

# The `trfrac` package\*

Kevin W. Hamlen

May 21, 2011

## Abstract

The `trfrac` package provides convenient mechanisms for typesetting derivation trees in  $\text{\LaTeX}$ . Derivation trees look like nested fractions; thus, the primary contribution of this software is a macro `\trfrac` that functions much like  $\text{\LaTeX}$ 's `\frac` for fractions, but with a number of important differences that make it better suited to tree-like structures.

## 1 Introduction

Typing rules and typing derivations in formal typing systems are usually written like fractions in which the numerator consists of one or more premise judgments or derivations, and the denominator consists of a single conclusion judgment. For example, the typing rule for application in the simply-typed lambda calculus is typically written

$$\frac{\Gamma \vdash e_1 : \tau \rightarrow \tau' \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 e_2 : \tau'} (app)$$

where  $\Gamma \vdash e_1 : \tau \rightarrow \tau'$  and  $\Gamma \vdash e_2 : \tau$  are the two premise judgments of the rule, and  $\Gamma \vdash e_1 e_2 : \tau'$  is the conclusion judgment.

Simple rules like the above are easy to typeset with  $\text{\LaTeX}$ 's built-in `\frac` macro, or with fraction macros like `\dfrac` provided by the `amslatex` package; but when derivations consist of a tree of nested rules, `\frac` and `\dfrac` often prove inadequate because they yield spacing and alignment that is unsuited to derivation trees. For example, consider the derivation

$$\frac{\Gamma \vdash e_1 : \tau \rightarrow \tau' \quad \frac{\Gamma \vdash e_2 : \tau'' \rightarrow \tau \quad \Gamma \vdash e_3 : \tau''}{\Gamma \vdash e_2 e_3 : \tau} (app)}{\Gamma \vdash e_1 (e_2 e_3) : \tau'} (app)$$

which contains a nested subderivation of one of its premise judgments. This was easily typeset using the `\trfrac` macro provided by the `trfrac` package. The complete  $\text{\LaTeX}$  code is as follows:

---

\*This document corresponds to `trfrac` v2.3, dated 2011/5/20.

```

\[\ \trfrac[\small(\emph{app})]
  {\Gamma \vdash e_1:\tau \rightarrow \tau' \quad \Gamma \vdash e_2:\tau'' \rightarrow \tau \quad \Gamma \vdash e_3:\tau''}
  {\Gamma \vdash e_2 e_3:\tau}
  (app)
\]

```

In contrast, if one writes analogous code using `\dfrac` instead of `\trfrac`, the result is

$$\frac{\Gamma \vdash e_1 : \tau \rightarrow \tau' \quad \frac{\Gamma \vdash e_2 : \tau'' \rightarrow \tau \quad \Gamma \vdash e_3 : \tau''}{\Gamma \vdash e_2 e_3 : \tau} (app)}{\Gamma \vdash e_1 (e_2 e_3) : \tau'} (app)$$

which is unsuitable for two reasons: First, the leftmost premise is floating up above the fraction bar when it should be sitting on the bar. Second, the bottom fraction bar is longer than it should be; it has been unnecessarily extended rightward to span the entire “numerator” of the “fraction”. This can become problematic when derivations are large and horizontal space is at a premium.

In addition to providing `\trfrac` as an alternative to fraction macros, the `trfrac` package also provides two environments that can be helpful for squeezing numerous and/or lengthy premise judgments into a smaller amount of horizontal space. Derivations can sometimes contain long judgments that must be wrapped, or numerous premise judgments that must be placed above one another for lack of space. The `trfrac` package provides math environments `trgather` and `tralign` for stacking sets of judgments vertically within a derivation tree. The `trgather` environment horizontally centers each judgment in the stack, whereas the `tralign` environment aligns each judgment at a specified point. For example, the derivation

$$\frac{\Gamma \vdash e : \tau_1 + \tau_2 \quad \Gamma[x_1 \mapsto \tau_1] \vdash e_1 : \tau \quad \Gamma[x_2 \mapsto \tau_2] \vdash e_2 : \tau}{\Gamma \vdash \mathbf{case} \ e \ \mathbf{of} \ in_1(x_1) \Rightarrow e_1 \mid in_2(x_2) \Rightarrow e_2 : \tau} (case)$$

can be produced with

```

\[\ \trfrac[\small(\emph{case})]
  {\begin{trgather}
    \Gamma \vdash e:\tau_1+\tau_2 \ \backslash\backslash
    \Gamma[x_1 \mapsto \tau_1] \vdash e_1:\tau \ \backslash\backslash
    \Gamma[x_2 \mapsto \tau_2] \vdash e_2:\tau \ \backslash\backslash
  \end{trgather}}
  {\begin{tralign}
    \Gamma \vdash \mathbf{case} \ e \ \mathbf{of} \ \mathit{in}_1(x_1) \Rightarrow e_1 \ \backslash\backslash
    \mid \ , \ \mathit{in}_2(x_2) \Rightarrow e_2 : \tau \ \backslash\backslash
  \end{tralign}}
\]

```

The `split` and `aligned` environments provided by the `amsmath` package provide facilities for vertically stacking sub-equations in similar ways, but like `\dfrac`, these often prove inadequate for derivation trees because they vertically center the material they typeset, causing any adjacent material to float above the fraction bar.

## 2 Usage

To start using the macros and environments provided by the `trfrac` package, place the line

```
\usepackage{trfrac}
```

somewhere near the top of your `TEX` file.

`\trfrac` Subsequently, within any math environment you can use

```
\trfrac[⟨name⟩]{⟨premises⟩}{⟨consequent⟩}
```

to typeset a derivation rule, where `⟨name⟩` is optional material processed in horizontal text mode that is to sit to the right of the fraction bar, `⟨premises⟩` is math material that is to sit above the fraction bar, and `⟨consequent⟩` is math material that is to sit below the fraction bar. The `⟨consequent⟩` and `⟨name⟩` may not contain additional `\trfrac` macros, but the `⟨premises⟩` may.

The outermost `\trfrac` in a derivation is vertically centered on its fraction bar, but inner `\trfrac`'s in a derivation have baselines at the bottoms of their consequents. This makes it possible to typeset something like

$$\mathcal{D} = \frac{\Gamma \vdash e_1 : \tau \rightarrow \tau' \quad \frac{\Gamma \vdash e_2 : \tau'' \rightarrow \tau \quad \Gamma \vdash e_3 : \tau''}{\Gamma \vdash e_2 e_3 : \tau} (app)}{\Gamma \vdash e_1 (e_2 e_3) : \tau'} (app)$$

in which the material “ $\mathcal{D} =$ ” should be vertically centered with respect to the bottom fraction bar, but the left premise should not be centered with respect to the right subderivation’s fraction bar.

`trgather` The environment

```
\begin{trgather}
  ⟨eqn1⟩ \\[⟨len1⟩]
  ⟨eqn2⟩ \\[⟨len2⟩]
  \vdots
  ⟨eqnn⟩ \\[⟨lenn⟩]
\end{trgather}
```

can be used in math mode to stack a set of equations, each of which is to be centered horizontally over one another. Here, `⟨eqni⟩` is material processed in math mode and `⟨leni⟩` is an optional length that modifies the vertical spacing between

the current equation and the next. The baseline of the resulting stack of equations is the baseline of the bottom equation in the stack.

`tralign`      The environment

```

\begin{tralign}
  \langle left_1 \rangle \& \langle right_1 \rangle \ \ll [\langle len_1 \rangle]
  \langle left_2 \rangle \& \langle right_2 \rangle \ \ll [\langle len_2 \rangle]
  \vdots
  \langle left_n \rangle \& \langle right_n \rangle \ \ll [\langle len_n \rangle]
\end{tralign}

```

can be used in math mode to stack a set of equations, each of which is to be aligned horizontally by the position of the `&` in the equation. Here,  $\langle left_i \rangle$  and  $\langle right_i \rangle$  is the math material to be placed to the left and right, respectively, of the vertical center axis of the resulting stack of equations; and  $\langle len_i \rangle$  is an optional length that modifies the vertical spacing between the current equation and the next. The baseline of the resulting stack of equations is the baseline of the bottom equation in the stack.

`\trtopgap`      The space just above and just below each fraction bar can be adjusted by  
`\trbotgap`      changing the values of lengths `\trtopgap` and `\trbotgap`. For example, to decrease the distance between fraction bars and numerators by 2 points and increase the distance between fraction bars and denominators by 1 point of space, you could write:

```

\trtopgap=-2pt
\trbotgap=1pt

```

### 3 Implementation

The following is a verbatim listing of the `trfrac` package implementation with comments explaining how it works.

```

\trtopgap      Declare dimension registers that allow users to control the amount of space above
\trbotgap      and below fraction bars.
  1 \newdimen\trtopgap\trtopgap\z@
  2 \newdimen\trbotgap\trbotgap\z@

\TRF@startp    When processing a numerator, the \TRF@trfrac code must be able to determine
\TRF@endp      if (a) the current derivation is the first thing being added to the current horizontal
                list, and if (b) the current horizontal list ends in a derivation. To accomplish this,
                two sentinel penalties are added to horizontal lists: \TRF@startp is added at the
                beginning of a horizontal list comprising a numerator, and \TRF@endp is added
                just after every derivation. Then \lastpenalty can be used to inquire whether
                the current horizontal list ends in either penalty. (Since derivations encased within
                numerators cannot undergo line-breaking, these extra penalties will have no effect
                on LATEX's processing of the actual output.)

```

```

3 \newcommand\TRF@startp{31415926}
4 \newcommand\TRF@endp{27182818}

\ifTRF@first In accordance with the above, the following conditional remembers if the current
derivation is the first in the horizontal list comprising the numerator of a larger
derivation.
5 \newif\ifTRF@first

\ifTRF@outer The outermost derivation behaves slightly differently than subderivations within
a derivation tree. (Its baseline is its fraction bar instead of its denominator's
baseline, and it should not include the spurious \TRF@endp penalty, since line-
breaking might occur.) The following conditional therefore remembers whether
we're at the outermost level.
6 \newif\ifTRF@outer\TRF@outertrue

\ifTRF@derivok Subderivations are not permitted to appear within denominators or within rule
names. The \ifTRF@derivok conditional is set to false when processing such
material.
7 \newif\ifTRF@derivok\TRF@derivoktrue

\TRF@lhang After processing a horizontal list that might contain one or more derivations,
\TRF@rhang \TRF@lhang holds the distance from the left edge of the resulting box to the
leftmost item's denominator (or Opt if the leftmost list item is not a derivation),
and \TRF@rhang holds the distance from the rightmost item's denominator to the
right edge of the resulting box (or Opt if the rightmost list item is not a derivation).
8 \newdimen\TRF@lhang\TRF@lhang\z@
9 \newdimen\TRF@rhang\TRF@rhang\z@

\TRF@lsave We also need a place to save the value of \TRF@lhang so that we can (sometimes)
restore it to its original value after it has been globally modified by nested \trfrac
macros.
10 \newdimen\TRF@lsave

\TRF@ni The temporary registers declared below store the the numerator indentation, bar
\TRF@bi indentation, and bar width, respectively, when typesetting a derivation rule.
\TRF@bw 11 \newdimen\TRF@ni
12 \newdimen\TRF@bi
13 \newdimen\TRF@bw

\TRF@numbox While constructing a derivation, the following three boxes hold the derivation's
\TRF@denombox numerator, denominator, and rule name, respectively.
\TRF@eqbox 14 \newbox\TRF@numbox
15 \newbox\TRF@denombox
16 \newbox\TRF@eqbox

```

`\TRF@trfrac` The following macro typesets a single derivation rule in a derivation tree. Its arguments are (1) the rule name (processed in horizontal mode), (2) the numerator (processed in math mode), and (3) the denominator (processed in math mode).

```
17 \newcommand\TRF@trfrac[3]{%
18 \begingroup%
```

We start by saving the original value of `\TRF@lhang` and deciding whether this is the first subderivation in a numerator of a larger derivation. These must come before anything else, and the results used later.

```
19 \ifnum\lastpenalty=\TRF@startp\relax%
20 \TRF@firsttrue%
21 \else%
22 \TRF@firstfalse%
23 \fi%
24 \TRF@lsave\TRF@lhang%
```

Typeset the equation number and put it in a box.

```
25 \setbox\TRF@eqbox\hbox{\TRF@derivokfalse#1}%
26 \ifdim\wd\TRF@eqbox>\z@%
27 \setbox\TRF@eqbox\hbox{\kern\p@\unhbox\TRF@eqbox}%
28 \fi%
```

Typeset the numerator and put it in a box. We begin the box with the `\TRF@startp` penalty so that if the first list item is a derivation, it will see that nothing has preceded it and will therefore set `\TRF@lhang`. We initialize it to zero so that if the first item is not a derivation, `\TRF@lhang` will remain zero. If the last thing in the numerator is a `\TRF@endp` penalty, then the list must have ended with a subderivation whose denominator is a distance `\TRF@rhang` from its right edge. Otherwise the rightmost item was not a subderivation, so we zero `\TRF@rhang`.

```
29 \TRF@lhang\z@%
30 \setbox\TRF@numbox\hbox{%
31 \TRF@outerfalse%
32 \m@th$\strut%
33 \penalty\TRF@startp\relax%
34 #2%
35 \ifnum\lastpenalty=\TRF@endp\relax\else%
36 \global\TRF@rhang\z@%
37 \fi%
38 $%
39 }%
```

Finally, typeset the denominator in a box. Denominators may not contain subderivations, so this preserves the values of `\TRF@lhang` and `\TRF@rhang`.

```
40 \setbox\TRF@denombox\hbox{%
41 \TRF@derivokfalse%
42 \m@th$\strut#3$%
43 }%
```

Compute the bar width  $bw = \max(n - l - r, d)$ , denominator indentation  $l' = \max((n - r + l) - d, 0)/2$ , numerator indentation  $ni = \max(d - (n - r + l), 0)/2$ , and bar indentation  $bi = \min(ni + l, di)$ , respectively, where  $n$  is the numerator

width,  $d$  is the denominator width,  $l$  is the old value of `\TRF@lhang`, and  $r$  is `\TRF@rhang`.

```

44   \dimen@\wd\TRF@numbox%
45   \advance\dimen@-\TRF@rhang%
46   \TRF@bw\dimen@%
47   \advance\TRF@bw-\TRF@lhang%
48   \ifdim\wd\TRF@denombox>\TRF@bw\TRF@bw\wd\TRF@denombox\fi%
49   \advance\dimen@\TRF@lhang%
50   \advance\dimen@-\wd\TRF@denombox%
51   \TRF@ni\ifdim\dimen@>\z@\z@\else-.5\dimen@\fi%
52   \TRF@bi\TRF@ni%
53   \advance\TRF@bi\TRF@lhang%
54   \TRF@lhang\ifdim\dimen@<\z@\z@\else.5\dimen@\fi%
55   \ifdim\TRF@bi>\TRF@lhang\TRF@bi\TRF@lhang\fi%

```

Compute the new value of `\TRF@rhang` as  $r' = \max(ni + n, bi + bw + e) - l' - d$ , where  $e$  is the width of the equation number. The final result is globally assigned so that its value will persist beyond the local scope in which numerators are typeset.

```

56   \TRF@rhang\TRF@ni%
57   \advance\TRF@rhang\wd\TRF@numbox%
58   \dimen@\TRF@bi%
59   \advance\dimen@\TRF@bw%
60   \advance\dimen@\wd\TRF@eqbox%
61   \ifdim\dimen@>\TRF@rhang\TRF@rhang\dimen@\fi%
62   \advance\TRF@rhang-\TRF@lhang%
63   \advance\TRF@rhang-\wd\TRF@denombox%
64   \global\TRF@rhang\TRF@rhang%

```

Begin typesetting the final fraction as a math-inner node. The vbox that comprises the node has an hbox comprising the denominator on the bottom and an hbox comprising everything else on the top. This allows the denominator baseline to become the baseline of the entire fraction. If this is the outermost level, the final box is set with `\vtop` so that the numerator is the baseline, and then raised a bit so that the fraction bar becomes the baseline.

```

65   \mathinner{%
66     \ifTRF@outer%
67       \raise\fontdimen22\textfont\tw@\vtop%
68     \else%
69       \vbox%
70       \fi%
71     {%
72       \offinterlineskip%
73       \hbox{%
74         \vbox{%

```

Compute the distance between the numerator and the fraction bar. The formula for computing this number is given in step 15d of Appendix G of *The T<sub>E</sub>Xbook* (p. 445).

```

75       \dimen@\fontdimen8\textfont\tw@%
76       \advance\dimen@-\fontdimen22\textfont\tw@%

```

```

77         \advance\dimen@-.5\fontdimen8\textfont\thr@@%
78         \advance\dimen@-\dp\TRF@numbox%
79         \ifdim\dimen@<3\fontdimen8\textfont\thr@@%
80             \dimen@3\fontdimen8\textfont\thr@@%
81         \fi%
82         \advance\dimen@\trtopgap%

```

Typeset the numerator part, horizontally centered over the fraction bar (but possibly overhanging the equation number).

```

83         \hbox{\kern\TRF@ni\unhbox\TRF@numbox}%
84         \kern\dimen@%

```

Typeset the fraction bar followed by the equation number. The height of the fraction bar is based on math font metric 8. The equation number is typeset at zero height so that it does not push the numerator or denominator away from the fraction bar.

```

85         \hbox{%
86             \kern\TRF@bi%
87             \vrule\@width\TRF@bw%
88                 \@height.5\fontdimen8\textfont\thr@@%
89                 \@depth.5\fontdimen8\textfont\thr@@%
90             \vbox{%
91                 \dimen@.5\ht\TRF@eqbox%
92                 \advance\dimen@.5\dp\TRF@eqbox%
93                 \kern-\dimen@%
94                 \box\TRF@eqbox%
95                 \kern-\dimen@%
96             }%
97         }%
98     }%
99 }%

```

Insert the vertical space that separates a fraction bar from its denominator. See step 15d of Appendix G of *The T<sub>E</sub>Xbook* (p. 445) for a derivation of this distance.

```

100         \dimen@\fontdimen11\textfont\tw@%
101         \advance\dimen@\fontdimen22\textfont\tw@%
102         \advance\dimen@-.5\fontdimen8\textfont\thr@@%
103         \advance\dimen@-\ht\TRF@denombox%
104         \ifdim\dimen@<3\fontdimen8\textfont\thr@@%
105             \dimen@3\fontdimen8\textfont\thr@@%
106         \fi%
107         \advance\dimen@\trbotgap%
108         \kern\dimen@%

```

Typeset the denominator horizontally centered below the fraction bar.

```

109         \hbox{\kern\TRF@lhang\box\TRF@denombox}%
110     }%
111 }%

```

If this is a subderivation within another, add a `\TRF@endp` penalty to the current horizontal list so that if this is the last item in a numerator, the value we computed

for `\TRF@rhang` will be used by the outer derivation to determine the amount by which its numerator may overhang the fraction bar to the right.

```
112 \ifTRF@outer\else%
113 \penalty\TRF@endp\relax%
114 \fi%
```

If this is the leftmost subderivation in the numerator of another derivation, the value of `\TRF@lhang` must be made global so that it can be used by the outer derivation to compute the amount by which its numerator may overhang the fraction bar to the left. Otherwise `\TRF@lhang` must be restored to whatever value it had on entrance to this `\trfrac`. (It is not enough to just let it get restored to whatever it was before the current local scope, since some nested modifications to it might have been global.)

```
115 \global\TRF@lhang\ifTRF@first\TRF@lhang\else\TRF@lsave\fi%
116 \endgroup%
117 }
```

`\trfrac` The following is the entrypoint to the derivation code. The `\trfrac` macro takes one optional argument which, if present, is the name of the rule (processed in horizontal mode), followed by two mandatory arguments—the rule’s numerator and denominator. Derivations can only appear in math mode, and cannot appear within the denominator of another derivation or within its rule name.

```
118 \newcommand\trfrac[3] [] {%
119 \ifTRF@derivok%
120 \ifmmode%
121 \TRF@trfrac{#1}{#2}{#3}%
122 \else%
123 \TRF@mmerr%
124 \fi%
125 \else%
126 \TRF@derr%
127 \fi%
128 }
```

`\TRF@mmerr` Produce an error if `\trfrac` is used outside of math mode.

```
129 \newcommand\TRF@mmerr{%
130 \PackageError{trfrac}{\protect\trfrac\space%
131 only allowed in math mode}%
132 {I encountered a \protect\trfrac\space%
133 macro without first encountering a begin-math token.}%
134 }
```

`\TRF@derr` Subderivations within denominators are not supported because that would require computing chastic structures rather than tree structures.

```
135 \newcommand\TRF@derr{%
136 \PackageError{trfrac}{\protect\trfrac\space%
137 not allowed in the consequent of another \protect\trfrac}%
138 {\protect\trfrac\space can only appear in the premise (numerator)%
```

```

139     ) of another \protect\trfrac, not in the consequent (denominator%
140     ) of a \protect\trfrac.}%
141 }

\TRF@next The \TRF@next token is here reserved so that it can be used in conjunction with
\futurelet (when processing optional arguments).
142 \newcommand\TRF@next{}

\TRF@sep In a trgather or tralign environment, the \ macro is used to delimit lines. It
is temporarily set equal to \TRF@sep (below), which is just like \cr in an \halign
except that it takes an optional length argument that has the effect of inserting a
vertical space between the lines.
143 \newcommand\TRF@sep{\futurelet\TRF@next\TRF@@sep}

\TRF@@sep The following processes the optional argument to \TRF@sep.
144 \newcommand\TRF@@sep{%
145   \ifx[\TRF@next%
146     \let\TRF@next\TRF@separg%
147   \else%
148     \let\TRF@next\TRF@sepnoarg%
149   \fi%
150   \TRF@next%
151 }

\TRF@separg The following is the code for a \TRF@sep that has been provided the optional
argument.
152 \newcommand\TRF@separg{}
153 \def\TRF@separg[#1]{\cr\noalign{\kern#1}}

\TRF@sepnoarg The following is the code for a \TRF@sep that has not been provided the optional
argument. You might think that it would have been simpler to use \let instead
of \newcommand in the line below, or even to just use \cr within some of the code
above instead of defining this new macro at all, but \cr is a strange beast; it can
only be expanded onto the token stream when we're sure of everything that is to
follow it.
154 \newcommand\TRF@sepnoarg{\cr}

trgather environment is just like amsmath's gathered environment except that
its baseline is at the bottom instead of in the middle.
155 \newenvironment{trgather}{%
156   \vbox\bgroup%
157   \let\\ \TRF@sep%
158   \ialign\bgroup\hfil{\m@th$##$}\hfil\cr%
159 }{%
160   \crcr\egroup\egroup%
161 }

```

`tralign` A `tralign` environment is like `amsmath`'s `aligned` environment except that its baseline is at the bottom instead of in the middle, and it's restricted to two columns (for simplicity).

```

162 \newenvironment{tralign}{%
163   \vbox\bgroup%
164     \let\\TRF@sep%
165     \ialign\bgroup\hfil{\m@th$##\null$}&{\m@th$\null##$}\hfil\cr%
166 }{%
167   \crcr\egroup\egroup%
168 }

```

## Index

Numbers written in *italics* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

<b>Symbols</b>	<b>M</b>	<code>\TRF@lhang</code> . . . . .
<code>\\</code> . . . . . 157, 164	<code>\mathinner</code> . . . . . 65	8, 24, 29, 47, 49, 53–55, 62, 109, 115
<b>C</b>	<b>N</b>	<code>\TRF@lsave</code> . 10, 24, 115
<code>\cr</code> . . 153, 154, 158, 165	<code>\newbox</code> . . . . . 14–16	<code>\TRF@mmerr</code> . . . 123, <u>129</u>
<code>\crcr</code> . . . . . 160, 167	<code>\newdimen</code> . . . 1, 2, 8–13	<code>\TRF@next</code> . <u>142</u> , 143, 145, 146, 148, 150
<b>D</b>	<code>\noalign</code> . . . . . 153	<code>\TRF@ni</code> <u>11</u> , 51, 52, 56, 83
<code>\dimen@</code> . . . . . 44–46, 49–51, 54, 58– 61, 75–80, 82, 84, 91–93, 95, 100–105, 107, 108	<b>T</b>	<code>\TRF@numbox</code> . . . . . <u>14</u> , 30, 44, 57, 78, 83
<b>E</b>	<code>tralign</code> (environment) <u>162</u>	<code>\TRF@outerfalse</code> . . . 31
environments:	<code>\trbotgap</code> . . . . . <u>1</u> , 107	<code>\TRF@outertrue</code> . . . . . 6
<code>tralign</code> . . . . . <u>162</u>	<code>\TRF@@sep</code> . . . . . 143, <u>144</u>	<code>\TRF@rhang</code> . . 8, 36, 45, 56, 57, 61–64
<code>trgather</code> . . . . . <u>155</u>	<code>\TRF@bi</code> . . . . . <u>11</u> , 52, 53, 55, 58, 86	<code>\TRF@sep</code> . . <u>143</u> , 157, 164
<b>I</b>	<code>\TRF@bw</code> <u>11</u> , 46–48, 59, 87	<code>\TRF@separg</code> . . . 146, <u>152</u>
<code>\ialign</code> . . . . . 158, 165	<code>\TRF@denombox</code> <u>14</u> , 40, 48, 50, 63, 103, 109	<code>\TRF@sepnarg</code> . 148, <u>154</u>
<code>\ifTRF@derivok</code> . . <u>7</u> , 119	<code>\TRF@derivokfalse</code> 25, 41	<code>\TRF@startp</code> . . . <u>3</u> , 19, 33
<code>\ifTRF@first</code> . . . . <u>5</u> , 115	<code>\TRF@derivoktrue</code> . . . 7	<code>\TRF@trfrac</code> . . . . <u>17</u> , 121
<code>\ifTRF@outer</code> . <u>6</u> , 66, 112	<code>\TRF@derr</code> . . . . 126, <u>135</u>	<code>\trfrac</code> . . . . . <u>118</u> , 130, 132, 136–140
<b>L</b>	<code>\TRF@endp</code> . . . <u>3</u> , 35, 113	<code>trgather</code> (environ- ment) . . . . . <u>155</u>
<code>\lastpenalty</code> . . . . 19, 35	<code>\TRF@eqbox</code> . <u>14</u> , 25– 27, 60, 91, 92, 94	<code>\trtopgap</code> . . . . . <u>1</u> , 82
	<code>\TRF@firstfalse</code> . . . 22	
	<code>\TRF@firsttrue</code> . . . . 20	