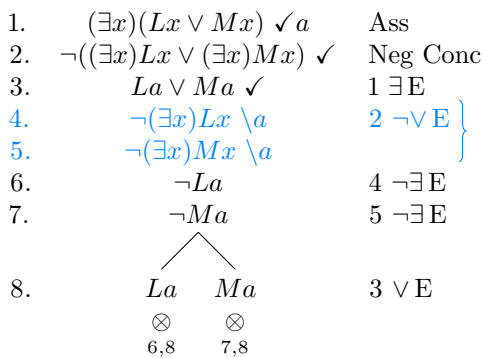
$$\{P \vee (Q \vee \neg R), P \rightarrow \neg R, Q \rightarrow \neg R\} \vdash \neg R$$


3 prooftrees: sample output

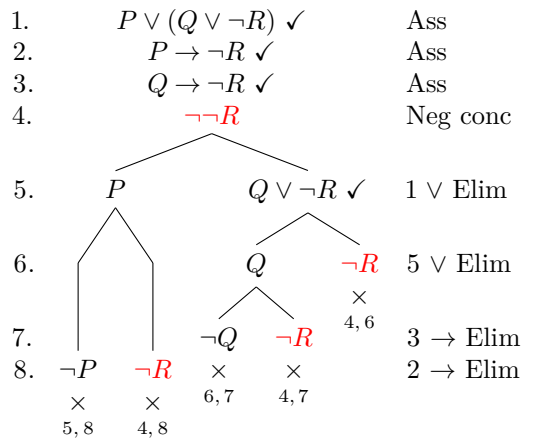
$\{P \vee (Q \vee \neg R), P \rightarrow \neg R, Q \rightarrow \neg R\} \vdash \neg R$



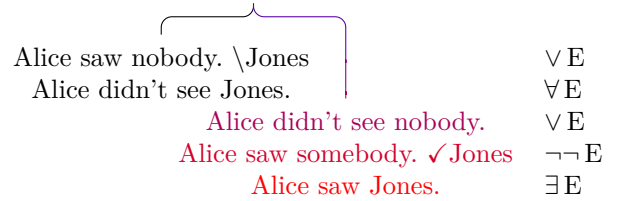
$(\exists x)(Lx \vee Mx) \vdash (\exists x)Lx \vee (\exists x)Mx$



$\{P \vee (Q \vee \neg R), P \rightarrow \neg R, Q \rightarrow \neg R\} \therefore \neg R$



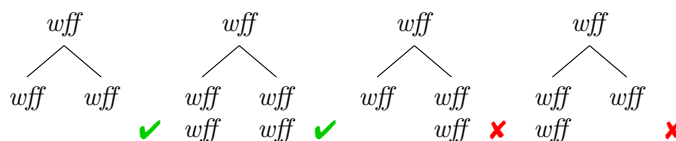
Either Alice saw nobody
or she didn't see nobody.



2 Assumptions & Limitations

`prooftrees` makes certain assumptions about the nature of the proof system, \mathcal{L} , on which proofs are based.

- All derivation rules yield equal numbers of *wff*s on all branches.



If \mathcal{L} fails to satisfy this condition, `prooftrees` is likely to violate the requirements of affected derivation rules by splitting branches ‘mid-inference’.

- No derivation rule yields *wff*s on more than two branches.
- All derivation rules proceed in a downwards direction at an angle of -90° i.e. from north to south.
- Any justifications are set on the far right of the proof tree.
- Any line numbers are set on the far left of the proof tree.
- Justifications can refer only to earlier lines in the proof. `prooftrees` can typeset proofs if \mathcal{L} violates this condition, but the cross-referencing system explained in section 7.2 cannot be used for affected justifications.

`prooftrees` does not support the automatic breaking of proof trees across pages. Proof trees can be manually broken by using `line no shift` with an appropriate value for parts after the first (section 7.1). However, horizontal alignment across page breaks will not be consistent in this case.

In addition, `prooftrees` almost certainly relies on additional assumptions not articulated above and certainly depends on a feature of `forest` which its author classifies as experimental (`do dynamics`).

3 Typesetting a Proof Tree

After loading `prooftrees` in the document preamble:

```
% in document's preamble
\usepackage{prooftrees}
```

the `prooftree` environment is available for typesetting proof trees. This takes an argument used to specify a $\langle tree preamble \rangle$, with the body of the environment consisting of a $\langle tree specification \rangle$ in `forest`’s notation. The $\langle tree preamble \rangle$ can be as simple as an empty argument — `{}` — or much more complex.

Customisation options and further details concerning loading and invocation are explained in section 4, section 5, section 6, section 7 and section 8. In this section, we begin by looking at a simple example using the default settings.

Suppose that we wish to typeset the proof tree for

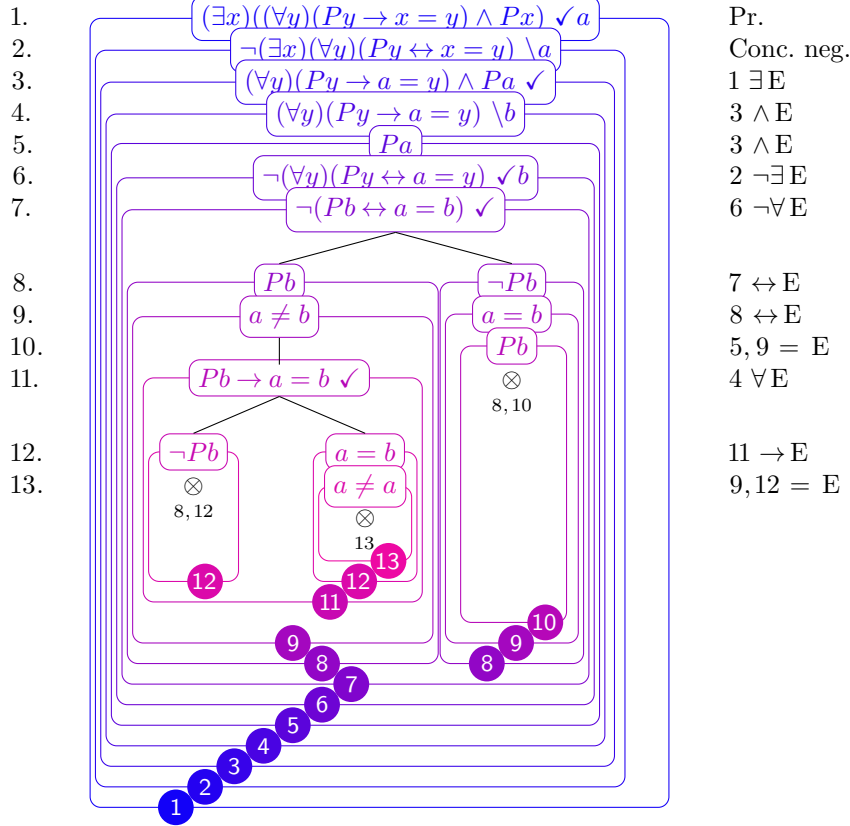
$$(\exists x)((\forall y)(Py \rightarrow x = y) \wedge Px) \vdash (\exists x)(\forall y)(Py \leftrightarrow x = y)$$

and we would like to typeset the entailment established by our proof at the top of the tree. Then we should begin like this:

```
\begin{tableau}
{
  to prove={(\exists x)((\forallall y)(Py \liff x = y) \land Px) \sststile{}{} (\exists x)(\forallall y)(
Py \liff x = y)}
}
\end{tableau}
```

4 Nested structure of proof tree

$$(\exists x)((\forall y)(Py \rightarrow x = y) \wedge Px) \vdash (\exists x)(\forall y)(Py \leftrightarrow x = y)$$



That is all the preamble we want, so we move onto consider the *<tree specification>*. `forest` uses square brackets to specify trees' structures. To typeset a proof, think of it as consisting of nested trees, trunks upwards, and work from the outside in and the trunks down (box 4).

Starting with the outermost tree 12 and the topmost trunk, we replace the with square brackets and enter the first *wff* inside, adding `just=Pr.` for the justification on the right and `checked=a` so that the line will be marked as discharged with *a* substituted for *x*. We also use `forest`'s `name` to label the line for ease of reference later. (Technically, it is the node rather than the line which is named, but, for our purposes, this doesn't matter. `forest` will create a name if we don't specify one, but it will not necessarily be one we would have chosen for ease of use!)

```
\begin{tableau}
{
  to prove={(\exists x)((\forall y)(Py \lif x = y) \land Px) \sststyle{}{}} (\exists x)(\forall y)(
Py \liff x = y)}
}
[{\exists x)((\forall y)(Py \lif x = y) \land Px)}, checked=a, just=Pr., name=pr
]
\end{tableau}
```

We can refer to this line later as `pr.`

We then consider the next tree 12. Its goes inside that for 12, so the square brackets containing the next *wff* go inside those we used for 12. Again, we add the justification with `just`, but we use `subs=a` rather than `checked=a` as we want to mark substitution of *a* for *x* without discharging the line. Again, we use `name` so

that we can refer to the line later as `neg conc`.

```
\begin{tableau}
{
  to prove={{(\exists x)((\forall y)(Py \lif x = y) \land Px) \sststile{}{} (\exists x)(\forall y)(
Py \liff x = y)}}
}
[{{(\exists x)((\forall y)(Py \lif x = y) \land Px)}, checked=a, just=Pr., name=pr
  [{{\lnot (\exists x)(\forall y)(Py \liff x = y)}, subs=a, just=Conc.\neg., name=neg conc
  ]
}
]
\end{tableau}
```

Turning to tree 12, we again note that its \square is nested within the previous two, so the square brackets for its *wff* need to be nested within those for the previous *wff*s. This time, we want to mark the line as discharged without substitution, so we simply use `checked` without a value. Since the justification for this line includes mathematics, we need to ensure that the relevant part of the justification is surrounded by `...$` or `\(...\)`. This justification also refers to an earlier line in the proof. We could write this as `just=1 $\exists\elim$`, but instead we use the name we assigned earlier with the referencing feature provided by `prooftrees`. To do this, we put the reference, `pr` after the rest of the justification, separating the two parts by a colon i.e. `$\exists\elim$:pr` and allow `prooftrees` to figure out the correct number.

```
\begin{tableau}
{
  to prove={{(\exists x)((\forall y)(Py \lif x = y) \land Px) \sststile{}{} (\exists x)(\forall y)(
Py \liff x = y)}}
}
[{{(\exists x)((\forall y)(Py \lif x = y) \land Px)}, checked=a, just=Pr., name=pr
  [{{\lnot (\exists x)(\forall y)(Py \liff x = y)}, subs=a, just=Conc.\neg., name=neg conc
    [{{(\forall y)(Py \lif a = y) \land Pa}, checked, just=$\exists\elim$:pr
    ]
  ]
}
]
\end{tableau}
```

Continuing in the same way, we surround each of the *wff*s for 12, 12, 12 and 12 within square brackets nested within those surrounding the previous *wff* since each of the trees is nested within the previous one. Where necessary, we use `name` to label lines we wish to refer to later, but we also use `forest`'s *relative* naming system when this seems easier. For example, in the next line we add, we specify the justification as `just=$\land\elim$:!u`. `!u` tells `forest` that the reference specifies a relationship between the current line and the referenced one, rather than referring to the other line by name. `!u` refers to the current line's parent line — in this case, `{(\forall y)(Py \lif a = y) \land Pa}`, `checked, just=$\exists\elim$:pr`. `!uu` refers to the current line's parent line's parent line and so on.


```
\begin{tableau}
{
  to prove={{(\exists x)((\forall y)(Py \lif x = y) \land Px) \sststile{}{} (\exists x)(\forall y)(
Py \liff x = y)}}
}
[{{(\exists x)((\forall y)(Py \lif x = y) \land Px)}, checked=a, just=Pr., name=pr
  [{{\lnot (\exists x)(\forall y)(Py \liff x = y)}, subs=a, just=Conc.\neg., name=neg conc
    [{{(\forall y)(Py \lif a = y) \land Pa}, checked, just=$\exists\elim$:pr
      [{{(\forall y)(Py \lif a = y)}, subs=b, just=$\land\elim$:!u, name=mark
        [Pa, just=$\land\elim$:!uu, name=simple
          [{{\lnot (\forall y)(Py \liff a = y)}, checked=b, just=$\lnot\exists\elim$:neg conc
            [{{\lnot (Pb \liff a = b)}, checked, just=$\lnot\forall\elim$:!u
            ]
          ]
        ]
      ]
    ]
  ]
}
]
\end{tableau}
```



```

]
]
]
\end{tableau}


```

Reaching [12](#), things get a little more complex since we now have not one, but *two*  nested within [12](#). This means that we need *two* sets of square brackets for [12](#) — one for each of its two trees. Again, both of these should be nested within the square brackets for [12](#) but neither should be nested within the other because the trees for the two branches at [12](#) are distinct.

```

\begin{tableau}
{
  to prove={(\exists x)((\forall y)(Py \wedge x = y) \wedge Px) \wedge (\exists x)(\forall y)(Py \wedge x = y)}
}
[{\exists x}(\forall y)(Py \wedge x = y) \wedge Px], checked=a, just=Pr., name=pr
[{\neg (\exists x)(\forall y)(Py \wedge x = y)}, subs=a, just=Conc.\neg., name=neg conc
[{\forall y}(Py \wedge a = y) \wedge Pa], checked, just={\exists}\elim$:pr
[{\forall y}(Py \wedge a = y)], subs=b, just={\land}\elim$:!u, name=mark
[Pa, just={\land}\elim$:!uu, name=simple
[{\neg (\forall y)(Py \wedge a = y)}, checked=b, just={\neg}\exists\elim$:neg conc
[{\neg (Pb \wedge a = b)}, checked, just={\neg}\forall\elim$:!u
[Pb, just={\wedge}\elim$:!u, name=to Pb or not to Pb
]
[{\neg Pb
]
]
]
]
]
]
]
\end{tableau}

```

At this point, we need to work separately or in parallel on each of our two branches since each constitutes its own tree. Turning to trees [12](#), each needs to be nested within the relevant tree [12](#), since each  is nested within the applicable branch's tree. Hence, we nest square brackets for each of the *wff*'s at [12](#) within the previous set.

```

\begin{tableau}
{
  to prove={(\exists x)((\forall y)(Py \wedge x = y) \wedge Px) \iff (\exists x)(\forall y)(Py \wedge x = y)}
}
[{\exists x}((\forall y)(Py \wedge x = y) \wedge Px)], checked=a, just=Pr., name=pr
[{\neg (\exists x)(\forall y)(Py \wedge x = y)}], subs=a, just=Conc.\neg., name=neg conc
[{\forall y}(Py \wedge a = y) \wedge Pa], checked, just={\exists\elim$:pr
[{\forall y}(Py \wedge a = y)], subs=b, just={\land\elim$:!u, name=mark
[Pa, just={\land\elim$:!uu, name=simple
[{\neg (\forall y)(Py \wedge a = y)}], checked=b, just={\neg\exists\elim$:neg conc
[{\neg (Pb \wedge a = b)}], checked, just={\neg\forall\elim$:!u
[Pb, just={\liff\elim$:!u, name=to Pb or not to Pb
[a \neg b, just={\liff\elim$:!u
]
]
[{\neg Pb
[{a = b}
]
]

```

```

    ]
  ]
]
\end{tableau}

```

We only have one tree 12 as there is no corresponding tree in the left-hand branch. This isn't a problem: we just need to ensure that we nest it within the appropriate tree 12. There are two additional complications here. The first is that the justification contains a comma, so we need to surround the argument we give just with curly brackets. That is, we must write `just={5,9 $\$=\text{elim}\$$ }` or `just={ $\$=\text{elim}\$$:{simple,!u}}`. The second is that we wish to close this branch with an indication of the line numbers containing inconsistent *wffs*. We can use `close={8,10}` for this or we can use the same referencing system we used to reference lines when specifying justifications and write `close={:to Pb or not to Pb,!c}`. In either case, we again surrounding the argument with curly brackets to protect the comma. !c refers to the current line — something useful in many close annotations, but not helpful in specifying non-circular justifications.

```

\begin{tableau}
{
  to prove={{(\exists x)((\forall y)(Py \text{ \texttt{\textbackslash}liff} x = y) \text{ \texttt{\textbackslash}land} Px) \text{ \texttt{\textbackslash}sststile}{}}{(\exists x)(\forall y)(
Py \text{ \texttt{\textbackslash}liff} x = y)}}
}
[{{(\exists x)((\forall y)(Py \text{ \texttt{\textbackslash}liff} x = y) \text{ \texttt{\textbackslash}land} Px)}, checked=a, just=Pr., name=pr
[{{\text{ \texttt{\textbackslash}not} }(\exists x)(\forall y)(Py \text{ \texttt{\textbackslash}liff} x = y)}, subs=a, just=Conc.\text{ \texttt{\textbackslash}neg.}, name=neg conc
[{{(\forall y)(Py \text{ \texttt{\textbackslash}liff} a = y) \text{ \texttt{\textbackslash}land} Pa}, checked, just= $\$=\text{elim}\$$ :pr
[{{(\forall y)(Py \text{ \texttt{\textbackslash}liff} a = y)}, subs=b, just= $\$=\text{land}\text{ \texttt{\textbackslash}elim}\$$ :!u, name=mark
[Pa, just= $\$=\text{land}\text{ \texttt{\textbackslash}elim}\$$ :!uu, name=simple
[{{\text{ \texttt{\textbackslash}not} }(\forall y)(Py \text{ \texttt{\textbackslash}liff} a = y)}, checked=b, just= $\$=\text{ \texttt{\textbackslash}not} \text{ \texttt{\textbackslash}exists} \text{ \texttt{\textbackslash}elim}\$$ :neg conc
[{{\text{ \texttt{\textbackslash}not} } (Pb \text{ \texttt{\textbackslash}liff} a = b)}, checked, just= $\$=\text{ \texttt{\textbackslash}not} \text{ \texttt{\textbackslash}forall} \text{ \texttt{\textbackslash}elim}\$$ :!u
[Pb, just= $\$=\text{ \texttt{\textbackslash}liff} \text{ \texttt{\textbackslash}elim}\$$ :!u, name=:to Pb or not to Pb
[a \text{ \texttt{\textbackslash}neq} b, just= $\$=\text{ \texttt{\textbackslash}liff} \text{ \texttt{\textbackslash}elim}\$$ :!u
]
]
[{\text{ \texttt{\textbackslash}not} Pb
[ {a = b}
[Pb, just={ $\$=\text{elim}\$$ :{simple,!u}}, close={:to Pb or not to Pb,!c}
]
]
]
]
]
]
]
]
\end{tableau}

```

This completes the main right-hand branch of the tree and we can focus solely on the remaining left-hand one. Tree 12 is straightforward — we just need to nest it within the left-hand tree 12.

```

\begin{tableau}
{
  to prove={{(\exists x)((\forall y)(Py \text{ \texttt{\textbackslash}liff} x = y) \text{ \texttt{\textbackslash}land} Px) \text{ \texttt{\textbackslash}sststile}{}}{(\exists x)(\forall y)(
Py \text{ \texttt{\textbackslash}liff} x = y)}}
}
[{{(\exists x)((\forall y)(Py \text{ \texttt{\textbackslash}liff} x = y) \text{ \texttt{\textbackslash}land} Px)}, checked=a, just=Pr., name=pr

```

```
[{ \lnot (\exists x)(\forall y) (Py \liff x = y)}, subs=a, just=Conc.\neg., name=neg conc  
[{\forall y}(Py \lif a = y) \land Pa}, checked, just=\exists\elim$:pr  
[{\forall y}(Py \lif a = y)}, subs=b, just=\land\elim$:!u, name=mark  
[Pa, just=\land\elim$:!uu, name=simple  
[{\lnot (\forall y)(Py \liff a = y)}, checked=b, just=\lnot\exists\elim$:neg conc  
[{\lnot (Pb \liff a = b)}, checked, just=\lnot\forall\elim$:!u  
[Pb, just=\liff\elim$:!u, name=to Pb or not to Pb  
[a \neq b, just=\liff\elim$:!u  
[{Pb \lif a = b}], checked, just=\forall\elim$:mark%, move by=1  
]  
]  
]  
[\lnot Pb  
[{a = b}  
[Pb, just={=\elim$:{simple,!u}}, close={:to Pb or not to Pb,!c}  
]  
]  
]  
]  
]  
]  
]  
]  
]  
]  
]\end{tableau}
```

At this point, the main left-hand branch itself branches, so we have two trees (12). Treating this in the same way as the earlier branch at (12), we use two sets of square brackets nested within those for tree (12), but with neither nested within the other. Since we also want to mark the leftmost branch as closed, we add `close={:to Pb or not to Pb,!c}` in the same way as before.

```

\begin{tableau}
{
  to prove={(\exists x)((\forall y)(Py \wedge x = y) \wedge Px) \wedge (\exists x)(\forall y)(Py \wedge x = y)}
}
[{\exists x)((\forall y)(Py \wedge x = y) \wedge Px)}, checked=a, just=Pr., name=pr
[{\neg (\exists x)(\forall y)(Py \wedge x = y)}, subs=a, just=Conc.\neg., name=neg conc
[{\forall y)(Py \wedge a = y) \wedge Pa}, checked, just=${\exists}\elim$:pr
[{\forall y)(Py \wedge a = y)}, subs=b, just=${\wedge}\elim$:!u, name=mark
[Pa, just=${\wedge}\elim$:!uu, name=simple
[{\neg (\forall y)(Py \wedge a = y)}, checked=b, just=${\neg\exists}\elim$:neg conc
[{\neg (Pb \wedge a = b)}, checked, just=${\neg\forall}\elim$:!u
[Pb, just=${\wedge}\elim$:!u, name=to Pb or not to Pb
[a \neq b, just=${\wedge}\elim$:!u
[{\neg Pb \wedge a = b}, checked, just=4 ${\forall}\elim$:
[{\neg Pb, close={:to Pb or not to Pb,!c}, just=${\wedge}\elim$:!u
]
[{a = b}
]
]
]
[{\neg Pb
[{a = b}
[Pb, just={${\elim$:simple,!u}}, close={:to Pb or not to Pb,!c}
]
]
]
]
]

```

```
\end{tableau}
```

We complete our initial specification by nesting (12) within the appropriate tree (12), again marking closure appropriately.

```
\begin{tableau}
{
to prove={(\exists x)((\forall y)(Py \lif x = y) \land Px)} (\exists x)(\forall y)(
Py \liff x = y)}
}{
[(\exists x)((\forall y)(Py \lif x = y) \land Px)], checked=a, just=Pr., name=pr
[{\not (\exists x)(\forall y)(Py \liff x = y)}, subs=a, just=Conc.\neg., name=neg conc
[{(\forall y)(Py \lif a = y) \land Pa}], checked, just=${\exists}\elim$:pr
[{(\forall y)(Py \lif a = y)}, subs=b, just=${\land}\elim$:\!u, name=mark
[P_a, just=${\land}\elim$:\!uu, name=simple
[{{\not (\forall y)(Py \liff a = y)}, checked=b, just=${\not {\exists}}\elim$\neg conc
[{{\not (P_b \liff a = b)}, checked, just=${\not {\forall}}\elim$:\!u
[P_b, just=${\liff}\elim$:\!u, name=to P_b or not to P_b
[a {\neq} b, just=${\liff}\elim$:\!u
[{{P_b \lif a = b}, checked, just=4 ${\forall}\elim$
[{\not P_b, close={:to P_b or not to P_b,!c}, just=${\lif}\elim$!\!u
]
[{{a = b}
[a {\neq} a, close={:!c}, just={{=\elim$:{!uu,\!u}}}
]
]
]
]
]
]
[\not P_b
[{{a = b}
[P_b, just={{=\elim${simple,\!u}}, close={:to P_b or not to P_b,!c}
]
]
]
]
]
]
]
]\end{tableau}
```

Compiling our code, we find that the line numbering is not quite right:

	$(\exists x)((\forall y)(Py \rightarrow x = y) \wedge Px) \vdash (\exists x)(\forall y)(Py \leftrightarrow x = y)$	
1.	$(\exists x)((\forall y)(Py \rightarrow x = y) \wedge Px) \checkmark a$	Pr.
2.	$\neg(\exists x)(\forall y)(Py \leftrightarrow x = y) \setminus a$	Conc. neg.
3.	$(\forall y)(Py \rightarrow a = y) \wedge Pa \checkmark$	1 $\exists E$
4.	$(\forall y)(Py \rightarrow a = y) \setminus b$	3 $\wedge E$
5.	Pa	3 $\wedge E$
6.	$\neg(\forall y)(Py \leftrightarrow a = y) \checkmark b$	2 $\neg \exists E$
7.	$\neg(Pb \leftrightarrow a = b) \checkmark$	6 $\neg \forall E$
	$\begin{array}{cc} & \swarrow \quad \searrow \\ Pb & \neg Pb \end{array}$	7 $\leftrightarrow E$
8.	$a \neq b$	8 $\leftrightarrow E$
9.	$a = b$	4 $\forall E$; 5, 9 = E
10.	$\begin{array}{cc} Pb \rightarrow a = b \checkmark & Pb \\ & \otimes \\ & 8, 10 \end{array}$	
11.	$\begin{array}{cc} \neg Pb & a = b \\ & \otimes \\ & 8, 11 \end{array}$	10 $\rightarrow E$
12.	$\begin{array}{cc} \otimes & a \neq a \\ 8, 11 & \\ & \otimes \\ & 12 \end{array}$	9, 11 = E

prooftrees warns us about this:

Package prooftrees Warning: Merging conflicting justifications for line 10! Please examine the output carefully and use "move by" to move lines later in the proof if required. Details of how to do this are included in the documentation.

We would like line 10 in the left-hand branch to be moved down by one line, so we add `move by=1` to the relevant line of our proof. That is, we replace the line

$$[\{Pb \mid a = b\}, \text{checked}, \text{just}=4 \text{ } \$\text{forall}\backslash\text{elim}\$$$

by

$$[\{Pb \mid a = b\}, \text{checked}, \text{just}=\$ \text{forall} \backslash \text{elim} \$: \text{mark}, \text{move by}=1$$

giving us the following code:

```

\begin{tableau}
{
  to prove={(\exists x)((\forall y)(Py \wedge x = y) \wedge Px) \text{sststle}}{(\exists x)(\forall y)(
Py \wedge x = y)}
}
[{\exists x)((\forall y)(Py \wedge x = y) \wedge Px)}, checked=a, just=Pr., name=pr
[{\neg (\exists x)(\forall y)(Py \wedge x = y)}, subs=a, just=Conc.\neg., name=neg conc
[{\forall y)(Py \wedge a = y) \wedge Pa}, checked, just=${\exists}\elim$:pr
[{\forall y)(Py \wedge a = y)}, subs=b, just=${\wedge}\elim$:!u, name=mark
[Pa, just=${\wedge}\elim$:!uu, name=simple
[{\neg (\forall y)(Py \wedge a = y)}, checked=b, just=${\neg\exists}\elim$:neg conc
[{\neg (Pb \wedge a = b)}, checked, just=${\neg\forall}\elim$:!u
[Pb, just=${\wedge}\elim$:!u, name=to Pb or not to Pb
[a \neg b, just=${\wedge}\elim$:!u
[{\wedge a = b}, checked, just=${\forall}\elim$:mark, move by=1
[{\neg Pb, close={:to Pb or not to Pb,!c}, just=${\wedge}\elim$:!u
]
[{a = b}
[a \neg a, close={:!c}, just={${}\elim$:!uuu,!u}}
]
]
]
]
]

```

```

[\lnot Pb
  [{a = b}
    [Pb, just={\$=\elim$:{simple,!u}}, close={:to Pb or not to Pb,!c}
    ]
  ]
]
]
]
]
]
]
]
]
]
]
\end{tableau}

```

which produces our desired result:

	$(\exists x)((\forall y)(Py \rightarrow x = y) \wedge Px) \vdash (\exists x)(\forall y)(Py \leftrightarrow x = y)$	
1.	$(\exists x)((\forall y)(Py \rightarrow x = y) \wedge Px) \checkmark a$	Pr.
2.	$\neg(\exists x)(\forall y)(Py \leftrightarrow x = y) \setminus a$	Conc. neg.
3.	$(\forall y)(Py \rightarrow a = y) \wedge Pa \checkmark$	1 \exists E
4.	$(\forall y)(Py \rightarrow a = y) \setminus b$	3 \wedge E
5.	Pa	3 \wedge E
6.	$\neg(\forall y)(Py \leftrightarrow a = y) \checkmark b$	2 $\neg\exists$ E
7.	$\neg(Pb \leftrightarrow a = b) \checkmark$	6 $\neg\forall$ E
8.	Pb	7 \leftrightarrow E
9.	$a \neq b$	8 \leftrightarrow E
10.	Pb	5, 9 = E
11.	$Pb \rightarrow a = b \checkmark$	4 \forall E
	\otimes 8, 10	
12.	$\neg Pb$	11 \rightarrow E
13.	$a = b$	9, 12 = E
	\otimes 8, 12	
	$a \neq a$	
	\otimes 13	

4 Loading the Package

To load the package simply add the following to your document's preamble.

```
\usepackage{prooftrees}
```

`prooftrees` will load `forest` automatically.

The only option currently supported is `tableaux`. If this option is specified, the `prooftree` environment will be called `tableau` instead.

Example: `\usepackage[tableaux]prooftrees`

would cause the `tableau` environment to be defined *rather than* `prooftree`.

Any other options given will be passed to `forest`.

Example: `\usepackage[debug]prooftrees`

would enable `forest`'s debugging.

If one or more of `forest`'s libraries are to be loaded, it is recommended that these be loaded separately and their defaults applied, if applicable, within a local \TeX group so that they do not interfere with `prooftrees`'s environment.

5 Invocation

`prooftree`
environment

```
\begin{prooftree}{\langle tree preamble \rangle}\langle tree specification \rangle\end{prooftree}
```

The $\langle tree preamble \rangle$ is used to specify any non-default options which should be applied to the tree. It may contain any code valid in the preamble of a regular `forest` tree, in addition to setting `prooftree` options. The preamble may be empty, but the argument is *required*¹. The $\langle tree specification \rangle$ specifies the tree in the bracket notation parsed by `forest`.

Users of `forest` should note that the environments `prooftree` and `forest` differ in important ways.

- *`prooftree`'s argument is mandatory.*
- *The tree's preamble cannot be given in the body of the environment.*
- *`\end{prooftree}` must follow the $\langle tree specification \rangle$ immediately.*

`tableau`
environment

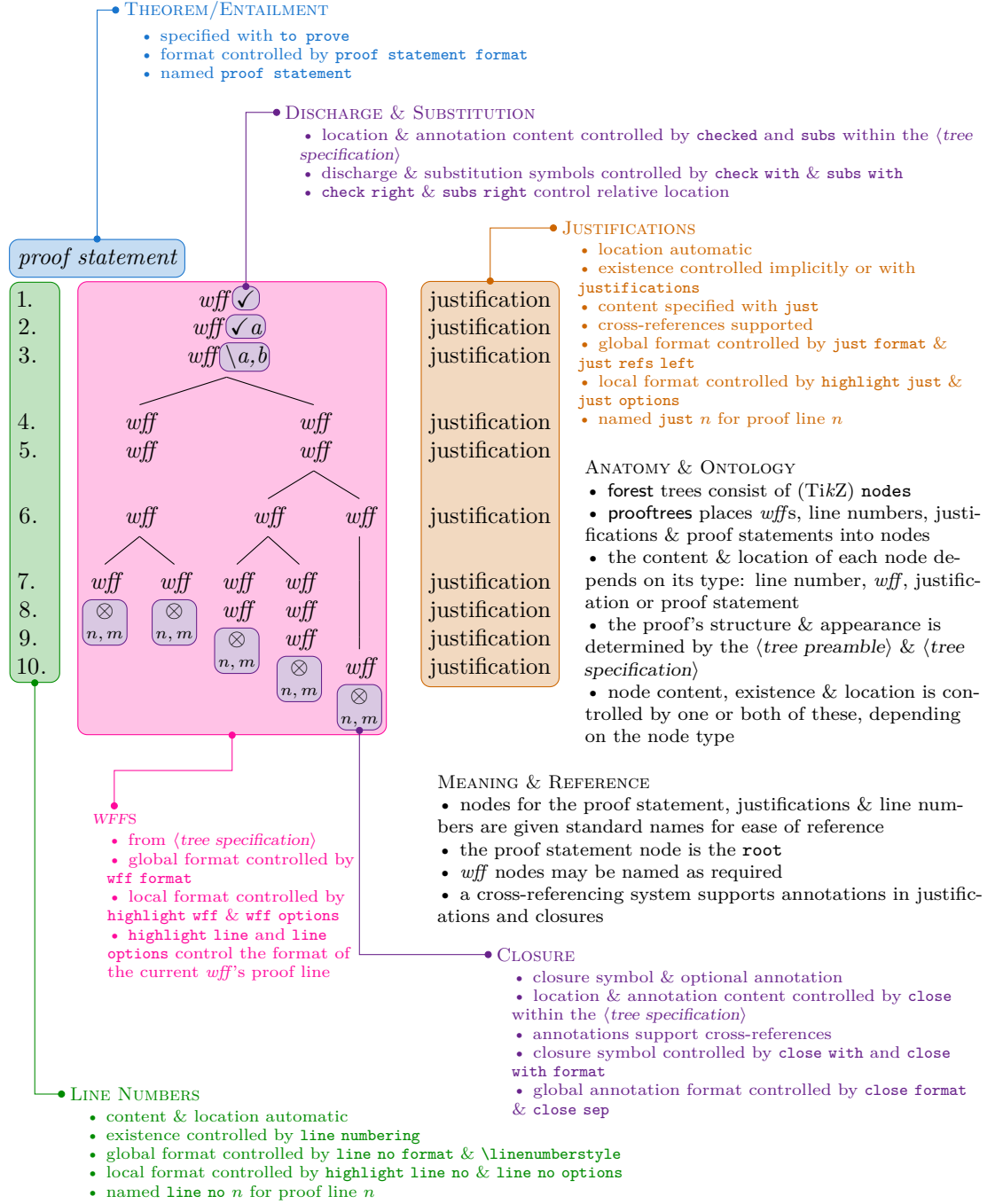
```
\begin{tableau}{\langle tree preamble \rangle}\langle tree specification \rangle\end{tableau}
```

A substitute for `prooftree`, defined *instead* of `prooftree` if the package option `tableaux` is specified or a `\prooftree` macro is already defined when `prooftrees` is loaded. See section 4 for details and section 10 for this option's *raison d'être*.

6 Proof Tree Anatomy

The following diagram provides an overview of the configuration and anatomy of a `prooftrees` proof tree. Detailed documentation is provided in section 7 and section 8.

¹Failure to specify a required argument does not always yield a compilation error in the case of environments. However, failure to specify required arguments to environments often fails to achieve the best consequences, even when it does not result in compilation failures, and will, therefore, be avoided by the prudent.



7 Options

Most configuration uses the standard key/value interface provided by TikZ and extended by forest. These are divided into those which determine the overall appearance of the proof as a whole and those with more local effects.

7.1 Global Options

The following options affect the global style of the tree and should typically be set in the tree's preamble if non-default values are desired. The default values for the document can be set outside the `prooftree` environment using `\forestset{<settings>}`. If *only* proof trees will be typeset, a default style can be configured using forest's default preamble.

`auto move` = true|false
`not auto move`
Forest boolean register

Default: true

Determines whether `prooftrees` will move lines automatically, where possible, to avoid combining different justifications when different branches are treated differently. The default is to avoid conflicts automatically where possible. Turning this off permits finer-grained control of what gets moved using `move by`. The following are equivalent to the default setting:

```
auto move
auto move=true
```

Either of the following will turn auto move off:

```
not auto move
auto move=false
```

`line numbering` = true|false
`not line numbering`
Forest boolean register

Default: true

This determines whether lines should be numbered. The default is to number lines. The following are equivalent to the default setting:

```
line numbering
line numbering=true
```

Either of the following will turn line numbering off:

```
not line numbering
line numbering=false
```

`justifications` = true|false
`not justifications`
Forest boolean register

This determines whether justifications for lines of the proof should be typeset to the right of the tree. It is rarely necessary to set this option explicitly as it will be automatically enabled if required. The only exception concerns a proof for which a line should be moved but no justifications are specified. In this case either of the following should be used to activate the option:

```
justifications
justifications=true
```

This is not necessary if `just` is used for any line of the proof.

`single branches` = true|false
`not single branches`
Forest boolean register

Default: false

This determines whether inference steps which do not result in at least two branches should draw and explicit branch. The default is to not draw single branches explicitly. The following are equivalent to the default setting:

```
not single branches
single branches=false
```

Either of the following will turn line numbering off:

```
single branches
single branches=true
```

`line no width` = $\langle dimension \rangle$

Forest dimension register

The maximum width of line numbers. By default, this is set to the width of the formatted line number 99.

Example: `line no width=20pt`

`just sep` = $\langle dimension \rangle$

Forest dimension register

Default: 1.5em

Amount by which to shift justifications away from the tree. A larger value will shift the justifications further to the right, increasing their distance from the tree, while a smaller one will decrease this distance. Note that a negative value ought never be given. Although this will not cause an error, it may result in strange things happening. If you wish to decrease the distance between the tree and the justifications further, please set `just sep` to zero and use the options provided by `forest` and/or `TikZ` to make further negative adjustments.

Example: `just sep=.5em`

`line no sep` = $\langle dimension \rangle$

Forest dimension register

Default: 1.5em

Amount by which to shift line numbers away from the tree. A larger value will shift the line numbers further to the left, increasing their distance from the tree, while a smaller one will decrease this distance. Note that a negative value ought never be given. Although this will not cause an error, it may result in strange things happening. If you wish to decrease the distance between the tree and the line numbers further, please set `line no sep` to zero and use the options provided by `forest` and/or `TikZ` to make further negative adjustments.

Example: `line no sep=5pt`

`close sep` = $\langle dimension \rangle$

Forest dimension register

Default: `.75\baselineskip`

Distance between the symbol marking branch closure and any following annotation. If the format of such annotations is changed with `close format`, this dimension may require adjustment.

Example: `close sep=\baselineskip`

`proof tree inner proof width` = $\langle dimension \rangle$

Forest dimension register

Default: 0pt

`proof tree inner proof midpoint` = $\langle dimension \rangle$

Forest dimension register

Default: 0pt

`line no shift` = $\langle integer \rangle$

Forest count register

Default: 0

This value increments or decrements the number used for the first line of the proof. By default, line numbering starts at 1.

Example: `line no shift=3`

would begin numbering the lines at 4.

`zero start`

Forest style

Start line numbering from 0 rather than 1. The following are equivalent:

```
zero start
line no shift=-1
```

`to prove` = $\langle wff \rangle$

Forest style

Statement of theorem or entailment to be typeset above the proof. In many cases, it will be necessary to enclose the statement in curly brackets.

Example: `to prove={\sststyle{}} P \lif P`

By default, the content is expected to be suitable for typesetting in maths mode and should *not*, therefore, be enclosed by dollar signs or equivalent.

`check with` = $\langle symbol \rangle$

Forest toks register

Default: `\ensuremath{\checkmark}` (✓)

Symbol with which to mark discharged lines.

Example: `check with={\text{\ding{52}}}`

Within the tree, `checked` is used to identify discharged lines.

`check right` = `true|false`

`not check right`

Forest boolean register

Default: `true`

Determines whether the symbol indicating that a line is discharged should be placed to the right of the *wff*. The alternative is, unsurprisingly, to place it to the left of the *wff*. The following are equivalent to the default setting:

```
check right
check right=true
```

`check left`

Forest style

Set `check right=false`. The following are equivalent ways to place the markers to the left:

```
check right=false
not check right
check left
```

`close with` = $\langle symbol \rangle$

Forest toks register

Default: `\ensuremath{\otimes}` (⊗)

Symbol with which to close branches.

Example: `close with={\ensuremath{\ast}}`

Within the tree, `close` is used to identify closed branches.

`close with format` = $\langle key\text{-}value\ list \rangle$

Forest keylist register

Additional TikZ keys to apply to the closure symbol. Empty by default.

Example: `close with format={red, font=}`

To replace a previously set value, rather than adding to it, use `close with format'` rather than `close with format`.

`close format` = $\langle key\text{-}value\ list \rangle$

Forest keylist register

Default: `font=\scriptsize`

Additional TikZ keys to apply to any annotation following closure of a branch.

Example: `close format={font=\footnotesize\sffamily, text=gray!75}`

To replace the default value of `close format`, rather than adding to it, use `close format'` rather than `close format`.

Example: `close format'={text=red}`

will produce red annotations in the default font size, whereas

Example: `close format={text=red}`

will produce red annotations in `\scriptsize`.

`subs with`
Forest toks register

= $\langle symbol \rangle$

Default: `\ensuremath{\backslash}` (`\`)

Symbol to indicate variable substitution.

Example: `\text{:}`

Within the tree, `subs` is used to indicate variable substitution.

`subs right`
`not subs right`
Forest boolean register

= `true|false`

Default: `true`

Determines whether variable substitution should be indicated to the right of the *wff*. The alternative is, again, to place it to the left of the *wff*. The following are equivalent to the default setting:

```
subs right
subs right=true
```

`subs left`
Forest style

Set `subs right=false`. The following are equivalent ways to place the annotations to the left:

```
subs right=false
not subs right
subs left
```

`just refs left`
`not just refs left`
Forest boolean register

= `true|false`

Default: `true`

Determines whether line number references should be placed to the left of justifications. The alternative is to place them to the right of justifications. The following are equivalent to the default setting:

```
just refs left
just refs left=true
```

`just refs right`
Forest style

Set `just refs left=false`. The following are equivalent ways to place the references to the right:

```
just refs left=false
not just refs left
just refs right
```

Note that this setting *only affects the placement of line numbers specified using the cross-referencing system* explained in section 7.2. Hard-coded line numbers in justifications will be typeset as is.

`just format`
Forest keylist register

= $\langle key-value list \rangle$

Additional TikZ keys to apply to line justifications. Empty by default.

Example: `just format={red, font=}`

To replace a previously set value, rather than adding to it, use `just format'` rather than `just format`.

line no format = $\langle \text{key-value list} \rangle$
Forest keylist register

Additional TikZ keys to apply to line numbers. Empty by default.

Example: `line no format={align=right, text=gray}`

To replace a previously set value, rather than adding to it, use `line no format'` rather than `line no format`. To change the way the number itself is formatted — to eliminate the dot, for example, or to put the number in brackets — redefine `\linenumberstyle` (see section 8).

wff format = $\langle \text{key-value list} \rangle$
Forest keylist register

Additional TikZ keys to apply to *wff*s. Empty by default.

Example: `wff format={draw=orange}`

To replace a previously set value, rather than adding to it, use `wff format'` rather than `wff format`.

proof statement format = $\langle \text{key-value list} \rangle$
Forest keylist register

Additional TikZ keys to apply to the proof statement. Empty by default.

Example: `proof statement format={text=gray, draw=gray}`

To replace a previously set value, rather than adding to it, use `proof statement format'` rather than `proof statement format`.

highlight format = $\langle \text{key-value list} \rangle$
Forest autowrapped toks register

Default: `draw=gray, rounded corners`

Additional TikZ keys to apply to highlighted *wff*s.

Example: `highlight format={text=red}`

To apply highlighting, use the `highlight wff`, `highlight just`, `highlight line no` and/or `highlight line` keys (see section 7.2).

merge delimiter = $\langle \text{punctuation} \rangle$
Forest toks register

Default: `\text{;; } (;)`

Punctuation to separate distinct justifications for a single proof line. Note that `prooftrees` will issue a warning if it detects different justifications for a single proof line and will suggest using `move by` to avoid the need for merging justifications. In general, justifications ought not be merged because it is then less clear to which *wff*(s) each justification applies. Moreover, later references to the proof line will be similarly ambiguous. That is, `merge delimiter` ought almost never be necessary because it is almost always better to restructure the proof to avoid ambiguity.

7.2 Local Options

The following options affect the local structure or appearance of the tree and should typically be passed as options to the relevant node(s) within the tree.

grouped
not grouped
Forest boolean option

Indicate that a line is not an inference. When `single branches` is false, as it is with the default settings, this key is applied automatically and need not be given in the specification of the tree. When `single branches` is true, however, this key must be specified for any line which ought not be treated as an inference.

Example: `grouped`

checked
Forest style

Mark a complex *wff* as resolved, discharging the line.

Example: `checked`

checked
Forest style

= $\langle \text{name} \rangle$

Example: `just=From:bertha`

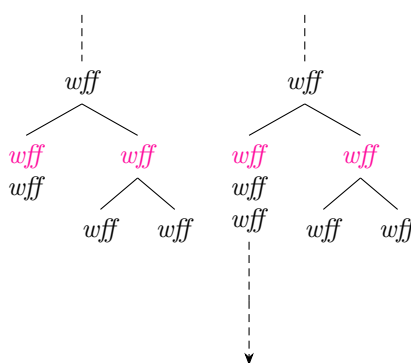
`move by`
Forest style

= $\langle \text{positive integer} \rangle$

Move the content of the current line $\langle \text{positive integer} \rangle$ lines later in the proof. If the current line has a justification and the content is moved, the justification will be moved with the line. Later lines in the same branch will be moved appropriately, along with their justifications.

Example: `move by=3`

Note that, in many cases, **prooftrees** will automatically move lines later in the proof. It does this when it detects a condition in which it expects conflicting justifications may be required for a line while initially parsing the tree. Essentially, **prooftrees** tries to detect cases in which a branch is followed closely by asymmetry in the structure of the branches. This happens, for example, when the first branch's first *wff* is followed by a single *wff*, while the second branch's first *wff* is followed by another branch. Diagrammatically:



In this case, **prooftrees** tries to adjust the tree by moving lines appropriately if required.

However, this detection is merely structural — **prooftrees** does not examine the content of the *wff*s or justifications for this purpose. Nor does it look for slightly more distant structural asymmetries, conflicting justifications in the absence of structural asymmetry or potential conflicts with justifications for lines in other, more distant parallel branches. Although it is not that difficult to detect the *need* to move lines in a greater proportion of cases, the problem lies in providing general rules for deciding *how* to resolve such conflicts. (Indeed, some such conflicts might be better left unresolved e.g. to fit a proof on a single Beamer slide.) In these cases, a human must tell **prooftrees** if something should be moved, what should be moved and how far it should be moved.

Because simple cases are automatically detected, it is best to typeset the proof before deciding whether or where to use this option since **prooftrees** will assume that this option specifies movements which are required *in addition to* those it automatically detects. Attempting to move a line ‘too far’ is not advisable. **prooftrees** tries to simply ignore such instructions, but the results are likely to be unpredictable.

Not moving a line far enough — or failing to move a line at all — may result in the content of one justification being combined with that of another. This happens if `just` is specified more than once for the same proof line with differing content. **prooftrees** *does* examine the content of justifications for *this* purpose. When conflicting justifications are detected for the same proof line, the justifications are merged and a warning issued suggesting the use of `move by`.

`highlight wff`
`not highlight wff`
Forest boolean option

Highlight *wff*.

Example: `highlight wff`

`highlight just`
`not highlight just`
Forest boolean option

Highlight justification.

Example: `highlight just`

Highlight line number.

`highlight line no`
`not highlight line no`
Forest boolean option

Example: `highlight line no`

Highlight proof line.

`highlight line`
`not highlight line`
Forest boolean option

Example: `highlight line`

= \langle key-value list \rangle

Additional TikZ keys to apply to the line number for this line.

Example: `line no options={blue}`

= \langle key-value list \rangle

Additional TikZ keys to apply to the justification for this line.

Example: `just options={draw, font=\bfseries}`

= \langle key-value list \rangle

Additional TikZ keys to apply to the *wff* for this line.

Example: `wff options={magenta, draw}`

Note that this key is provided primarily for symmetry as it is faster to simply give the options directly to `forest` to pass on to TikZ. Unless `wff format` is set to a non-default value, the following are equivalent:

```
wff options={magenta, draw}
magenta, draw
```

= \langle key-value list \rangle

Additional TikZ keys to apply to this proof line.

Example: `line options={draw, rounded corners}`

= \langle text \rangle

Substitute \langle text \rangle for the programmatically-assigned line number. \langle text \rangle will be wrapped by `\linenumberstyle`, so should not be anything which would not make sense in that context.

Example: `line no override={n}`

Do not typeset a line number for this line. Intended for use in trees where `line numbering` is activated, but some particular line should not have its number typeset. Note that the number for the line is still assigned and the node which would otherwise contain that number is still typeset. If the next line is automatically numbered, the line numbering will, therefore, ‘jump’, skipping the omitted number.

Example: `no line no`

8 Macros

$\{\langle$ number $\rangle\}$

`\linenumberstyle`
macro

This macro is responsible for formatting the line numbers. The default definition is

```
\newcommand*\linenumberstyle[1]{\#1.}
```

It may be redefined with `\renewcommand*` in the usual way. For example, if for some reason you would like bold line numbers, try


```
\renewcommand*\linenumberstyle[1]{\textbf{#1.}}
```

9 Memoization

Tableaux created by `prooftrees` cannot, in general, be externalised with `TikZ`'s `external` library. Since `pgf/TikZ`, in general, and `prooftrees`, in particular, can be rather slow to compile, this is a serious issue. If you only have a two or three small tableaux, the compilation time will be negligible. But if you have large, complex proofs or many smaller ones, compilation time will quickly become excessive.

Version 0.9 does not cure the disease, but it does offer an extremely effective remedy for the condition. While it does not make `prooftrees` any faster, it supports the `memoize` package developed by `forest`'s author, Sašo Živanović (2023). Memoization is faster, more secure, more robust and easier to use than `TikZ`'s externalisation.

It is faster. It does not require separate compilations for each memoized object, so it is comparatively fast even when memoizing.

It is more secure. It requires only restricted shell-escape, which almost all \TeX installations enable by default, so it is considerably more secure and can be utilised even where shell-escape is disabled.

It is more robust. It can successfully memoize code which defeats all ordinary mortals' attempts to externalize with the older `TikZ` library.

It is easier to use. It requires less configuration and less intervention. For example, it detects problematic code and aborts memoization automatically in many cases in which `TikZ`'s `external` would either cause a compilation error or silently produce nonsense output, forcing the user to manually disable the process for relevant code.

There is always a 'but', but this is a pretty small 'but' as 'but's go.

But installation requires slightly more work. To reap the full benefits, you want to use either the `perl` or the `python` 'extraction' method. There is a third method, which does not require any special installation, but this lacks several of the advantages explained above and is not recommended.

If you use \TeX Live, you have `perl` already, but you may need to install a couple of libraries. `python` is not a prerequisite for \TeX Live but, if you happen to have it installed, you will probably only need an additional library to use this method.

See *Memoize* (Živanović 2023) for further details.

Once you have the prerequisites setup, all you need do is load `memoize` *before* `prooftrees`.

```
\usepackage[extraction method=perl]{memoize}% or python
\usepackage{prooftrees}
```

After a single compilation, your document will have expanded to include extra pages. At this point, it will look pretty weird. After the next compilation, your document will return to its normal self, the only difference being the speed with which it does so as all your memoized tableaux will simply be included, as opposed to recompiled. Only when you alter the code for a tableau, delete the generated files, disable memoization or explicitly request it will the proof be recompiled.

Memoization is compatible with both `prooftrees`'s cross-referencing system and $\text{\LaTeX} 2_{\epsilon}$'s cross-references, but the latter require an additional compilation. In general, if a document element takes n compilations to stabilise, it will take $n + 1$ compilations to complete the memoization process. See *Memoize* (Živanović 2023) for details.

10 Compatibility

Versions of `prooftrees` prior to 0.5 are incompatible with `bussproofs`, which also defines a `prooftree` environment. Version 0.6 is compatible with `bussproofs` provided

either `bussproofs` is loaded *before* `prooftrees`

or `prooftrees` is loaded with option `tableaux` (see section 4).

In either case, `prooftrees` will *not* define a `prooftree` environment, but will instead define `tableau`. This allows you to use `tableau` for `prooftrees` trees and `prooftree` for `bussproofs` trees.

11 Version History

11.1 0.9

Add support for `memoize` and utilise for documentation.

Use `\NewDocumentEnvironment`, removing direct dependency on `environ`.

11.2 0.8

Add previously unnoticed dependency on `amstext`. Attempt to fix straying closure symbols evident in documentation and a T_EX SE question²

Documentation now loads `enumitem`, since it depended on it already anyway and specifies `doc2` in options for `ltxdoc` as the code is incompatible with the current version.

11.3 0.7

Implement `auto move`. See section 7.1. The main point of this option is to allow automatic moves to be switched off if one teaches students to first apply all available non-branching rules for the tableau as a whole, as opposed to all non-branching rules for the sub-tree. The automatic algorithm is consistent with the latter, but not former, approach. The algorithm favours compact trees, which are more likely to fit on `beamer` slides. Switching the algorithm off permits users to specify exactly how things should or should not be moved. Thanks to Peter Smith for prompting this.

Fix bug reported at tex.stackexchange.com/q/479263/39222.

11.4 0.6

Add compatibility option for use with `bussproofs`. See section 4. Thanks to Peter Smith for suggesting this.

11.5 0.5

Significant re-implementation leveraging the new argument processing facilities in `forest` 2.1. This significantly improves performance as the code is executed much faster than the previous `pgfmath` implementation.

11.6 0.41

Update for compatibility with `forest` 2.1.

²<https://tex.stackexchange.com/q/619314/>.

11.7 0.4

Bug fix release:

- `line no shift` was broken;
- in some cases, an edge was drawn where no edge belonged.

11.8 0.3

First CTAN release.

References

- Hodges, Wilfred (1991). *Logic: An Introduction to Elementary Logic*. Penguin.
- Tantau, Till (2015). *The TikZ and PGF Packages. Manual for Version 3.0.1a*. 3.0.1a. 29th Aug. 2015. URL: <http://sourceforge.net/projects/pgf>.
- Živanović, Sašo (2016). *Forest: A PGF/TikZ-Based Package for Drawing Linguistic Trees*. 2.0.2. 4th Mar. 2016. URL: <http://spj.ff.uni-lj.si/zivanovic/>.
- (2023). *Memoize*. 1.0.0. 10th Oct. 2023. URL: <https://www.ctan.org/pkg/memoize>.

Index

Features are sorted by kind. Page references are given for both definitions and comments on use.

FOREST AUTOWRAPPED TOKS OPTIONS

- just, [7](#), [10](#), [17](#), [22](#), [23](#)
- just options, [16](#), [24](#)
- line no options, [16](#), [24](#)
- line options, [16](#), [24](#)
- wff options, [16](#), [24](#)

FOREST AUTOWRAPPED TOKS REGISTERS

- highlight format, [21](#)

FOREST BOOLEAN OPTIONS

- grouped, [21](#)
- highlight just, [16](#), [21](#), [23](#)
- highlight line, [16](#), [21](#), [24](#)
- highlight line no, [16](#), [21](#), [24](#)
- highlight wff, [16](#), [21](#), [23](#)
- line numbering, [24](#)
- not grouped, [21](#)
- not highlight line, [24](#)
- not highlight line no, [24](#)
- not highlight just, [23](#)
- not highlight wff, [23](#)

FOREST BOOLEAN REGISTERS

- auto move, [17](#), [26](#)
- check right, [16](#), [19](#)
- just refs left, [16](#), [20](#), [22](#)
- justifications, [17](#)
- line numbering, [17](#)
- not auto move, [17](#)
- not check right, [19](#)
- not just refs left, [20](#)
- not justifications, [17](#)
- not line numbering, [17](#)
- not single branches, [17](#)
- not subs right, [20](#)
- single branches, [17](#), [21](#)
- subs right, [16](#), [20](#)

FOREST COUNT REGISTERS

- line no shift, [6](#), [18](#), [27](#)

FOREST DIMENSION REGISTERS

- close sep, [16](#), [18](#)
- just sep, [18](#)
- line no sep, [18](#)
- line no width, [18](#)
- proof tree inner proof midpoint, [18](#)
- proof tree inner proof width, [18](#)

FOREST KEYLIST REGISTERS

- close format, [16](#), [18](#), [19](#)
- close format', [19](#)
- close with format, [16](#), [19](#)
- close with format', [19](#)
- just format, [16](#), [20](#)
- just format', [20](#)

- line no format, [16](#), [21](#)
- line no format', [21](#)
- proof statement format, [16](#), [21](#)
- proof statement format', [21](#)
- wff format, [16](#), [21](#), [24](#)
- wff format', [21](#)

FOREST STYLES

- check left, [19](#)
- checked, [8](#), [16](#), [19](#), [21](#)
- close, [16](#), [19](#), [22](#)
- just refs right, [20](#)
- line no override, [24](#)
- move by, [17](#), [21](#), [23](#)
- no line no, [24](#)
- subs, [16](#), [20](#), [22](#)
- subs left, [20](#)
- to prove, [19](#)
- zero start, [18](#)

FOREST TOKS REGISTERS

- check with, [16](#), [19](#)
- close with, [16](#), [19](#)
- merge delimiter, [21](#)
- subs with, [16](#), [20](#)

ENVIRONMENTS

- prooftree, [15](#)
- tableau, [15](#)

MACROS

- \linenumberstyle, [24](#)
- \linenumberstyle, [24](#)

PACKAGE OPTIONS

- tableaux, [15](#)

PACKAGES

- external, [25](#)
- forest, [1](#), [15](#)
- memoize, [1](#), [25](#)
- pgf, [25](#)
- prooftrees, [1](#), [15](#)

A Implementation

```
1 %% Copyright 2016-2024 Clea F. Rees
2 %%
3 %% This work may be distributed and/or modified under the
4 %% conditions of the LaTeX Project Public License, either version 1.3c
5 %% of this license or (at your option) any later version.
6 %% The latest version of this license is in
7 %%   https://www.latex-project.org/lppl.txt
8 %% and version 1.3c or later is part of all distributions of LaTeX
9 %% version 2008-05-04 or later.
10 %%
11 %% This work has the LPPL maintenance status `maintained'.
12 %%
13 %% The Current Maintainer of this work is Clea F. Rees.
14 %%
15 %% This file may only be distributed together with a copy of the package
16 %% prooftrees.
17 %%
18 %% This work consists of all files listed in manifest.txt.
19 %%
20 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
21 \NeedsTeXFormat{LaTeX2e}
22 \RequirePackage{svn-prov}
23 \ProvidesPackageSVN{$Id: prooftrees.sty 10522 2024-10-23 16:31:08Z cfrees $}[v0.9 \revinfo]
24 % define \prooftrees@enw to hold the name of the environment
25 % default is to name the environment prooftree, this ensures backwards compatibility
26 \newcommand*\prooftrees@enw{prooftree}
27 % allow users to change the name to tableau using tableaux
28 \DeclareOption{tableaux}{\renewcommand*\prooftrees@enw{tableau}}
29 % just in case
30 \DeclareOption{tableau}{\renewcommand*\prooftrees@enw{tableau}}
31 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{forest}}
32 % if \prooftree is not yet defined, set the name to prooftree; otherwise, use tableau to avoid
33 % conflict with bussproofs (which uses 'prooftree' rather than 'bussproof' as one might expect)
34 \ifcsname prooftree\endcsname
35   \renewcommand*\prooftrees@enw{tableau}%
36 \else
37   \renewcommand*\prooftrees@enw{prooftree}%
38 \fi
39 % let users override the default prooftree in case they need to load bussproofs later
40 \ProcessOptions
41 \RequirePackage{forest}[2016/12/04]
42 \RequirePackage{amssymb,amstext}
43 \newcommand*\linenumberstyle[1]{#1.}
44 % currently, keys starting 'proof tree' and macros starting 'prooftree' or 'prooftree@' are intended
45 % for internal use only
46 % this does not apply to the environment prooftree
47 % other keys and macros are intended for use in documents
48 % in particular, the style 'proof tree' is NOT intended to be used directly by the user and its
49 % direct use is ABSOLUTELY NOT SUPPORTED IN ANY WAY, SHAPE OR FORM; it is intended only for
50 % implicit use when the prooftree environment calls it
51 \forestset{% don't use @ in register/option names - the documentation is lying when it says non-
52 % alphanumeric will be converted to underscores when forming pgfmath functions ;}
53   declare boolean register={line numbering},% line numbers
54   line numbering,% default is for line numbers
55   declare boolean register={justifications},% line justifications
56   not justifications,% default is for no line justifications (b/c there's no point in enabling this
57   if the user doesn't specify any content)
```

```
53 declare boolean register={single branches},% single branches: explicitly drawn branches and a
normal level distance between lone children and their parents
54 not single branches,% default is for lone children to be grouped with their parents
55 declare boolean register={auto move},% ble mae'n bosibl, symud pethau'n awtomatig
56 auto move,% default: symud yn awtomatig
57 declare dimen register={line no width},% default will be set to the width of 99 wrapped in the line
numbering style
58 line no width'=0pt,% fallback default is 0pt
59 declare dimen register={just sep},% amount by which to shift justifications away from the main tree
60 just sep'=1.5em,% default is 1.5em
61 declare dimen register={just dist},% distance of justifications from centre of inner tree;
overrides just sep
62 just dist'=0pt,
63 declare dimen register={line no sep},% amount by which to shift line numbers away from the main
tree
64 line no sep'=1.5em,
65 declare dimen register={line no dist},% distance of line nos. from centre of inner tree; overrides
line no sep
66 line no dist'=0pt,
67 declare dimen register={close sep},% distance between closure symbols and any following annotation
68 close sep'=.75\baselineskip,
69 declare dimen register={proof tree line no x},
70 proof tree line no x'=0pt,
71 declare dimen register={proof tree justification x},
72 proof tree justification x'=0pt,
73 declare dimen register={proof tree inner proof width},
74 proof tree inner proof width'=0pt,
75 declare dimen register={proof tree inner proof midpoint},
76 proof tree inner proof midpoint'=0pt,
77 declare count register={proof tree rhif lefelau},% count the levels in the proof tree
78 proof tree rhif lefelau'=0,
79 declare count register={proof tree lcount},% count the line numbers (on the left)
80 proof tree lcount'=0,
81 declare count register={proof tree jcount},% count the justifications (on the right)
82 proof tree jcount'=0,
83 declare count register={line no shift},% adjustment for line numbering
84 line no shift'=0,
85 declare count register={proof tree aros},
86 proof tree aros'=0,
87 declare toks register={check with},
88 check with={\ensuremath{\checkmark}},
89 declare boolean register={check right},
90 check right,
91 check left/.style={not check right},
92 declare toks register={subs with},
93 subs with={\ensuremath{\backslash}},
94 declare boolean register={subs right},
95 subs right,
96 subs left/.style={not subs right},
97 declare toks register={close with},
98 close with={\ensuremath{\otimes}},
99 declare keylist register={close format},
100 close format={font=\scriptsize},
101 declare keylist register={close with format},
102 close with format={},
103 declare toks register={merge delimiter},
104 merge delimiter={\text{; }},
105 declare boolean register={just refs left},
106 just refs left,
107 just refs right/.style={not just refs left},
108 declare keylist register={just format},
```

```

109 just format={},
110 declare keylist register={line no format},
111 line no format={},
112 declare autowrapped toks register={highlight format},
113 highlight format={draw=gray, rounded corners},
114 declare keylist register={proof statement format},
115 proof statement format={},
116 declare keylist register={wff format},
117 wff format={},
118 declare boolean={proof tree justification}{0},
119 declare boolean={proof tree line number}{0},
120 declare boolean={grouped}{0},
121 declare boolean={proof tree phantom}{0},
122 declare boolean={highlight wff}{0},
123 declare boolean={highlight just}{0},
124 declare boolean={highlight line no}{0},
125 declare boolean={highlight line}{0},
126 Autoforward={highlight line}{highlight just, highlight wff, highlight line no},
127 declare boolean={proof tree toing}{0},
128 declare boolean={proof tree toing with}{0},
129 declare boolean={proof tree rhiant cymysg}{0},
130 declare boolean={proof tree rhifo}{1},
131 declare boolean={proof tree arweinydd}{0},
132 declare autowrapped toks={just}{},
133 declare toks={proof tree rhestr rhifau llinellau}{},
134 declare toks={proof tree close}{},
135 declare toks={proof tree rhestr rhifau llinellau cau}{},
136 declare autowrapped toks={just options}{},
137 declare autowrapped toks={line no options}{},
138 declare autowrapped toks={wff options}{},
139 declare autowrapped toks={line options}{},
140 Autoforward={line options}{just options={#1}, line no options={#1}, wff options={#1}},
141 declare count={proof tree toing by}{0},
142 declare count={proof tree cadw toing by}{0},
143 declare count={proof tree toooing}{0},
144 declare count={proof tree proof line no}{0},
145 % keylists for internal storage
146 declare keylist={proof tree jrefs}{},
147 declare keylist={proof tree crefs}{},
148 % keylists for use in stages
149 declare keylist={proof tree ffurf}{},
150 declare keylist={proof tree symud awto}{},
151 declare keylist={proof tree creu nodiadau}{},
152 declare keylist={proof tree nodiadau}{},
153 % > not documented yet, I think
154 % > now indicates use of process when it is the first token, preceding a list of instructions as
    opposed to pgfmath stuff
155 define long step={proof tree symud}{%
156     root,sort by={>{0}{level},>{_0<}{1}{n children}},sort'=descendants
157 },
158 define long step={proof tree cywiro symud}{%
159     root,if line numbering={n=2}{n=1},sort by={>{0}{level},>{_0<}{1}{n children}},sort'=descendants
160 },
161 define long step={proof tree camau}{% updated version of defn. from saso's code (forest2-saso-
    ptsz.tex) & http://chat.stackexchange.com/transcript/message/28321501#28321501
162     root,sort by={>{0}{y},>{0w1+d}{x}{-#1}},sort'={filter={descendants}{>{00!&}{proof tree rhifo}{
    proof tree phantom}}}% angen +d - gweler http://chat.stackexchange.com/transcript/message
    /28607212#28607212
163 },
164 define long step={proof tree wffs}{% coeden brif yn unig ar ôl i greu nodiadau
165     fake=root,if line numbering={n=2}{n=1},tree

```

```
166 },
167 checked/.style={% mark discharge with optional name substituted into existential
168   delay={%
169     if check right={%
170       content+='{ \ \forestregister{check with}#1}',
171     }{%
172       +content='{ \forestregister{check with}#1\ }',
173     },
174   },
175 },
176 subs/.style={% mark substitution of name into universal
177   delay={%
178     if subs right={%
179       content+='{ \ \forestregister{subs with}#1}',
180     }{%
181       +content='{ \forestregister{subs with}#1\ }',
182     },
183   },
184 },
185 close/.style={% this now uses nodes rather than a label to accommodate annotations; closing must be
  done before packing the tree to ensure that sufficient space is allowed for the symbol and any
  following annotation; the annotations must be processed before anything is moved to ensure that the
  correct line numbers are used later, even if the references are given as relative node names
186   if={%
187     >{_{_}}{#1}{}%
188   }{}{%
189     temptoksb={},
190     temptoksa={#1},
191     split register={temptoksa}{:}{proof tree close,temptoksb},
192     if temptoksb={}{}{%
193       split register={temptoksb}{,}{proof tree cref},
194     },
195   },
196   delay={%
197     append={% this node holds the closure symbol
198       [\forestregister{close with},
199       not proof tree rhifo,
200       proof tree phantom,
201       grouped,
202       no edge,
203       process keylist register=close with format,
204       before computing xy={% adjust the distance between the closure symbol and any annotation
205         delay={%
206           l'=\baselineskip,% cywiro? fel arall, bydda'r peth byth yn cael ei wneud achos proof
tree phantom? dim yn siwr o gwbl
207           for children={%
208             l/.register=close sep,
209           },
210         },
211       },
212       before drawing tree={%
213         if={>{RR|}}{line numbering}{justifications}}{%
214           proof tree proof line no/.option=!parent.proof tree proof line no,
215         }{},
216       },
217       if={%
218         >{_{_}}{#1}{}%
219       }{}{% don't create a second node if there's no annotation
220         delay={%
221           append={% this node holds the annotation, possibly including cross-references which
will be relative to the node's grandparent
```



```
222     [,
223     not proof tree rhifo,
224     proof tree phantom,
225     grouped,
226     no edge,
227     process keylist register=close format,
228     if={%
229         >{0_}={!parent,parent.proof tree close}{}%
230     }{content/.option={!parent,parent}.proof tree close},
231     proof tree crefs/.option={!parent,parent}.proof tree crefs,
232     delay={%
233         {!parent,parent}.proof tree crefs'={},
234     },
235     before drawing tree={%
236         if={>{RR|}{line numbering}{justifications}}{%
237             proof tree proof line no/.option={!parent,parent}.proof tree proof line no,
238             }{ },
239         },
240     ]%
241 },
242 },
243 },
244 ]%
245 },
246 },
247 },
248 proof tree line no/.style={% creates the line numbers on the left; note that it *does* matter that
these are part of the tree, even though they do not need to be packed or to have xy computed;
moreover, it matters that each is the child of the previous line number... so it won't do for them to
*remain* siblings, even though that's fine when they are created.
249     anchor=base west,
250     no edge,
251     proof tree line number,
252     text width/.register=line no width,
253     x'/.register=proof tree line no x,
254     process keylist register=line no format,
255     delay={%
256         proof tree lcount'+=1,
257         tempcounta/.process={RRw2+n}{proof tree lcount}{line no shift}{##1+##2},
258         content/.process={Rw1}{tempcounta}{\linenumberstyle{##1}},% content i.e. the line number
259         name/.expanded={line no \forestregister{tempcounta}},% name them so they can be moved later
260         typeset node,
261         if proof tree lcount>=3{% the initial location of most line numbers is incorrect and they must
be moved
262             for previous={% move the line number below the previous line number
263                 append/.expanded={line no \forestregister{tempcounta}}
264             },
265             }{ },
266         },
267     },
268 proof tree line justification/.style={% creates the justifications on the right but does not yet
specify any content
269     anchor=base west,
270     no edge,
271     proof tree justification,
272     x'/.register=proof tree justification x,
273     process keylist register=just format,
274     delay={%
275         proof tree jcount'+=1,
276         tempcounta/.process={RRw2+n}{proof tree jcount}{line no shift}{##1+##2},
277         name/.expanded={just \forestregister{tempcounta}},% name them so they can be moved
```

```
278 typeset node,% angen i osgoi broblemau 'da highlight just/line etc.
279 if proof tree jcount>=3{% correct the location as for the line numbers (cf. line no style)
280   for previous={%
281     append/.expanded={just \forestregister{temptounta}},
282   },
283   {}},
284 },
285 },
286 zero start/.style={%
287   line no shift'+=-1,
288 },
289 to prove/.style={% sets a proof statement
290   for root={%
291     before typesetting nodes={%
292       content={#1},
293       phantom=false,
294       baseline,
295       if line numbering={anchor=base west}{anchor=base},
296       process keylist register=proof statement format
297     },
298     before computing xy={%
299       delay={%
300         for children={%
301           l=1.5*\baselineskip,
302         },
303       },
304     },
305   },
306 },
307 proof tree/.style={% this style should **NOT** be used directly in a forest environment - see notes
at top of this file
308   for tree={%
309     parent anchor=children,% manual 64
310     child anchor=parent,% manual 64
311     math content,
312     delay={%
313       if just={}{% if we've got justifications, make sure nodes are created for them later and
split out cross-references so we identify the correct nodes before anything gets moved, allowing the
use of relative node names
314         justifications,
315         temptoksa={},
316         split option={just}{:}{just,temptoksa},
317         if temptoksa={}{%
318           split register={temptoksa}{,}{proof tree jref},
319         },
320       },
321       if content={}{% if there's no proof statement
322         if level=0{%
323           shape=coordinate,
324         },
325       }},
326     },
327   },
328   where level=0{%
329     for children={% no edges from phantom root or proof statement to children
330       before typesetting nodes={%
331         no edge,
332       },
333     },
334     delay={%
335       if content={}{phantom}{},
```

```

336     if line numbering={% create the line numbers if appropriate
337         parent anchor=south west,
338         if line no width={0pt}{%
339             line no width/.pgfmath={width("\noexpand\linenumberstyle{99}")},
340         }{},
341     }{},
342 },
343 proof tree creu nodiadau={% this is processed after computing xy
344     if=>{RR|}{line numbering}{justifications}}{% count proof lines if necessary
345     proof tree rhif lefelau'/.register=line no shift,
346     for proof tree camau={%
347         if level>=1{%
348             if={%
349                 >{00<}{y}{!back.y}%
350             }{%
351                 proof tree rhif lefelau'+=1,
352                 proof tree proof line no'/.register=proof tree rhif lefelau,
353             }{%
354                 proof tree proof line no'/.register=proof tree rhif lefelau
355             },
356         }{},
357     },
358     proof tree inner proof midpoint/.min=>{00w2+d}{x}{min x}{##1+##2}}{fake=root,descendants},
359     proof tree inner proof width/.max=>{00w2+d}{x}{max x}{##1+##2}}{fake=root,descendants},
360     proof tree inner proof width-/ .register=proof tree inner proof midpoint,
361     proof tree inner proof midpoint+/.process={Rw+d{proof tree inner proof width}{##1/2}},
362 }{},
363 if line numbering={% get the x position of line numbers and adjust the location and alignment
of the proof statement
364     proof tree line no x/.min=>{00w2+d}{x}{min x}{##1+##2}}{fake=root,descendants},
365     if={%
366         > Rd= {line no dist}{0pt}%
367     }{%
368         proof tree line no x-/ .register=line no sep,
369     }{%
370         tempdima/.register=proof tree inner proof width,
371         tempdima:=2,
372         if={%
373             > RR< {line no dist}{tempdima}%
374         }{}{%
375             proof tree line no x/.register=proof tree inner proof midpoint,
376             proof tree line no x-/ .register=line no dist,
377         },
378     },
379     proof tree line no x-/ .register=line no width,
380     for root={%
381         tempdimc/.option=x,
382         x'/.register=proof tree line no x,
383         x'-.option=min x,
384     },
385     prepend={% create line numbers on left
386     [,
387         proof tree line no,
388         % () to group are required here - otherwise, the -1 (or -2 or whatever) is silently
ignored
389         repeat={((proof_tree_rhif_lefelau)-1)-(line_no_shift)}{% most are created in the wrong
place but proof tree line no moves them later
390             delay n={proof_tree_lcount}{
391                 append=[, proof tree line no]},
392             },
393     ],

```

```

394     ]%
395   },
396   {}},
397   if justifications={% get the x position of justifications and create the nodes which will
hold the justification content, if required
398     proof tree justification x/.max={>{00w2+d}{x}{max x}{##1+##2}}{fake=root,descendants},
399     if={%
400       > Rd= {just dist}{0pt}%
401     }{%
402       proof tree justification x+/.register=just sep,
403     }{%
404       tempdima/.register=proof tree inner proof width,
405       tempdima:=2,
406       if={%
407         > RR< {just dist}{tempdima}%
408       }{}{%
409         proof tree justification x/.register=proof tree inner proof midpoint,
410         proof tree justification x+/.register=just dist,
411       },
412     },
413     append={%
414       [,
415         proof tree line justification,
416         repeat={((proof_tree_rhif_lefelau)-1)-(line_no_shift)}{% most are created in the wrong
place but proof tree line justification moves them later
417           delay n={proof_tree_jcount}{%
418             append={[, proof tree line justification]},
419           },
420         ]%
421       ]%
422     },
423   }{}},
424 },
425 }{%
426   delay={%
427     if single branches={} {% automatically group lines if not using single branches
428       if n children=1{%
429         for children={%
430           grouped,
431         },
432       }{}},
433     },
434   },
435   before typesetting nodes={% apply wff-specific highlighting and additional TikZ keys
436     process keylist register=wff format,
437     if highlight wff={node options/.register=highlight format}{},
438     node options/.option=wff options,
439   },
440 },
441   proof tree ffurf={% processed before proof tree symud auto: adjusts the alignment of lines when
some levels of the tree are grouped together either whenever the number of children is only 1 or by
applying the grouped style to particular nodes when specifying the tree
442     if auto move={%
443       if single branches={%
444         where={%
445           >{0! _0< 0 &&}{grouped}{2}{level}{proof tree rhifo}%
446         }{%
447           if={%
448             >{_0= _0< &}{1}{!parent.n children}{1}{!parent,parent.n children}%
449           }{%
450             not tempboola,

```

```
451         for root/.process={0w1}{level}{%
452             for level={##1}{%
453                 if={%
454                     >{_0<_0= &}{1}{!parent.n children}{1}{n}%
455                 }{%
456                     tempboola,
457                     {}},
458             },
459         },
460         if tempboola={%
461             proof tree toing,
462             {}},
463         }{},
464     }{},
465     }{},
466     where={%
467         >{0_0<0 &&}{grouped}{1}{level}{proof tree rhifo}%
468     }{% this searches for certain kinds of structural asymmetry in the tree and attempts to move
lines appropriately in such cases - the algorithm is intended to be relatively conservative (not in
the sense of 'cautious' or 'safe' but in the sense of 'reflection of the overlapping consensus of
reasonable users' / 'what would be rationally agreed behind the prooftrees veil of ignorance';
however, I should have realised I actually had 'the overlapping consensus of reasonable Beamer users'
in mind rather than 'the overlapping consensus of reasonable users', so there is now an option to
turn it off; apologies if this comment previously misclassified you as 'unreasonable'; apologies for
the inconvenience if you are an unreasonable user)
469         not tempboola,
470         for root/.process={0w1}{level}{%
471             for level={##1}{%
472                 if={%
473                     >{_0<_0= &}{1}{!parent.n children}{1}{n}%
474                 }{%
475                     tempboola,
476                     {}},
477             },
478         },% Sao: http://chat.stackexchange.com/transcript/message/27874731#27874731, see also http://chat.stackexchange.com/transcript/message/27874722#27874722
479         if tempboola={%
480             if n children=0{%
481                 if={>{00|}{!parent.proof tree toing}{!parent.proof tree toing with}}{% we're already
moving the parent and the child will move with the parent, so we can just mark this and do nothing
else
482                 proof tree toing with,
483                 }{%
484                 for root/.process={0w1}{level}{% don't move a terminal node even in case of asymmetry
: instead, create a separate proof line for terminal nodes on this level which are only children, by
moving children with siblings on this level down a proof line, without altering their physical
location
485                 % this makes the tree more compact and stops it looking silly
486                 for level={##1}{%
487                     if={%
488                         >{_0<_0= &}{1}{!parent.n children}{1}{n}%
489                     }{% this just serves to keep the levels nice for the sub-tree and ensure things
align. We need this because we want to skip a level here to allow room for the terminal node in the
other branch
490                     for parent={%
491                         if proof tree rhiant cymysg={}{% we mark the parent to avoid increasing the
line number of its descendants more than once
492                         proof tree rhiant cymysg,
493                         for descendants={%
494                             proof tree toing by'+=1,
495                             },
```

```
496         },
497     },
498     {}},
499 },
500 },% Sao: http://chat.stackexchange.com/transcript/message/27874731#27874731, see
also http://chat.stackexchange.com/transcript/message/27874722#27874722
501 },
502 no edge,
503 }{%
504 if={%
505 >{_0= _0< &}{1}{!parent.n children}{1}{!parent,parent.n children}%
506 }{% don't try to move if the node has more than 1 child or the grandparent has no more
than that; otherwise, mark the node as one to move - we figure out where to move it later
507 proof tree toing,
508 }{no edge},
509 },
510 }{no edge},
511 }},
512 {}},
513 },
514 proof tree symud awto={% processed before typesetting nodes: if _this_ could be done during
packing, that would be very nice, even if the previous stuff can't be
515 if auto move={%
516 proof tree aros'=0,
517 for proof tree symud={%
518 if proof tree toing={% this relies on an experimental feature of forest, which is anffodus
519 for nodewalk={fake=parent,fake=sibling,descendants}{do dynamics},
520 delay n={\forestereregister{proof tree aros}}{%
521 tempcounta/.max={>{0000w4+n}{level}{proof tree toing by}{proof tree toooing}{proof tree
rhifo}{(##1+##2+##3)*##4}}{parent,sibling,descendants},
522 if tempcounta>=1{%
523 if={%
524 >{Rw1+n 00w2+n >}{tempcounta}{##1+1}{level}{proof tree toing by}{##1+##2}%
525 }{%
526 tempcounta-/.option=level,
527 tempcounta'+=1,
528 move by/.register=tempcounta,
529 }{no edge},
530 }{no edge},
531 },
532 proof tree aros'+=4,
533 }},
534 },
535 }},
536 },
537 proof tree nodiadau={% processed after proof tree creu nodiadau and before before drawing tree:
creates annotation content which may include cross-references, applies highlighting and additional
TikZ keys to line numbers, justifications and to wffs where specified for entire proof lines
538 where proof tree crefs={}{}{% resolve cross-refs in closures
539 split option={proof tree crefs}{,}{proof tree rhif llinell cau},
540 if content={}{}{%
541 content/.option=proof tree rhestr rhifau llinellau cau,
542 }{%
543 content+/.process={_0}{\ }{proof tree rhestr rhifau llinellau cau},
544 },
545 typeset node,
546 },
547 if line numbering={% apply highlighting and additional TikZ keys to line numbers; initial
alignment of numbers with proof lines
548 for proof tree wffs={%
549 if highlight line no={%
```

```

550         for name/.process={0w1000w3}{proof tree proof line no}{line no ##1}{proof tree proof line
no}{line no options}{y}{% from Sao's anti-pgfmth version - rhaid ddweud proof tree proof line no yn
ddwywaith ?! dim yn bosibl i ailddefnyddio'r gyntaf ?!
551         node options/.register=highlight format,
552         ##2,
553         y'##3,
554         proof tree proof line no'##1,
555         typeset node,
556     }%
557 }{%
558     if line no options={} {%
559         if proof tree phantom={} {%
560             for name/.process={0w100w2}{proof tree proof line no}{line no ##1}{proof tree proof
line no}{y}{%
561                 y'##2,
562                 proof tree proof line no'##1,
563             }%
564         },
565     }{%
566         for name/.process={0w1000w3}{proof tree proof line no}{line no ##1}{proof tree proof
line no}{line no options}{y}{%
567             ##2,
568             y'##3,
569             proof tree proof line no'##1,
570             typeset node,
571         }%
572     },
573 },
574 },
575 }{%,
576     if justifications={% initial alignment of justifications with proof lines, addition of content,
resolution of cross-references and application of highlighting and additional TikZ keys
577     for proof tree wffs={%
578         if just={} {%
579             if proof tree phantom={} {%
580                 for name/.process={0w100w2}{proof tree proof line no}{just ##1}{proof tree proof line
no}{y}{% from Sao's anti-pgfmth version - rhaid ddweud proof tree proof line no yn ddwywaith ?! dim
yn bosibl i ailddefnyddio'r gyntaf ?!
581                 y'##2,
582                 proof tree proof line no'##1,
583             }%
584         },
585         }{% puts the content of the justifications into the empty justification nodes on the right;
because this is done late, the nodes need to be typeset again
586         if proof tree jrefs={} {% resolve cross-refs in justifications
587             split option={proof tree jrefs}{,}{proof tree rhif llinell},
588             if just refs left={%
589                 +just/.process={0_}{proof tree rhestr rhifau llinellau}{\ },
590             }{%
591                 just+/.process={0_}{\ }{proof tree rhestr rhifau llinellau},
592             },
593         },
594         if highlight just={% apply highlighting and additional TikZ keys to justifications, set
content and merge any conflicting specifications, warning user if appropriate
595         for name/.process={0w1000w4}{proof tree proof line no}{just ##1}{proof tree proof line
no}{just}{just options}{y}{% from Sao's anti-pgfmth version - rhaid ddweud proof tree proof line no
yn ddwywaith ?! dim yn bosibl i ailddefnyddio'r gyntaf ?!
596             if={%
597                 >{0_= 0_= |}{content}{}{content}{##2}%
598             }{% gweler isod - o gôd Sao
599                 content={##2},

```

```
600     }{%
601         content+='{\\foresteregister{merge delimiter}##2},
602         TeX={\\PackageWarning{prooftrees}{Merging conflicting justifications for line ##1!
Please examine the output carefully and use "move by" to move lines later in the proof if required.
Details of how to do this are included in the documentation.}},
603     },
604     node options/.register=highlight format,
605     ##3,
606     y'==#4,
607     proof tree proof line no'==#1,
608     typeset node,
609     }% do NOT put a comma here!
610 }{%
611     for name/.process={0w1000w4}{proof tree proof line no}{just ##1}{proof tree proof line
no}{just}{just options}{y}{% from Sao's anti-pgfmath version - rhaid ddweud proof tree proof line no
yn ddwywaith ?! dim yn bosibl i ailddefnyddio'r gyntaf ?!
612     if={% from Sao's anti-pgfmath version - I appreciate this is faster, but why is it *
required*?!
613         >{0_= 0_= |}{content}{}{content}{##2}%
614     }{%
615         content={##2},
616     }{%
617         content+='{\\foresteregister{merge delimiter}##2},
618         TeX={\\PackageWarning{prooftrees}{Merging conflicting justifications for line ##1!
Please examine the output carefully and use "move by" to move lines later in the proof if required.
Details of how to do this are included in the documentation.}},
619     },
620     ##3,
621     y'==#4,
622     proof tree proof line no'==#1,
623     typeset node,
624     }% do NOT put a comma here!
625 }
626 },
627 },
628 }{},
629 for proof tree wffs={% apply highlighting and TikZ keys which are specified for whole proof
lines to all applicable wffs
630     if proof tree phantom={}%
631     if highlight line={%
632         for proof tree wffs/.process={00w2}{proof tree proof line no}{line options}{%
633             if proof tree proof line no={##1}{%
634                 node options/.register=highlight format,
635                 ##2,
636             }{%
637             },
638         }{%
639             for proof tree wffs/.process={00w2}{proof tree proof line no}{line options}{%
640                 if proof tree proof line no={##1}{##2}{},
641             },
642         },
643         delay={typeset node},
644     },
645 },
646 },
647 before packing={% initial alignment so we don't get proof line numbers incrementing due to
varying height/depth of nodes, for example - when single branches is true and few nodes are grouped,
this is also a reasonable first approximation
648     for tree={%
649         tier/.process={00w2+nw1}{level}{proof tree toing by}{##1+##2}{tier ##1},
650     },
```



```

651     for root={% if there's no proof statement, adjust the alignment of the proof relative to the
surrounding text
652         if content={}%
653             !{n=1}.baseline,
654         },
655     },
656 },
657 before computing xy={% adjust distance between levels for grouped nodes after tree is packed
658     for tree={%
659         if={%
660             >{0 _0< &}{grouped}{1}{level}%
661         }{% osgoi overlapping nodes, if posibl: cwestiwn https://tex.stackexchange.com/q/456254/
662             not tempboola,
663             tempcounta/.option=level,
664             tempcountb/.option=proof tree toing,
665             tempcountb+/.option=proof tree toooing,
666             for nodewalk={fake=root, descendants}{if={> R0= On> 0! 0! 00w2+nR= &&&&
667                 {tempcounta}{level} {!u.n children}{1} {proof tree arweinydd} {proof tree phantom} {
proof tree toing by} {proof tree toooing}{##1+##2} {tempcountb}
668                 }{tempboola}{}},
669             if tempboola={}{l'=\baselineskip},
670         },
671     },
672 },
673 before drawing tree={% set final alignment for proof lines which have been moved by effectively
grouping lead nodes and moving their subtrees accordingly - this requires that each line number and
justification be the child of the previous one and that if justifications are used at all, then
justifications exist for all proof lines, even if empty
674     if={>{RR|R!&}{line numbering}{justifications}{single branches}}{% correct the alignment of move
by lines when single branches is false - o fersiwn anti-pgfmth Sao
675         tempdimc'=0pt,% track cumulative adjustments to line numbers and justifications
676         for proof tree cywiro symud={%
677             if proof tree arweinydd={% only examine the lead nodes - their descendants need the same (
cumulative) adjustments
678                 tempdima'/.option=y,
679                 if line numbering={% if there are line numbers, we use the previous line number's
vertical position
680                     for name/.process={0w1+nw1}{proof tree proof line no}{##1-1}{line no ##1}{% arafach ?
681                         tempdimb'/.option=y,
682                     }%
683                 }{% if not, we use the previous justification's vertical position
684                     for name/.process={0w1+nw1}{proof tree proof line no}{##1-1}{just ##1}{% arafach ?
685                         tempdimb'/.option=y,
686                     }%
687                 },
688                 for parent={% the parent (which will be a phantom) gets aligned with the previous line
689                     y'/.register=tempdimb,
690                 },
691                 if tempdimb<={0pt}{% adjust so we align this line below the previous one (assuming we're
going down)
692                     tempdimb'--\baselineskip,
693                 }{%
694                     tempdimb'+=\baselineskip,
695                 },
696                 tempdimb'-/.register=tempdima,% how far are we moving?
697                 for tree={% adjust this node and all descendants
698                     y'+/.register=tempdimb,
699                 },
700                 tempdimb'-/.register=tempdimc,% deduct any tracked cumulative adjustments to line numbers
and justifications
701                 if line numbering={% adjust the line numbers, if any

```

```
702         for name/.process={0w1}{proof tree proof line no}{line no ##1}{%
703             for tree={%
704                 y'+/.register=tempdimb,
705                 },
706             }%,
707         }{},
708         if justifications={% adjust the justifications, if any
709             for name/.process={0w1}{proof tree proof line no}{just ##1}{% t. 60 manual 2.1 rc1
710                 for tree={%
711                     y'+/.register=tempdimb,
712                     },
713                 }%,
714             }{},
715             tempdimc'/.register=tempdimb,% add the adjustment just implemented to the tracked
cumulative adjustments for line numbers and/or justifications
716         }{},
717     },
718     }{},
719     if={%
720         > RR| {auto move}{single branches}%
721     }{}{%
722         where proof tree arweinydd={%
723             for nodewalk={%
724                 save append={proof tree walk}{%
725                     current,
726                     do until={%
727                         > 0+t_+t=! {content}{}%
728                     }{parent}%
729                 }%,
730             }{},
731         }{},
732         where level>=1{%
733             if grouped={%
734                 if in saved nodewalk={current}{proof tree walk}{}{%
735                     no edge,
736                 },
737             }{},
738         }{},
739     },
740 },
741 },
742 move by/.style={% this implements both the automated moves prooftrees finds necessary and any
additional moves requested by the user - more accurately, it implements initial moves, which may get
corrected later (e.g. to avoid skipping numbers or creating empty proof lines, which we assume aren't
wanted)
743     if={
744         >{_n<}{0}{#1}%
745     }{% only try to move the node if the target line number exceeds the one i.e. the line number is
to be positively incremented
746         proof tree cadw toing by/.option=proof tree toing by,
747         proof tree arweinydd,
748         for tree={%
749             if={%
750                 >{_n<}{1}{#1}%
751             }{% track skipped lines for which we won't be creating phantom nodes
752                 proof tree toing by+=#1-2,
753                 proof tree toooing'+=1,
754             }{},
755         },
756         delay={%
757             replace by={% insert our first phantom
```

```
758     [,
759     if={%
760     >{_n<}{1}{#1}%
761     }{%
762     child anchor=parent,
763     parent anchor=parent,
764     }{%
765     child anchor=children,
766     parent anchor=children,
767     },
768     proof tree phantom,
769     edge path/.option=!last dynamic node.edge path,% Sao ivanovi: http://chat.stackexchange.com/transcript/message/27990955#27990955
770     edge/.option=!last dynamic node.edge,
771     append,
772     before drawing tree={%
773     if={>{RR|}{line numbering}{justifications}}{%
774     proof tree proof line no/.process={0w1+n}{!parent.proof tree proof line no}{##1+1},
775     }{},
776     },
777     if={%
778     >{_n<}{1}{#1}%
779     }{% if we are moving by more than 1, we insert a second phantom so that a node with
siblings which is moved a long way will not get a unidirectional edge but an edge which looks similar
to others in the tree (by default, sloping down a line or so and then plummeting straight down
rather than a sharply-angled steep descent)
780     delay={%
781     append={%
782     [,
783     child anchor=parent,
784     parent anchor=parent,
785     proof tree toing by=#1-2+proof_tree_cadw_toing_by,
786     proof tree phantom,
787     edge path/.option=!u.edge path,
788     edge/.option=!u.edge,
789     before drawing tree={%
790     if={>{RR|}{line numbering}{justifications}}{%
791     proof tree proof line no/.process={0w1+n}{!n=1.proof tree proof line no
}{##1-1},
792     }{},
793     },
794     append=!sibling,
795     ]%
796     },
797     },
798     }{%
799     if single branches={} {%
800     delay={%
801     for children={%
802     no edge,
803     },
804     },
805     },
806     },
807     ]%
808     },
809     },
810     }{%
811     TeX/.process={0w1}{name}{\PackageWarning{prooftrees}{Line not moved! I can only move things
later in the proof. Please see the documentation for details. ##1}},
812     },
```

```
813 },
814 proof tree cref/.style={% get the names of nodes cross-referenced in closure annotations for use
later
815 proof tree crefs+/.option=#1.name,
816 },
817 proof tree rhif llinell cau/.style={% get the proof line numbers of the cross-referenced nodes in
closure annotations, using the list of names created earlier
818 if proof tree rhestr rhifau llinellau cau={}{}{%
819 proof tree rhestr rhifau llinellau cau+={,\},
820 },
821 proof tree rhestr rhifau llinellau cau+/.option=#1.proof tree proof line no,
822 },
823 proof tree jref/.style={% get the names of nodes cross-referenced in justifications for use later
824 proof tree jrefs+/.option=#1.name,
825 },
826 proof tree rhif llinell/.style={% get the proof line numbers of the cross-referenced nodes in
justifications, using the list of names created earlier
827 if proof tree rhestr rhifau llinellau={}{}{%
828 proof tree rhestr rhifau llinellau+={,\},
829 },
830 proof tree rhestr rhifau llinellau+/.option=#1.proof tree proof line no,% works according to Sao'
s anti-pgfmath version
831 },
832 line no override/.style={% 2018-02-19 ateb https://tex.stackexchange.com/a/416037/
833 before drawing tree={
834 for name/.process={0w}{proof tree proof line no}{line no ##1}{
835 content=\linenumberstyle{#1},
836 typeset node,
837 },
838 },
839 },
840 no line no/.style={% 2018-02-19 gweler uchod
841 before drawing tree={
842 for name/.process={0w}{proof tree proof line no}{line no ##1}{
843 content=,
844 typeset node,
845 },
846 },
847 },
848 proof tree dadfygio/.style={% style for use in debugging moves which displays information about
nodes in the tree
849 before packing={%
850 for tree={%
851 label/.process={000w3}{level}{proof tree toing by}{id}{[red,font=\tiny,inner sep=0pt,outer
sep=0pt, anchor=south]below:##1/##2/##3},
852 },
853 },
854 before drawing tree={%
855 for tree={%
856 delay={%
857 tikz+/.process={0w1}{proof tree proof line no}{\node [anchor=west, font=\tiny, text=blue,
inner sep=0pt] at (.east) {##1}; },
858 },
859 },
860 },
861 },
862 proof tree alino/.style={% debugging / dangos dimension stuff
863 before drawing tree={%
864 tikz+/.process={%
865 RRRRw4{proof tree inner proof midpoint}{line no width}{line no dist}{just dist}
866 {
```

```

867     \begin{scope}[densely dashed]
868     \draw [darkgray] (##1,0) coordinate (a) -- (a |- current bounding box.south);
869     \draw [green] (current bounding box.west) -- ++(##2,0) coordinate (b);
870     \draw [blue] (b) -- ++(##3,0) coordinate (c);
871     \draw [magenta] (c) -- ++(##4,0);
872     \end{scope}
873   }%
874 },
875 },
876 },
877 }
878 % \environbodyname\prooftreebody
879 \bracketset{action character=@}
880 \NewDocumentEnvironment{\prooftrees@enw}{ m +b }{% \forest/\endforest from egreg's answer at http://tex.stackexchange.com/a/229608/
881   \forest
882   (
883     stages={% customised definition of stages - we don't use any custom stages, but we do use
884       several custom keylists, where the processing order of these is critical
885       for root'={% nothing is removed from the standard forest definition - we only change it by
886         adding to it
887         process keylist register=default preamble,
888         process keylist register=preamble,
889       },
890       process keylist=given options,
891       process keylist=before typesetting nodes,
892       % first two additions: process two custom keylists after before typesetting nodes and before
893       typesetting nodes
894       process keylist=proof tree ffurf,
895       process keylist=proof tree symud awto,
896       typeset nodes stage,
897       process keylist=before packing,
898       pack stage,
899       process keylist=before computing xy,
900       compute xy stage,
901       % second two additions: process two custom keylists after computing xy and before before
902       drawing tree
903       process keylist=proof tree creu nodiadau,
904       process keylist=proof tree nodiadau,
905       process keylist=before drawing tree,
906       draw tree stage,
907     },
908   )%
909   proof tree,% apply the proof tree style, which sets keylists from both forest's defaults and our
910   custom additions
911   #1,% insert user's preamble, empty or otherwise - this allows the user both to override our
912   defaults (e.g. by setting a non-empty proof statement or a custom format for line numbers) and to
913   customise the tree using forest's facilities in the usual way - BUT customisations of the latter kind
914   may or may not be effective, may or may not have undesirable - not to say chaotic - consequences,
915   and may or may not cause compilation failures (structural changes, in particular, should be avoided
916   completely)
917   [, name=proof statement @#2]%
918   \endforest
919 }{}
920
921 \ExplSyntaxOn
922 \cs_new_protected_nopar:Npn \__prooftrees_memoize:n #1
923 {
924   \mmzset{
925     auto = { #1 } { memoize },
926   }

```

```
917 }
918 \cs_generate_variant:Nn \__prooftrees_memoize:n { V }
919 \hook_gput_code:nnn { begindocument / before } { . }
920 {% paid â memoize bussproofs prooftree ...
921   \@ifpackageloaded{memoize}{
922     \__prooftrees_memoize:V \prooftrees@enw
923   }{}
924 }
925 \ExplSyntaxOff
926
927 \endinput
928 %% end prooftrees.sty
```