# The ebproofx package
# ebproof extended

Jay Lee*

Version 1.0.0 – Released 2024-01-10

## Contents


1 Introduction 1

2 Environments 2

3 Statements 2

A Implementation 3

A.1 Parameters . . . . . 3

A.2 Proof boxes . . . . . 5

A.3 Making inferences . . . . . 9

A.4 Stack-based interface . . . . 10

A.5 Document interface . . . . . 13


## 1 Introduction

The ebproofx package provides an ergonomic and scalable interface for typesetting proof trees. However, it has been a long-standing issue to stack hypotheses using ebproof.

This is an extended work of ebproof that allows users to stack multiple hypotheses vertically[1]. It generalizes `\infer` to accept a comma-separated list of integers, and a single integer works as well. Moreover, I believe this will reduce, if not resolve, a need to make ebproof compatible with the mathpartir environment[2].

In addition, it provides a wrapper environment `InfRule` that typesets the rule names more space-efficiently. Here is an example[3]:

TA-App
$$\frac{\Gamma \vdash t_1 : T_1 \qquad \Gamma \vdash T_1 \Uparrow (T_{11} \to T_{12}) \qquad \Gamma \vdash t_2 : T_2 \qquad \Gamma \vdash T_2 <: T_{11}}{\Gamma \vdash t_1\ t_2 : T_{12}}$$

```
\begin{InfRule}{TA-App}
  \hypo{\Gamma \vdasharrow t_1 : T_1}
  \hypo{%
    \Gamma \vdasharrow%
      T_1 \Uparrow (T_{11} \to T_{12})}
  \hypo{\Gamma \vdasharrow t_2 : T_2}
  \hypo{\Gamma \vdasharrow T_2 <: T_{11}}
  \infer{2, 2}{%
    \Gamma \vdasharrow t_1\ t_2 : T_{12}}
\end{InfRule}
```

The web page of the project is at https://github.com/Zeta611/ebproofx.

*E-mail: jaeho.lee@snu.ac.kr

[1] https://framagit.org/manu/ebproof/issues/7, https://tex.stackexchange.com/q/530939

[2] https://framagit.org/manu/ebproof/issues/6, https://github.com/jonsterling/latex-ebproof-rules

[3] This example is taken from *Types and Programming Languages* by Benjamin C. Pierce.

## 2 Environments

Here we only explain the newly introduced environment `InfRule`. Consult the documentation of ebproof for other environments.

`InfRule`

```
\begin{InfRule}{⟨name⟩}[⟨alignment⟩]
  ⟨statements⟩
\end{InfRule}
```

This typesets the rule described by the ⟨*statements*⟩. The ⟨*alignment*⟩ provides an alignment specifier for the rule name ⟨*name*⟩—`l` for left, `r` for right, and `c` for center are available. `l` is the default alignment specifier.

$$\frac{\Gamma \vdash t_1 : T_{11} \to T_{12} \qquad \Gamma \vdash t_2 : T_2}{\Gamma \vdash t_1\ t_2 : T_{12}}\ \text{T-App}$$

```
\begin{InfRule}{T-App}[r]
  \hypo{\Gamma \vdash t_1 : T_{11} \to T_{12}}
  \hypo{\Gamma \vdash t_2 : T_2}
  \infer2{\Gamma \vdash t_1\ t_2 : T_{12}}
\end{InfRule}
```

## 3 Statements

Here we only explain the extended statement `\infer`. Consult the documentation of ebproof for other statements.

`\infer`

```
\infer[⟨options⟩]{⟨arities⟩}[⟨label⟩]{⟨text⟩}
```

The statement `\infer` builds an inference step by taking some proof trees from the top of the stack, assembling them with a rule joining their conclusions and putting a new conclusion below. The ⟨*arities*⟩ are the comma-separated list of sub-proofs, it may be a list of any number including 0 (in this case there will be a line above the conclusion but no sub-proof). A single number is also accepted, as the original `\infer` command from the ebproof package. If ⟨*label*⟩ is present, it is used as the label on the right of the inference line; it is equivalent to using the `right label` option.[4]

$$\dfrac{\dfrac{\Gamma \vdash 1 \quad \Gamma \vdash 2 \quad \Gamma \vdash 3}{\Gamma \vdash 4}\ \dfrac{\Gamma \vdash 5 \quad \Gamma \vdash 6}{\Gamma \vdash 7}\ \dfrac{\Gamma \vdash 8 \quad \Gamma \vdash 9}{\Gamma \vdash 56789}}{\Gamma \vdash 1234} \qquad \Gamma \vdash 10}{\Gamma \vdash 12345678910}$$

```
\begin{prooftree}
  \hypo{\Gamma \vdash 1}
  \hypo{\Gamma \vdash 2}
  \hypo{\Gamma \vdash 3}
  \hypo{\Gamma \vdash 4}
  \infer{3,1}{\Gamma \vdash 1234}
  \hypo{\Gamma \vdash 5}
  \hypo{\Gamma \vdash 6}
  \hypo{\Gamma \vdash 7}
  \hypo{\Gamma \vdash 8}
  \hypo{\Gamma \vdash 9}
  \infer{2,1,2}{\Gamma \vdash 56789}
  \hypo{\Gamma \vdash 10}
  \infer{2,1}{\Gamma \vdash 12345678910}
\end{prooftree}
```

[4] This paragraph is largely taken from the original ebproof package documentation word-by-word.

# A Implementation

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}
\RequirePackage{expl3}
\RequirePackage{xparse}
\RequirePackage{relsize}
\RequirePackage{tabularray}
\ProvidesExplPackage{ebproofx}{2024/01/10}{1.0.0}{ebproof eXtended}
⟨@@=ebproofx⟩
```

## A.1 Parameters

We first declare all options.

```
\bool_new:N \l__ebproofx_updown_bool
\keys_define:nn { ebproofx } {
center .bool_set:N = \l__ebproofx_center_bool,
proof~style .choice: ,
proof~style / upwards .code:n = \bool_set_false:N \l__ebproofx_uprown_bool,
proof~style / downwards .code:n = \bool_set_true:N \l__ebproofx_updown_bool,
separation .dim_set:N = \l__ebproofx_separation_dim,
rule~margin .dim_set:N = \l__ebproofx_rule_margin_dim,
rule~thickness .dim_set:N = \l__ebproofx_rule_thickness_dim,
rule~separation .dim_set:N = \l__ebproofx_rule_separation_dim,
rule~dash~length .dim_set:N = \l__ebproofx_rule_dash_length_dim,
rule~dash~space .dim_set:N = \l__ebproofx_rule_dash_space_dim,
rule~code .tl_set:N = \l__ebproofx_rule_code_tl,
rule~style .choice:,
template .tl_set:N = \l__ebproofx_template_tl,
left~template .tl_set:N = \l__ebproofx_left_template_tl,
right~template .tl_set:N = \l__ebproofx_right_template_tl,
left~label .tl_set:N = \l__ebproofx_left_label_tl,
right~label .tl_set:N = \l__ebproofx_right_label_tl,
left~label~template .tl_set:N = \l__ebproofx_left_label_template_tl,
right~label~template .tl_set:N = \l__ebproofx_right_label_template_tl,
label~separation .dim_set:N = \l__ebproofx_label_separation_dim,
label~axis .dim_set:N = \l__ebproofx_label_axis_dim,
}
```

`\ebproofxnewrulestyle` We then define the document-level macro `\ebproofxnewrulestyle` and use it to define the default styles. This simply consists in defining a meta-key.

```
\NewDocumentCommand \ebproofxnewrulestyle { mm } {
  \keys_define:nn { ebproofx } {
    rule~style / #1 .meta:nn = { ebproofx } { #2 }
  }
}
```

(*End of definition for* `\ebproofxnewrulestyle`. *This function is documented on page* **??**.)

The styles `simple`, `no rule` and `double` are defined in a straightforward way.

```
\ebproofxnewrulestyle { simple } {
  rule~code = { \tex_hrule:D height \l__ebproofx_rule_thickness_dim }
}
\ebproofxnewrulestyle { no~rule } {
  rule~code =
```

```
}
\ebproofxnewrulestyle { double } {
  rule~code = {
    \tex_hrule:D height \l__ebproofx_rule_thickness_dim
    \skip_vertical:N \l__ebproofx_rule_separation_dim
    \tex_hrule:D height \l__ebproofx_rule_thickness_dim
  }
}
```

The dashed style uses leaders and filling for repeating a single dash. We use TEX primitives that have no LATEX3 counterpart for this.

```
\ebproofxnewrulestyle { dashed } {
  rule~code = {
    \hbox_to_wd:nn { \tex_hsize:D } {
      \dim_set:Nn \l_tmpa_dim { \l__ebproofx_rule_dash_space_dim / 2 }
      \skip_horizontal:n { -\l_tmpa_dim }
      \tex_cleaders:D \hbox:n {
        \skip_horizontal:N \l_tmpa_dim
        \tex_vrule:D
          height \l__ebproofx_rule_thickness_dim
          width \l__ebproofx_rule_dash_length_dim
        \skip_horizontal:N \l_tmpa_dim
      } \tex_hfill:D
      \skip_horizontal:n { -\l_tmpa_dim }
    }
  }
}
```

Now we can define the default values, including the default rule style.

```
\keys_set:nn { ebproofx } {
  center = true,
  proof~style = upwards,
  separation = 1.5em,
  rule~margin = .7ex,
  rule~thickness = .4pt,
  rule~separation = 2pt,
  rule~dash~length = .2em,
  rule~dash~space = .3em,
  rule~style = simple,
  template = $\inserttext$,
  left~template = $\inserttext\mathrel{}$,
  right~template = $\mathrel{}\inserttext$,
  left~label = ,
  right~label = ,
  left~label~template = \inserttext,
  right~label~template = \inserttext,
  label~separation = 0.5em,
  label~axis = 0.5ex
}
```

`\ebproofxnewstyle` Defining a style simply means defining a meta-key.

```
\NewDocumentCommand \ebproofxnewstyle { mm } {
  \keys_define:nn { ebproofx } { #1 .meta:n = { #2 } }
}
```

(*End of definition for* `\ebproofxnewstyle`. *This function is documented on page* **??**.)

## A.2 Proof boxes

TEX does not actually provide data structures, so we have to encode things. We provide an allocator for "registers" holding boxes with attributes. Such a register consists in a box register and a property list for marks, which maps mark names to values as explicit dimensions with units.

`\__ebproofx_new:N` Using only public interfaces forces a convoluted approach to allocation: we use a global counter `\g_ebproofx_register_int` to number registers, then each allocation creates registers named `\S_ebproofx_K_N` where S is the scope of the register (local or global, deduced from the argument), K is the kind of component (box or marks) and N is the identifier of the register. The proof box register itself only contains the identifier used for indirection.

```
\int_new:N \g__ebproofx_register_int
\cs_new:Nn \__ebproofx_box:N {
  \str_item:nn { #1 } { 2 } __ebproofx_ \tl_use:N #1 _box
}
\cs_new:Nn \__ebproofx_marks:N {
  \str_item:nn { #1 } { 2 } __ebproofx_ \tl_use:N #1 _prop
}
\cs_new:Nn \__ebproofx_new:N {
  \tl_new:N #1
  \int_gincr:N \g__ebproofx_register_int
  \str_if_eq:eeTF { \str_item:nn { #1 } { 2 } } { g }
    { \tl_gset:Nx #1 { \int_to_arabic:n { \g__ebproofx_register_int } } }
    { \tl_set:Nx #1 { \int_to_arabic:n { \g__ebproofx_register_int } } }
  \box_new:c { \__ebproofx_box:N #1 }
  \prop_new:c { \__ebproofx_marks:N #1 }
}
```

(*End of definition for* `\__ebproofx_new:N`.)

`\__ebproofx_clear:N` The box is cleared by setting it to an empty hbox. Using `\box_clear:N` instead would not work because trying to push this box on the stack would not actually append any box.

```
\cs_new:Nn \__ebproofx_clear:N {
  \hbox_set:cn { \__ebproofx_box:N #1 } {}
  \prop_clear:c { \__ebproofx_marks:N #1 }
  \__ebproofx_set_mark:Nnn #1 { left } { 0pt }
  \__ebproofx_set_mark:Nnn #1 { right } { 0pt }
  \__ebproofx_set_mark:Nnn #1 { axis } { 0pt }
}
```

(*End of definition for* `\__ebproofx_clear:N`.)

### A.2.1 Mark operations

`\__ebproofx_set_mark:Nnn` Setting the value of a mark uses a temporary register to evaluate the dimension expression because values are stored textually in a property list.

```
\dim_new:N \l__ebproofx_transit_dim
\cs_new:Nn \__ebproofx_set_mark:Nnn {
  \dim_set:Nn \l__ebproofx_transit_dim { #3 }
  \prop_put:cnV { \__ebproofx_marks:N #1 } { #2 }
```

```
    \l__ebproofx_transit_dim
}
```

(*End of definition for* `\__ebproofx_set_mark:Nnn`.)

`\__ebproofx_mark:Nn` Getting the value of a mark simply consists in getting an item in a property list.

```
\cs_new:Nn \__ebproofx_mark:Nn {
  \prop_item:cn { \__ebproofx_marks:N #1 } { #2 }
}
```

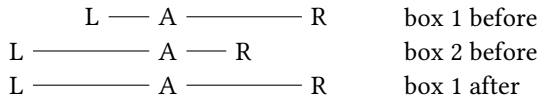
(*End of definition for* `\__ebproofx_mark:Nn`.)

`\__ebproofx_shift_x:Nn` This function shifts the marks by a specified amount, without modifying the box.

```
\cs_new:Nn \__ebproofx_shift_x:Nn {
  \prop_map_inline:cn { \__ebproofx_marks:N #1 } {
    \__ebproofx_set_mark:Nnn #1 { ##1 } { ##2 + #2 }
  }
}
```

(*End of definition for* `\__ebproofx_shift_x:Nn`.)

`\__ebproofx_enlarge_conclusion:NN` This function moves the left and right marks of the first tree so that they are at least as far from the axis as they are in the second tree. For instance we get the following:




The contents of the trees are unchanged.

```
\cs_new:Nn \__ebproofx_enlarge_conclusion:NN {
  \dim_set:Nn \l_tmpa_dim { \__ebproofx_mark:Nn #1 {axis}
    + \__ebproofx_mark:Nn #2 {left} - \__ebproofx_mark:Nn #2 {axis} }
  \dim_compare:nNnT { \l_tmpa_dim } < { \__ebproofx_mark:Nn #1 {left} } {
    \__ebproofx_set_mark:Nnn #1 {left} { \l_tmpa_dim } }
  \dim_set:Nn \l_tmpa_dim { \__ebproofx_mark:Nn #1 {axis}
    + \__ebproofx_mark:Nn #2 {right} - \__ebproofx_mark:Nn #2 {axis} }
  \dim_compare:nNnT { \l_tmpa_dim } > { \__ebproofx_mark:Nn #1 {right} } {
    \__ebproofx_set_mark:Nnn #1 {right} { \l_tmpa_dim } }
}
```

(*End of definition for* `\__ebproofx_enlarge_conclusion:NN`.)

### A.2.2 Building blocks

`\__ebproofx_make_simple:Nn` Make a tree with explicit material in horizontal mode. Set the left and right marks to extremal positions and set the axis in the middle.

```
\cs_new:Nn \__ebproofx_make_simple:Nn {
  \hbox_set:cn { \__ebproofx_box:N #1 } { #2 }
  \__ebproofx_set_mark:Nnn #1 { left } { 0pt }
  \__ebproofx_set_mark:Nnn #1 { axis } { \box_wd:c { \__ebproofx_box:N #1 } / 2 }
  \__ebproofx_set_mark:Nnn #1 { right } { \box_wd:c { \__ebproofx_box:N #1 } }
}
```

(*End of definition for* `\__ebproofx_make_simple:Nn`.)

__ebproofx_make_split:Nnn Make a tree with explicit material in horizontal mode, split in two parts. Set the left and right marks to extremal positions and set the axis between the two parts.

```
\cs_new:Nn \__ebproofx_make_split:Nnn {
  \__ebproofx_set_mark:Nnn #1 { left } { 0pt }
  \hbox_set:cn { \__ebproofx_box:N #1 } { #2 }
  \__ebproofx_set_mark:Nnn #1 { axis } { \box_wd:c { \__ebproofx_box:N #1 } }
  \hbox_set:cn { \__ebproofx_box:N #1 } { \hbox_unpack:c { \__ebproofx_box:N #1 } #3 }
  \__ebproofx_set_mark:Nnn #1 { right } { \box_wd:c { \__ebproofx_box:N #1 } }
}
```

(*End of definition for* __ebproofx_make_split:Nnn.)

__ebproofx_make_vertical:Nnnn Make a tree with explicit material in vertical mode, using an explicit width and axis.

```
\cs_new:Nn \__ebproofx_make_vertical:Nnnn {
  \__ebproofx_set_mark:Nnn #1 { left } { 0pt }
  \__ebproofx_set_mark:Nnn #1 { axis } { #2 }
  \__ebproofx_set_mark:Nnn #1 { right } { #3 }
  \vbox_set:cn { \__ebproofx_box:N #1 } {
    \dim_set:Nn \tex_hsize:D { \__ebproofx_mark:Nn #1 {right} }
    #4
  }
  \box_set_wd:cn { \__ebproofx_box:N #1 } { \__ebproofx_mark:Nn #1 {right} }
}
```

(*End of definition for* __ebproofx_make_vertical:Nnnn.)

### A.2.3 Assembling boxes

__ebproofx_extend:Nnnnn Extend a tree box. The marks are shifted so that alignment is preserved. The arguments are dimensions for the left, top, right and bottom sides respectively.

```
\cs_new:Nn \__ebproofx_extend:Nnnnn {
  \dim_compare:nNnF { #2 } = { 0pt } {
    \hbox_set:cn { \__ebproofx_box:N #1 } {
      \skip_horizontal:n { #2 }
      \box_use:c { \__ebproofx_box:N #1 }
    }
    \__ebproofx_shift_x:Nn #1 { #2 }
  }
  \box_set_ht:Nn #1 { \box_ht:c { \__ebproofx_box:N #1 } + #3 }
  \box_set_wd:Nn #1 { \box_wd:c { \__ebproofx_box:N #1 } + #4 }
  \box_set_dp:Nn #1 { \box_dp:c { \__ebproofx_box:N #1 } + #5 }
}
```

(*End of definition for* __ebproofx_extend:Nnnnn.)

__ebproofx_append_right:NnN Append the contents of the second tree to the first one on the right, with matching baselines. The marks of both trees are preserved. The middle argument specifies the space to insert between boxes.

```
\cs_new:Nn \__ebproofx_append_right:NnN {
  \hbox_set:cn { \__ebproofx_box:N #1 } {
    \box_use:c { \__ebproofx_box:N #1 }
    \dim_compare:nNnF { #2 } = { 0pt } { \skip_horizontal:n { #2 } }
    \box_use:c { \__ebproofx_box:N #3 }
  }
}
```

(*End of definition for* __ebproofx_append_right:NnN.)

__ebproofx_append_left:NnN Append the contents of the second tree to the first one on the left, with matching baselines. The marks of the first tree are shifted accordingly. The middle argument specifies the space to insert between boxes.

```
\cs_new:Nn \__ebproofx_append_left:NnN {
  \__ebproofx_shift_x:Nn #1 { \box_wd:c { \__ebproofx_box:N #3 } + #2 }
  \hbox_set:cn { \__ebproofx_box:N #1 } {
    \box_use:c { \__ebproofx_box:N #3 }
    \dim_compare:nNnF { #2 } = { 0pt } { \skip_horizontal:n { #2 } }
    \box_use:c { \__ebproofx_box:N #1 }
  }
}
```

(*End of definition for* __ebproofx_append_left:NnN.)

__ebproofx_align:NN Shift one of two trees to the right so that their axes match. The marks of the one that is shifted are updated accordingly.

```
\cs_new:Nn \__ebproofx_align:NN {
  \dim_set:Nn \l_tmpa_dim
    { \__ebproofx_mark:Nn #2 {axis} - \__ebproofx_mark:Nn #1 {axis} }
  \dim_compare:nNnTF \l_tmpa_dim < { 0pt } {
    \__ebproofx_extend:Nnnnn #2 { -\l_tmpa_dim } { 0pt } { 0pt } { 0pt }
  } {
    \__ebproofx_extend:Nnnnn #1 { \l_tmpa_dim } { 0pt } { 0pt } { 0pt }
  }
}
```

(*End of definition for* __ebproofx_align:NN.)

__ebproofx_append_above:NN Append the contents of the second tree above the first one, with matching axes. The marks of the first tree are preserved.

```
\cs_new:Nn \__ebproofx_append_above:NN {
  \__ebproofx_align:NN #1 #2
  \vbox_set:cn  { \__ebproofx_box:N #1 } {
    \box_use:c { \__ebproofx_box:N #2 }
    \tex_prevdepth:D -1000pt
    \box_use:c { \__ebproofx_box:N #1 }
  }
}
```

(*End of definition for* __ebproofx_append_above:NN.)

__ebproofx_append_below:NN Append the contents of the second tree below the first one, with matching axes. The marks of the first tree are preserved.

```
\cs_new:Nn \__ebproofx_append_below:NN {
  \__ebproofx_align:NN #1 #2
  \vbox_set_top:cn { \__ebproofx_box:N #1 } {
    \box_use:c { \__ebproofx_box:N #1 }
    \tex_prevdepth:D -1000pt
    \box_use:c { \__ebproofx_box:N #2 }
  }
}
```

(*End of definition for* __ebproofx_append_below:NN.)

__ebproofx_overlay:NN Append the second tree as an overlay over the first one, so that the baselines and axes match. The bounding box of the result adjusts to contain both trees.

```
\cs_new:Nn \__ebproofx_overlay:NN {
  \__ebproofx_align:NN #1 #2
  \hbox_set:cn { \__ebproofx_box:N #1 } {
    \hbox_overlap_right:n { \box_use:c { \__ebproofx_box:N #1 } }
    \box_use:c { \__ebproofx_box:N #2 }
    \dim_compare:nNnT
     { \box_wd:c { \__ebproofx_box:N #2 } } < { \box_wd:c { \__ebproofx_box:N #1 } }
     { \skip_horizontal:n
       { \box_wd:c { \__ebproofx_box:N #1 } - \box_wd:c { \__ebproofx_box:N #2 } } }
  }
}
```

(*End of definition for* __ebproofx_overlay:NN.)

__ebproofx_vcenter:N Shift the material in a tree vertically so that the height and depth are equal (like TEX's \vcenter but around the baseline).

```
\cs_new:Nn \__ebproofx_vcenter:N {
  \dim_set:Nn \l_tmpa_dim
   { ( \box_ht:c { \__ebproofx_box:N #1 } - \box_dp:c { \__ebproofx_box:N #1 } ) / 2 }
  \box_set_eq:Nc \l_tmpa_box { \__ebproofx_box:N #1 }
  \hbox_set:cn { \__ebproofx_box:N #1 }
    { \box_move_down:nn { \l_tmpa_dim } { \box_use:N \l_tmpa_box } }
}
```

(*End of definition for* __ebproofx_vcenter:N.)

## A.3 Making inferences

The following commands use the parameters defined at the beginning of the package for actually building proof trees using the commands defined above.

__ebproofx_append_vertical:NN Append the contents of the second tree above or below the first one, depending on current settings. Axes are aligned and the marks of the first tree are preserved.

```
\cs_new:Nn \__ebproofx_append_vertical:NN {
  \bool_if:NTF \l__ebproofx_updown_bool
    { \__ebproofx_append_below:NN #1 #2 }
    { \__ebproofx_append_above:NN #1 #2 }
}
```

(*End of definition for* __ebproofx_append_vertical:NN.)

__ebproofx_make_rule_for:NNN Make a box containing an inference rule with labels, using the current settings. The width and axis position are taken as those of the conclusion of another tree box. The third argument is used as a temporary register for building labels.

```
\cs_new:Nn \__ebproofx_make_rule_for:NNN {
```

Build the rule.

```
  \__ebproofx_make_vertical:Nnnn #1
    { \__ebproofx_mark:Nn #2 {axis} - \__ebproofx_mark:Nn #2 {left} }
    { \__ebproofx_mark:Nn #2 {right} - \__ebproofx_mark:Nn #2 {left} }
    {
      \skip_vertical:N \l__ebproofx_rule_margin_dim
```

```
      \tl_if_empty:NF { \l__ebproofx_rule_code_tl } {
        \tl_use:N \l__ebproofx_rule_code_tl
        \skip_vertical:N \l__ebproofx_rule_margin_dim
      }
    }
  \__ebproofx_vcenter:N #1
```

Append the left label.

```
  \tl_if_blank:VF \l__ebproofx_left_label_tl {
    \__ebproofx_make_simple:Nn #3 {
      \box_move_down:nn { \l__ebproofx_label_axis_dim } { \hbox:n {
        \cs_set_eq:NN \inserttext \l__ebproofx_left_label_tl
        \tl_use:N \l__ebproofx_left_label_template_tl
      } }
    }
    \box_set_ht:cn { \__ebproofx_box:N #3 } { 0pt }
    \box_set_dp:cn { \__ebproofx_box:N #3 } { 0pt }
    \__ebproofx_append_left:NnN
      \l__ebproofx_c_box \l__ebproofx_label_separation_dim \l__ebproofx_d_box
  }
```

Append the right label.

```
  \tl_if_blank:VF \l__ebproofx_right_label_tl {
    \__ebproofx_make_simple:Nn #3 {
      \box_move_down:nn { \l__ebproofx_label_axis_dim } { \hbox:n {
        \cs_set_eq:NN \inserttext \l__ebproofx_right_label_tl
        \tl_use:N \l__ebproofx_right_label_template_tl
      } }
    }
    \box_set_ht:cn { \__ebproofx_box:N #3 } { 0pt }
    \box_set_dp:cn { \__ebproofx_box:N #3 } { 0pt }
    \__ebproofx_append_right:NnN
      \l__ebproofx_c_box \l__ebproofx_label_separation_dim \l__ebproofx_d_box
  }
}
```

(*End of definition for* `\__ebproofx_make_rule_for:NNN`.)

## A.4 Stack-based interface

### A.4.1 The stack

Logically, box structures are stored on a stack. However, TEX does not provide data structures for that and the grouping mechanism is not flexible enough, so we encode them using what we actually have. A stack for boxes is implemented using a global hbox `\g__ebproofx_stack_box` that contains all the boxes successively. A sequence `\g__ebproofx_stack_seq` is used to store the dimensions property lists textually. We maintain a counter `\g__ebproofx_level_int` with the number of elements on the stack, for consistency checks.

```
\int_new:N \g__ebproofx_level_int
\box_new:N \g__ebproofx_stack_box
\seq_new:N \g__ebproofx_stack_seq
```

`\__ebproofx_clear_stack:` Clear the stack.

```
\cs_new:Nn \__ebproofx_clear_stack: {
```

```
  \int_gset:Nn \g__ebproofx_level_int { 0 }
  \hbox_gset:Nn \g__ebproofx_stack_box { }
  \seq_gclear:N \g__ebproofx_stack_seq
}
```

(*End of definition for* `\__ebproofx_clear_stack:`.)

`\__ebproofx_push:N` Push the contents of a register on the stack.

```
\cs_new:Nn \__ebproofx_push:N {
  \int_gincr:N \g__ebproofx_level_int
  \hbox_gset:Nn \g__ebproofx_stack_box
    { \hbox_unpack:N \g__ebproofx_stack_box \box_use:c { \__ebproofx_box:N #1 } }
  \seq_gput_left:Nv \g__ebproofx_stack_seq
    { \__ebproofx_marks:N #1 }
}
```

(*End of definition for* `\__ebproofx_push:N`.)

`\__ebproofx_pop:N` Pop the value from the top of the stack into a register.

```
\cs_new:Nn \__ebproofx_pop:N {
  \int_compare:nNnTF { \g__ebproofx_level_int } > { 0 } {
    \int_gdecr:N \g__ebproofx_level_int
    \hbox_gset:Nn \g__ebproofx_stack_box {
      \hbox_unpack:N \g__ebproofx_stack_box
      \box_gset_to_last:N \g_tmpa_box
    }
    \box_set_eq_drop:cN { \__ebproofx_box:N #1 } \g_tmpa_box
    \seq_gpop_left:NN \g__ebproofx_stack_seq \l_tmpa_tl
    \tl_set_eq:cN { \__ebproofx_marks:N #1 } \l_tmpa_tl
  } {
    \PackageError{ebproofx}{Missing~premiss~in~a~proof~tree}{}
    \__ebproofx_clear:N #1
  }
}
```

(*End of definition for* `\__ebproofx_pop:N`.)

### A.4.2 Assembling trees

```
\__ebproofx_new:N \l__ebproofx_a_box
\__ebproofx_new:N \l__ebproofx_b_box
\__ebproofx_new:N \l__ebproofx_c_box
\__ebproofx_new:N \l__ebproofx_d_box
```

`\__ebproofx_join_horizontal:n` Join horizontally a number of elements at the top of the stack. If several trees are joined, use the left mark of the left tree, the right mark of the right tree and set the axis in the middle of these marks.

```
\cs_new:Nn \__ebproofx_join_horizontal:n {
  \int_case:nnF { #1 } {
  { 0 } {
    \group_begin:
    \__ebproofx_clear:N \l__ebproofx_a_box
    \__ebproofx_push:N \l__ebproofx_a_box
    \group_end:
  }
```

```
  { 1 } { }
  } {
    \group_begin:
    \__ebproofx_pop:N \l__ebproofx_a_box
    \prg_replicate:nn { #1 - 1 } {
      \__ebproofx_pop:N \l__ebproofx_b_box
      \__ebproofx_append_left:NnN
        \l__ebproofx_a_box \l__ebproofx_separation_dim \l__ebproofx_b_box
    }
    \__ebproofx_set_mark:Nnn \l__ebproofx_a_box { left }
      { \__ebproofx_mark:Nn \l__ebproofx_b_box { left } }
    \__ebproofx_set_mark:Nnn \l__ebproofx_a_box { axis }
      { ( \__ebproofx_mark:Nn \l__ebproofx_a_box { left }
        + \__ebproofx_mark:Nn \l__ebproofx_a_box { right } ) / 2 }
    \__ebproofx_push:N \l__ebproofx_a_box
    \group_end:
  }
}
```

(*End of definition for* `\__ebproofx_join_horizontal:n`.)

`\__ebproofx_join_vertical:` Join vertically the two elements at the top of the stack, with a horizontal rule of the appropriate size.

```
\cs_new:Nn \__ebproofx_join_vertical: {
  \group_begin:
  \__ebproofx_pop:N \l__ebproofx_a_box
  \__ebproofx_pop:N \l__ebproofx_b_box
  \__ebproofx_enlarge_conclusion:NN \l__ebproofx_b_box \l__ebproofx_a_box
  \__ebproofx_make_rule_for:NNN \l__ebproofx_c_box \l__ebproofx_b_box
    \l__ebproofx_d_box
  \__ebproofx_append_vertical:NN \l__ebproofx_a_box \l__ebproofx_c_box
  \__ebproofx_append_vertical:NN \l__ebproofx_a_box \l__ebproofx_b_box
  \__ebproofx_push:N \l__ebproofx_a_box
  \group_end:
}
```

(*End of definition for* `\__ebproofx_join_vertical:`.)

### A.4.3 High-level commands

`\__ebproofx_statement_parse:w` An auxiliary function for parsing the argument in `\__ebproofx_push_statement:n`.

```
\cs_new:Npn \__ebproofx_statement_parse:w #1 & #2 & #3 \q_stop {
  \tl_if_empty:nTF { #3 } {
    \__ebproofx_make_simple:Nn \l__ebproofx_a_box
      { \cs_set:Npn \inserttext { #1 } \tl_use:N \l__ebproofx_template_tl }
  } {
    \__ebproofx_make_split:Nnn \l__ebproofx_a_box
      { \cs_set:Npn \inserttext { #1 } \tl_use:N \l__ebproofx_left_template_tl }
      { \cs_set:Npn \inserttext { #2 } \tl_use:N \l__ebproofx_right_template_tl }
  }
  \__ebproofx_push:N \l__ebproofx_a_box
}
```

(*End of definition for* `\__ebproofx_statement_parse:w`.)

__ebproofx_push_statement:n Push a box with default formatting, using explicit alignment if the code contains a & character

```
\cs_new:Nn \__ebproofx_push_statement:n {
  \__ebproofx_statement_parse:w #1 & & \q_stop
}
```

(*End of definition for* __ebproofx_push_statement:n.)

## A.5 Document interface

### A.5.1 Functions to define statements

The \g__ebproofx_statements_seq variable contains the list of all defined statements. For each statement X, there is a document command \ebproofxX and the alias \X is defined when entering a prooftree environment.

```
\seq_new:N \g__ebproofx_statements_seq
```

__ebproofx_setup_statements: Install the aliases for statements, saving the original value of the control sequences.

```
\cs_new:Nn \__ebproofx_setup_statements: {
  \seq_map_inline:Nn \g__ebproofx_statements_seq {
    \cs_set_eq:cc { ebproofx_saved_ ##1 } { ##1 }
    \cs_set_eq:cc { ##1 } { ebproofx ##1 }
  }
}
```

(*End of definition for* __ebproofx_setup_statements:.)

__ebproofx_restore_statements: Restore the saved meanings of the control sequences. This is useful when interpreting user-provided code in statement arguments. The meanings are automatically restored when leaving a prooftree environment because of grouping.

```
\cs_new:Nn \__ebproofx_restore_statements: {
  \seq_map_inline:Nn \g__ebproofx_statements_seq {
    \cs_set_eq:cc { ##1 } { ebproofx_saved_ ##1 }
  }
}
```

(*End of definition for* __ebproofx_restore_statements:.)

__ebproofx_new_statement:nnn Define a new statement. The first argument is the name, the second one is an argument specifier as used by xparse and the third one is the body of the command.

```
\cs_new:Nn \__ebproofx_new_statement:nnn {
  \exp_args:Nc \NewDocumentCommand { ebproofx#1 }{ #2 } { #3 }
  \seq_gput_right:Nn \g__ebproofx_statements_seq { #1 }
}
```

(*End of definition for* __ebproofx_new_statement:nnn.)

__ebproofx_new_deprecated_statement:nnnn Define a deprecated statement. The syntax is the same as above except that an extra argument in third position indicates what should be used instead. The effect is the same except that a warning message is issued the first time the statement is used.

```
\cs_new:Nn \__ebproofx_new_deprecated_statement:nnnn {
  \cs_new:cpn { ebproofx_#1_warning: } {
    \PackageWarning { ebproofx } { \token_to_str:c{#1}~is~deprecated,~#3 }
    \cs_gset:cn { ebproofx_#1_warning: } { }
  }
```

```
  \__ebproofx_new_statement:nnn { #1 } { #2 }
    { \use:c { ebproofx_#1_warning: } #4 }
}
```

(*End of definition for* `\__ebproofx_new_deprecated_statement:nnnn`.)

### A.5.2 Basic commands

`\ebproofxset` `\set` This is a simple wrapper around `\keys_set:nn`.

```
\__ebproofx_new_statement:nnn { set } { m } {
  \keys_set:nn { ebproofx } { #1 }
}
```

(*End of definition for* `\ebproofxset` *and* `\set`. *These functions are documented on page* ??.)

`\hypo` This is mostly a wrapper around `\ebproofx_push_statement:n`, with material to handle options and the statements macros.

```
\__ebproofx_new_statement:nnn { hypo } { O{} m } {
  \group_begin:
  \__ebproofx_restore_statements:
  \keys_set:nn { ebproofx } { #1 }
  \__ebproofx_push_statement:n { #2 }
  \group_end:
}
```

(*End of definition for* `\hypo`. *This function is documented on page* ??.)

`\infer` This is a bit more involved than `\hypo` because we have to handle rule style options and joining.

```
\__ebproofx_new_statement:nnn { infer } { O{} m O{} m } {
  \group_begin:
  \__ebproofx_restore_statements:
  \keys_set_known:nnN { ebproofx / rule~style } { #1 } \l_tmpa_tl
  \keys_set:nV { ebproofx } \l_tmpa_tl
  \tl_set:Nn \l__ebproofx_right_label_tl { #3 }
  \__ebproofx_join_horizontal:n { #2 }
  \__ebproofx_push_statement:n { #4 }
  \__ebproofx_join_vertical:
  \group_end:
}
```

(*End of definition for* `\infer`. *This function is documented on page* 2.)

`\ellipsis` An ellipsis is made by hand using vertical leaders to render the dots after rendering the label.

```
\__ebproofx_new_statement:nnn { ellipsis } { m m } {
  \group_begin:
  \__ebproofx_restore_statements:
  \tl_clear:N \l__ebproofx_rule_code_tl
  \__ebproofx_make_split:Nnn \l__ebproofx_a_box { } {
    \vbox_set:Nn \l_tmpa_box {
      \skip_vertical:n { 1.2ex }
      \hbox:n { \tex_ignorespaces:D #1 }
      \skip_vertical:n { 1.2ex }
    }
    \vbox_to_ht:nn { \box_ht:N \l_tmpa_box } {
      \tex_xleaders:D \vbox_to_ht:nn { .8ex }
```

```
        { \tex_vss:D \hbox:n { . } \tex_vss:D }
      \tex_vfill:D
    }
    \hbox_overlap_right:n { ~ \box_use:N \l_tmpa_box }
  }
  \__ebproofx_push:N \l__ebproofx_a_box
  \__ebproofx_join_vertical:
  \__ebproofx_push_statement:n {#2}
  \__ebproofx_join_vertical:
  \group_end:
}
```

(*End of definition for* `\ellipsis`. *This function is documented on page* **??**.)

### A.5.3 Modifying trees

`\rewrite` Rewrite the box at the top of the stack while preserving its dimensions an marks. The code is typeset in horizontal mode, with control sequences to access the original box and its marks.

```
\__ebproofx_new_statement:nnn { rewrite } { m } {
  \group_begin:
  \__ebproofx_restore_statements:
  \__ebproofx_pop:N \l__ebproofx_a_box
  \box_set_eq:Nc \l_tmpa_box { \__ebproofx_box:N \l__ebproofx_a_box }
  \hbox_set:Nn \l_tmpb_box {
    \cs_set_eq:NN \treebox \l_tmpa_box
    \cs_set:Npn \treemark { \__ebproofx_mark:Nn \l__ebproofx_a_box }
    { #1 }
  }
  \box_set_wd:Nn \l_tmpb_box { \box_wd:c { \__ebproofx_box:N \l__ebproofx_a_box } }
  \box_set_ht:Nn \l_tmpb_box { \box_ht:c { \__ebproofx_box:N \l__ebproofx_a_box } }
  \box_set_dp:Nn \l_tmpb_box { \box_dp:c { \__ebproofx_box:N \l__ebproofx_a_box } }
  \box_set_eq:cN { \__ebproofx_box:N \l__ebproofx_a_box } \l_tmpb_box
  \__ebproofx_push:N \l__ebproofx_a_box
  \group_end:
}
```

(*End of definition for* `\rewrite`. *This function is documented on page* **??**.)

`\delims` Insert `\left` and `\right` delimiters without changing the alignment.

```
\__ebproofx_new_statement:nnn { delims } { m m } {
  \group_begin:
  \__ebproofx_restore_statements:
  \__ebproofx_pop:N \l__ebproofx_a_box
  \hbox_set:Nn \l_tmpa_box
    { $ \tex_vcenter:D { \box_use:c { \__ebproofx_box:N \l__ebproofx_a_box } } $ }
  \dim_set:Nn \l_tmpa_dim
    { \box_ht:N \l_tmpa_box - \box_ht:c { \__ebproofx_box:N \l__ebproofx_a_box } }
  \hbox_set:cn { \__ebproofx_box:N \l__ebproofx_a_box } {
    $ #1 \tex_vrule:D
      height \box_ht:N \l_tmpa_box depth \box_dp:N \l_tmpa_box width 0pt
    \tex_right:D . $
  }
  \__ebproofx_shift_x:Nn \l__ebproofx_a_box
    { \box_wd:c { \__ebproofx_box:N \l__ebproofx_a_box } }
```

```
  \hbox_set:cn { \__ebproofx_box:N \l__ebproofx_a_box } {
    \hbox_unpack:c { \__ebproofx_box:N \l__ebproofx_a_box }
    $ \tex_left:D . \box_use:N \l_tmpa_box #2 $
  }
  \hbox_set:cn { \__ebproofx_box:N \l__ebproofx_a_box }
    { \box_move_down:nn { \l_tmpa_dim }
        { \box_use:c { \__ebproofx_box:N \l__ebproofx_a_box } } }
  \__ebproofx_push:N \l__ebproofx_a_box
  \group_end:
}
```

(*End of definition for* \delims. *This function is documented on page* ??.)

\overlay Pop two trees and append the second tree as an overlay over the first one, so that the baselines and axes match. The bounding box of the result adjusts to contain both trees.

```
\__ebproofx_new_statement:nnn { overlay } { } {
  \group_begin:
  \__ebproofx_pop:N \l__ebproofx_a_box
  \__ebproofx_pop:N \l__ebproofx_b_box
  \__ebproofx_overlay:NN \l__ebproofx_a_box \l__ebproofx_b_box
  \__ebproofx_push:N \l__ebproofx_a_box
  \group_end:
}
```

(*End of definition for* \overlay. *This function is documented on page* ??.)

### A.5.4 The extension

```
\int_new:N \g__ebproofx_sublevel_int
\box_new:N \g__ebproofx_substack_box
\seq_new:N \g__ebproofx_substack_seq

\cs_new:Nn \__ebproofx_clear_substack:
  {
    \int_gset:Nn \g__ebproofx_sublevel_int { 0 }
    \hbox_gset:Nn \g__ebproofx_substack_box { }
    \seq_gclear:N \g__ebproofx_substack_seq
  }

\cs_new:Nn \__ebproofx_subpush:N
  {
    \int_gincr:N \g__ebproofx_sublevel_int
    \hbox_gset:Nn \g__ebproofx_substack_box
     { \hbox_unpack:N \g__ebproofx_substack_box \box_use:c { \__ebproofx_box:N #1 } }
    \seq_gput_left:Nv \g__ebproofx_substack_seq
      { \__ebproofx_marks:N #1 }
  }

\cs_new:Nn \__ebproofx_subpop:N
  {
    \int_compare:nNnTF { \g__ebproofx_sublevel_int } > { 0 }
      {
        \int_gdecr:N \g__ebproofx_sublevel_int
        \hbox_gset:Nn \g__ebproofx_substack_box {
          \hbox_unpack:N \g__ebproofx_substack_box
```

```
          \box_gset_to_last:N \g_tmpa_box
        }
        \box_set_eq_drop:cN { \__ebproofx_box:N #1 } \g_tmpa_box
        \seq_gpop_left:NN \g__ebproofx_substack_seq \l_tmpa_tl
        \tl_set_eq:cN { \__ebproofx_marks:N #1 } \l_tmpa_tl
      }
    { \PackageError{ebproofx}{Missing~premiss~in~a~proof~tree}{} \__ebproofx_clear:N #1 }
  }

\cs_new:Nn \__ebproofx_append_subvertical:NN
  {
    \bool_if:NTF \l__ebproofx_updown_bool
      { \__ebproofx_append_above:NN #1 #2 }
      { \__ebproofx_append_below:NN #1 #2 }
  }

\cs_new:Nn \__ebproofx_join_subvertical:n
  {
    \group_begin:
    \__ebproofx_subpop:N \l__ebproofx_a_box
    \prg_replicate:nn { #1 - 1 }
      {
        \__ebproofx_subpop:N \l__ebproofx_b_box
        \__ebproofx_enlarge_conclusion:NN \l__ebproofx_b_box \l__ebproofx_a_box

        \__ebproofx_make_vertical:Nnnn \l__ebproofx_c_box
          {
            \__ebproofx_mark:Nn \l__ebproofx_b_box {axis}
            - \__ebproofx_mark:Nn \l__ebproofx_b_box {left}
          }
          {
            \__ebproofx_mark:Nn \l__ebproofx_b_box {right}
            - \__ebproofx_mark:Nn \l__ebproofx_b_box {left}
          }
          { \skip_vertical:N \l__ebproofx_rule_margin_dim }
        \__ebproofx_vcenter:N \l__ebproofx_b_box
        \__ebproofx_append_subvertical:NN \l__ebproofx_a_box \l__ebproofx_c_box

        \__ebproofx_append_subvertical:NN \l__ebproofx_a_box \l__ebproofx_b_box
      }
    \__ebproofx_push:N \l__ebproofx_a_box
    \group_end:
  }

\cs_new:Nn \__ebproofx_renew_statement:nnn
  {
    \exp_args:Nc \RenewDocumentCommand { ebproofx#1 }{ #2 } { #3 }
    \seq_gput_right:Nn \g__ebproofx_statements_seq { #1 }
  }

\cs_generate_variant:Nn \clist_map_inline:nn { xn }
\__ebproofx_renew_statement:nnn { infer } { O{} m O{} m }
  {
    \group_begin:
```

```
    \__ebproofx_restore_statements:
    \keys_set_known:nnN { ebproofx / rule~style } { #1 } \l_tmpa_tl
    \keys_set:nV { ebproofx } \l_tmpa_tl
    \tl_set:Nn \l__ebproofx_right_label_tl { #3 }

    \__ebproofx_clear_substack:
    \clist_map_inline:xn { \clist_reverse:n { #2 } }
      {
        \__ebproofx_join_horizontal:n { ##1 }
        \__ebproofx_pop:N \l__ebproofx_a_box
        \__ebproofx_subpush:N \l__ebproofx_a_box
      }
    \__ebproofx_join_subvertical:n { \clist_count:n { #2 } }

    \__ebproofx_push_statement:n { #4 }
    \__ebproofx_join_vertical:
    \group_end:
  }
% \end{macrocode}
%
%
% \subsubsection{Deprecated statements}
%
% These statements were defined in versions 1.x of the package, they are
% preserved for temporary upwards compatibility and will be removed in a
% future version.
%    \begin{macrocode}
\__ebproofx_new_deprecated_statement:nnnn { Alter } { m }
  { use~\token_to_str:c{rewrite}~instead } { \ebproofxrewrite{ #1 \box\treebox } }
\__ebproofx_new_deprecated_statement:nnnn { Delims } { }
  { use~\token_to_str:c{delims}~instead } { \ebproofxdelims }
\__ebproofx_new_deprecated_statement:nnnn { Ellipsis } { }
  { use~\token_to_str:c{ellipsis}~instead } { \ebproofxellipsis }
\__ebproofx_new_deprecated_statement:nnnn { Hypo } { }
  { use~\token_to_str:c{hypo}~instead } { \ebproofxhypo }
\__ebproofx_new_deprecated_statement:nnnn { Infer } { }
  { use~\token_to_str:c{infer}~instead } { \ebproofxinfer }
```

### A.5.5 Environment interface

The stack is initialised globally. The `prooftree` environment does not clear the stack, instead it saves the initial level in order to check that statements are properly balanced. This allows for nested uses of the environment, if it ever happens to be useful.

```
\__ebproofx_clear_stack:
\tl_new:N \l__ebproofx_start_level_tl
```

`prooftree` `prooftree*` The `prooftree` environment.

```
\NewDocumentEnvironment { prooftree } { s O{} } {
  \group_align_safe_begin:
  \keys_set_known:nnN { ebproofx / proof~style } { #2 } \l_tmpa_tl
  \keys_set:nV { ebproofx } \l_tmpa_tl
  \tl_set:Nx \l__ebproofx_start_level_tl { \int_use:N \g__ebproofx_level_int }
  \vbox_set:Nw \l_tmpa_box
  \__ebproofx_setup_statements:
```

```
} {
  \vbox_set_end:
  \__ebproofx_pop:N \l__ebproofx_a_box
 \int_compare:nNnF { \g__ebproofx_level_int } = { \tl_use:N \l__ebproofx_start_level_tl } {
    \PackageError{ebproofx}{Malformed~proof~tree}{
      Some~hypotheses~were~declared~but~not~used~in~this~tree.}
  }
  \IfBooleanTF { #1 } {
    \[ \box_use:c { \__ebproofx_box:N \l__ebproofx_a_box } \]
    \ignorespacesafterend
  } {
    \hbox_unpack:N \c_empty_box
    \bool_if:NTF \l__ebproofx_center_bool {
    \hbox:n { $ \tex_vcenter:D { \box_use:c { \__ebproofx_box:N \l__ebproofx_a_box } } $ }
    } {
      \box_use:c { \__ebproofx_box:N \l__ebproofx_a_box }
    }
  }
  \group_align_safe_end:
}
```

A trick for the starred version:

```
\cs_new:cpn { prooftree* } { \prooftree* }
\cs_new:cpn { endprooftree* } { \endprooftree }
```

(*End of definition for* `prooftree` *and* `prooftree*`. *These functions are documented on page* ??.)

`InfRule` The `InfRule` environment.

```
\NewTblrEnviron{@ruleenv}
\SetTblrInner[@ruleenv]{belowsep=0pt,stretch=0}
\SetTblrOuter[@ruleenv]{baseline=b}
\NewDocumentEnvironment { InfRule } { m O{l} +b }
  {
    \begin{@ruleenv}{#2}
      \smaller{\textsc{#1}} \\
      \begin{prooftree} #3 \end{prooftree}
    \end{@ruleenv}
  }
  {}
```

(*End of definition for* `InfRule`. *This function is documented on page 2.*)

⟨/package⟩