functan*

Macros for functional analysis and related domains

Antoine Lejay
<Antoine.Lejay@iecn.u-nancy.fr>

July, 13 2004
Printed on July 22, 2004

1 Introduction

This package is designed especially for the people working in domains of mathematics such as functional analysis and PDE theory. Of course, this package may be used in other domains.

In functional analysis and PDE theory, one may face a lot of names of spaces, sometimes depending on one or more parameters, such as $\mathrm{L}^2(\Omega)$, $\mathrm{H}^1_0(\Omega)$, ... Besides, one has to deal with norms, convergence and scalar product for each of these spaces. As a control sequence, i.e., a TEX command, consists only of letters, using TEX's standard macros is not convenient for dealing with such spaces. Yet composing each of these space names at each their occurrence increases the risk of errors, and tends to be tedious.

Fortunately, TEX provides some way to create commands whose name may contain arbitrary characters. We exploit this possibility to deal with names of functional spaces and related objects (norms, convergence, ...).

This package allows for example to typeset:

> We denote by $\mathrm{L}^2(\Omega)$ the space of measurable functions on Ω which are square integrable. The associated norm $\|\cdot\|_{\mathrm{L}^2(\Omega)}$ is defined by $\|f\|_{\mathrm{L}^2(\Omega)} = \left(\int_\Omega f(x)^2 dx\right)^{1/2}$ and the corresponding scalar product is $\langle f, g\rangle_{\mathrm{L}^2(\Omega)} = \int_\Omega f(x)g(x)dx$. We write $f_n \xrightarrow[n\to\infty]{\mathrm{L}^2(\Omega)} f$ if the sequence $(f_n)_n$ converges to f in $\mathrm{L}^2(\Omega)$. If for all $g \in \mathrm{L}^2(\Omega)$, $\langle f_n, g\rangle_{\mathrm{L}^2(\Omega)}$ converges to $\langle f, g\rangle_{\mathrm{L}^2(\Omega)}$, then we say that $(f_n)_n$ converges weakly to f in $\mathrm{L}^2(\Omega)$, and we write $f_n \xrightarrow[n\to\infty]{\mathrm{L}^2(\Omega)\text{-weak}} f$.

*Package version v1.0 of 2004/07/03.

The code for this example is the following:

```
\Macro{L2Om}{\mathrm{L}^2(\Omega)}

We denote by $\m{L2Om}$ the space of measurable
functions on $\Omega$ which are square integrable.
The associated norm $\norm{L2Om}{\cdot}$ is defined
by $\norm{L2Om}{f}=\left(\int_\Omega f(x)^2 dx\right)^{1/2}$
and the corresponding scalar product is
$\scalprod{L2Om}{f}{g}=\int_\Omega f(x)g(x)dx$.
We write $f_n\conv{L2Om}{n\to\infty}f$
if the sequence $(f_n)_n$ converges to $f$ in $\m{L2Om}$.
If for all $g\in\m{L2Om}$, $\scalprod{L2Om}{f_n}{g}$
converges to $\scalprod{L2Om}{f}{g}$, then
we say that $(f_n)_n$ converges weakly to $f$
in $\m{L2Om}$, and we write $f_n\wkconv{L2Om}{n\to\infty}f$.
```

This package uses of some commands of the package amsmath (some of these commands are available only for version **2.0** or higher), which is then automatically loaded. All the options of amsmath may be defined as options of functan.
Before using functan, be sure you have a recent version of amsmath installed in your computer.

2 Loading the package

This package is loaded as any package by the command `\usepackage`. This package has no options, except those of amsmath in order to avoir an option clash. Thus, the possible options are: `leqno`, `intlimits`, `nointlimits`, `sumlimits`, `nosumlimits`, `namelimits`, `nonamelinits`, `reqno`, `centertags`, `tbtags`, `cmex10` and `fleqn`.

Incompatibilities : This package has no known incompatibilities with other packages.

3 The commands

3.1 An alternative system of macros

This system consists in associating a sequence of tokens to a name by the command `\Macro`, and call it later by the command `\m`.

`\Macro` The command `\Macro{`⟨*name*⟩`}{`⟨*expansion*⟩`}` associates the list of tokens ⟨*expansion*⟩ to the list of tokens ⟨*name*⟩, which may contain any arbitrary

(empty string)	`auto`	`big`	`Big`	`bigg`	`Bigg`

Table 1: Possible arguments for the size

sequence of characters (except `{`, `}` and active characters). This macro produces a warning message "The macro '⟨*name*⟩' has already been defined" if ⟨*name*⟩ already exists.

Nota: The sequence ⟨*name*⟩ may correspond to the control sequence of a TEX macro without creating interferences. Internally, the system stores the expansion of ⟨*name*⟩ as the TEX control sequence `functan@macro@`⟨*name*⟩.

`\m` The command `\m{`⟨*name*⟩`}` gives the expansion corresponding to ⟨*name*⟩ if a couple ⟨*name*⟩ and ⟨*expansion*⟩ has already been defined by `\Macro`. Otherwise, an error message "The macro '⟨*name*⟩' is not defined" is produced. The error message "No macro's name given" is also produced if the argument is given, that is if `\m{}` is encountered.

```
\Macro{L2Om}{\mathrm{L}^2(\Omega)}
Denote by  $\m{L2Om}$ the space of functions such that...
```
Denote by $\mathrm{L}^2(\Omega)$ the space of functions such that...

Nota: A command `\mempty` is also defined whose action is similar to `\m`, except that it does not produce an error message if its argument is empty.

3.2 Norms

The command `\norm` allows to typeset norms easily.

`\norm` The command `\norm[`⟨*size*⟩`]{`⟨*name*⟩`}{`⟨*arg*⟩`}` is roughly an equivalent to `\|`⟨*arg*⟩`\|_{\m{`⟨*name*⟩`}}` but with a control on the size of the vertical bars. If ⟨*name*⟩ is empty, then nothing is put in subscript. The optional argument ⟨*size*⟩ is one of the element in Table 1. By default, ⟨*size*⟩ is set to `auto`, which is equivalent in using `\left\|`...`\right\|` around ⟨*arg*⟩. If ⟨*size*⟩ is empty, that is when one call `\norm[]{...}{...}`, then the delimitor ‖ is used with its normal size. If ⟨*size*⟩ is equal to `big`, `Big`, `bigg` or `Bigg`, then ⟨*size*⟩ is inserted before the delimitors ‖.

Nota: If ⟨*size*⟩ is not one of the elements of Table 1, the the normal size is used.

`\norm{}{f}`	$\|f\|$
`\norm{L2Om}{f}`	$\|f\|_{\mathrm{L}^2(\Omega)}$
`\norm[Big]{L2Om}{f}`	$\left\|f\right\|_{\mathrm{L}^2(\Omega)}$
`\norm{L2Om}{\frac{1}{f}}`	$\left\|\frac{1}{f}\right\|_{\mathrm{L}^2(\Omega)}$
`\norm[]{L2Om}{\frac{1}{f}}`	$\|\frac{1}{f}\|_{\mathrm{L}^2(\Omega)}$

`\norm*` The command `\norm*[`⟨*size*⟩`]{`⟨*subscr*⟩`}{`⟨*arg*⟩`}` works as `\norm`, except that ⟨*subscr*⟩ is not one of the macro defined by `\Macro`, but any sequence of tokens which is put in subscript. For example `\norm*{\infty}{f}` is equivalent to `\norm{}{f}_{\infty}`.

`\norm*{\infty}{f}`	$\|f\|_\infty$
`\norm*[Big]{\infty}{f}`	$\Big\|f\Big\|_\infty$

`\newnorm` The command `\newnorm{`⟨*name*⟩`}{`⟨*expansion*⟩`}` allows to override the behavior of the `\norm` macro. Instead of using a `\|...\|` structure, the user may for example use other delimitors. Here ⟨*name*⟩ is any sequence of tokens (except `{`, `}` and active characters), already defined by `\Macro` or not. The arguments `#1` (for the size) et `#2` (for the argument) may be used in ⟨*expansion*⟩. The macros `\casesize` and `\delonearg` (see Section 3.5 below) may be used to create some new kind of norms.

```
\newnorm{L2Om}{\delonearg[#1]{#2}{|}{|}}
```

`\norm{L2Om}{f}`	$\|f\|$
`\norm[Big]{L2Om}{f}`	$\Big\|f\Big\|$

Nota: It seems that some side effects appear when `\newnorm` is used inside a group. Hence, it is recommended to use this command in the preamble.

3.3 Scalar products

Macros for scalar products are similar to macros for the norms, except that two argument are needed.

`\scalprod` The command `\scalprod[`⟨*size*⟩`]{`⟨*name*⟩`}{`⟨*arg 1*⟩`}{`⟨*arg 2*⟩`}` creates roughly `\langle`⟨*arg 1*⟩`,`⟨*arg 2*⟩`\rangle_{\m{`⟨*name*⟩`}}`. The optional ⟨*size*⟩ argument shall be taken in Table 1, and acts as a modifier for the size of the brackets.

`\scalprod{}{f}{g}`	$\langle f, g\rangle$
`\scalprod{L2Om}{f}{g}`	$\langle f, g\rangle_{\mathrm{L}^2(\Omega)}$
`\scalprod{L2Om}{f}{\frac{1}{g}}`	$\left\langle f, \frac{1}{g}\right\rangle_{\mathrm{L}^2(\Omega)}$
`\scalprod[big]{L2Om}{f}{\frac{1}{g}}`	$\big\langle f, \frac{1}{g}\big\rangle_{\mathrm{L}^2(\Omega)}$

`\scalprod*` The command `\scalprod*[`⟨*size*⟩`]{`⟨*subscr*⟩`}{`⟨*arg 1*⟩`}{`⟨*arg 2*⟩`}` acts like `\scalprod`, except that ⟨*subscr*⟩ is not the name already defined by `\Macro`, but a sequence of tokens put in subscript after the closing bracket. The rules are the same as for `\norm`.

`\scalprod*{X,Y}{Ax}{y}`	$\langle Ax, y\rangle_{X,Y}$
`\scalprod*{X,Y}{B^{-1}x}{y}`	$\big\langle B^{-1}x, y\big\rangle_{X,Y}$
`\scalprod*[]{X,Y}{B^{-1}x}{y}`	$\langle B^{-1}x, y\rangle_{X,Y}$

`\newscalprod` The command `\newscalprod{`⟨*name*⟩`}{`⟨*expansion*⟩`}` allows to override the

behavior of `\scalprod`, as for `\newnorm`. Here, ⟨*expansion*⟩ may used the three arguments `#1` (for the optional size), `#2` (⟨*arg 1*⟩) and `#3` (⟨*arg 2*⟩). The command `\towarg` (see Section 3.5 below) may be used to help the user to deal with new types of scalar products.

```
\newscalprod{L2Om}{\deltwoarg[#1]{#2}{#3}{(}{)}{|}}
\scalprod{L2}{f}{g}          (f | g)
\scalprod[Big]{L2}{f}{g}     \Big(f \Big| g\Big)
\scalprod[]{L2}{f^2}{g^2}    (f^2 | g^2)
```

Nota: It seems that some side effects appear when `\newnorm` is used inside a group. Hence, it is recommended to used this command in the preamble.

3.4 Convergence

The package amsmath provides a way to create *extensible arrows*, whose size depends on the material put above and below the arrow.

`\conv` The command `\conv{`⟨*name*⟩`}{`⟨*below*⟩`}` creates an extensible arrow by putting the expansion of ⟨*name*⟩ above the arrow, and ⟨*below*⟩ under the arrow. This command uses the commands `\xrightarrow` of amsmath. However, the `\xrightarrow` command takes as an optional argument the material to be put below. Here, it is a mandatory second argument.

```
x_n\conv{}{} x                  x_n \conv{}{} x
f_n\conv{}{n\to\infty} f        f_n \conv{}{n\to\infty} f
f_n\conv{L2Om}{n\to\infty} f    f_n \conv{L2Om}{n\to\infty} f
```

`\conv*` The command `\conv*{`⟨*above*⟩`}{`⟨*below*⟩`}` is similar to `\conv`, except that both ⟨*above*⟩ and ⟨*below*⟩ are lists of tokens that are put respectively above and below the arrow.

```
f_n\conv*{\text{uniformly}}{n\to\infty} f   f_n \conv*{\text{uniformly}}{n\to\infty} f
```

`\newconv` The command `\newconv{`⟨*name*⟩`}{`⟨*expansion*⟩`}` allows to override the behavior of `\conv{`⟨*name*⟩`}`, whether ⟨*name*⟩ corresponds to a name defined by `\Macro` or not. The token list ⟨*expansion*⟩ may use one argument `#1`.

```
\newconv{*L2Om}{\conv*{\star\text{-}\m{L2Om}}{#1}}
f_n\conv{*L2Om}{n\to\infty} f    f_n \conv{*L2Om}{n\to\infty} f
```

`\leftconv` `\leftconv*` The commands `\leftconv{`⟨*name*⟩`}{`⟨*below*⟩`}` and `\leftconv*{`⟨*above*⟩`}{`⟨*below*⟩`}` act like `\conv` and `\conv*`, except that an arrow pointing to the left (←) is used instead of an arrow pointing to the right (→).

`\leftrightconv` `\leftrightconv*` The commands `\leftrightconv{`⟨*name*⟩`}{`⟨*below*⟩`}`

and `\leftrightconv*{`⟨*above*⟩`}{`⟨*below*⟩`}` act like `\conv` and `\conv*`, except that an arrow pointing both to the left and to the right (⟷) is used instead of an arrow pointing to the right (→).

There are other commands similar to `\conv`, `\conv*`, `\leftconv`, `\leftconv*`, `\leftrightconv` and `\leftrightconv*` that is with the same arguments, but with different types of arrows.

`\wkconv`
`\wkconv*`
`\leftwkconv`
`\leftwkconv*`
`\leftrightwkconv`
`\leftrightwkconv*`

The commands `\wkconv`, `\wkconv*`, `\leftwkconv`, `\leftwkconv*`, `\leftrightwkconv` and `\leftrightwkconv*` use ⇀ instead of →. Such an arrow is generally used to denote *weak convergence* (hence the `wk`).

```
f_n\wkconv{}{n\to\infty} f      f_n \wkconv{}{n\to\infty} f
f_n\wkconv{H10}{n\to\infty} f   f_n \wkconv{H10}{n\to\infty} f
```

`\Conv`
`\Conv*`
`\Leftconv`
`\Leftconv*`
`\Leftrightconv`
`\Leftrightconv*`

The commands `\Conv`, `\Conv*`, `\Leftconv`, `\Leftconv*`, `\Leftrightconv` and `\Leftrightconv*` use ⇒ instead of →. This may be used to denote convergence in distribution in probability theory. This could also be used for the implication and equivalence symbols in logic.

```
\mu_n\Conv{}{n\to\infty} \mu   \mu_n \Conv{}{n\to\infty} \mu
x<0\Leftrightconv*{\text{Lemma 1.1}}{} f(x)\geq 0
  x<0 \Leftrightconv*{\text{Lemma 1.1}}{} f(x)\geq 0
```

3.5 Defining macros

The package functan provides a few macros to help the user to define its own norms, scalar products, ...

`\delonearg`

The command `\delonearg[`⟨*size*⟩`]{`⟨*arg*⟩`}{`⟨*left del*⟩`}{`⟨*right del*⟩`}` surrounds ⟨*arg*⟩ the left delimitor ⟨*left del*⟩ and the right delimitor ⟨*right del*⟩. The possible values of the optional argument ⟨*size*⟩ (which is by default set to `auto`) are given in Table 1.

```
\newcommand{\event}[2][auto]{\delonearg[#1]{#2}{\{}{\}}}
\event{1,\dotsc,\frac{1}{n}}
\cup\event[]{\frac{1}{n^2},\dotsc,\frac{2}{n^2}}
  {1,...,1/n} ∪ {1/n,...,2/n}
\newcommand{\ket}[2][auto]{\delonearg[#1]{#2}{|}{\rangle}}
\ket{Ax}   |Ax⟩
\newcommand{\bra}[2][auto]{\delonearg[#1]{#2}{\langle}{|}}
\bra{Ax}   ⟨Ax|
```

`\deltwoarg`

The command `\deltwoarg[`⟨*size*⟩`]{`⟨*arg 1*⟩`}{`⟨*arg 2*⟩`}...`
`...{`⟨*left del*⟩`}{`⟨*right del*⟩`}{`⟨*sep*⟩`}` is similar to `\delonearg`, except that it separates ⟨*arg 1*⟩ and ⟨*arg 2*⟩ by a separator ⟨*sep.*⟩. If ⟨*sep*⟩ is set to `|`, then a vertical bar is used, with an automatic adjustment to the size of the delimitors.

```
\newcommand{\set}[3][auto]{\deltwoarg[#1]{#2}{#3}{\{}{\}}{\,;\,}}
\set{x\geq 0}{P(x)=0}    {x ≥ 0 ; P(x) = 0}
\newcommand{\braket}[3][auto]{%
\deltwoarg[#1]{#2}{#3}{\langle}{\rangle}{|}}
\braket{Ax}{y}    ⟨Ax | y⟩
```

`\casesize` The command `\casesize{`⟨*size*⟩`}{`⟨*auto*⟩`}{`⟨*empty*⟩`}`...
...`{`⟨*big*⟩`}{`⟨*Big*⟩`}{`⟨*bigg*⟩`}{`⟨*Bigg*⟩`}` takes ⟨*size*⟩ in Table 1 and executes `{`⟨*empty*⟩`}` if ⟨*size*⟩ is an empty list of tokens, ⟨*auto*⟩ if ⟨*size*⟩ is `auto`, ⟨*big*⟩ if ⟨*size*⟩ is `big`, ... The error message "Size argument '⟨*size*⟩' of 'casesize' not valid" is produced if ⟨*size*⟩ is not in Table 1

`\delcasesize` The command `\delcasesize{`⟨*size*⟩`}{`⟨*auto*⟩`}{`⟨*empty*⟩`}{`⟨*other*⟩`}` is similar to `\casesize`, except that ⟨*other*⟩ is executed if ⟨*size*⟩ is different from `auto` or from an empty list of tokens.

4 Examples

4.1 Macros with arguments

It is possible to define a macro ⟨*name*⟩ such that `\m{`⟨*name*⟩`}` accepts some arguments. For that, it is sufficient that it expands into a TEX command that accepts arguments.

```
\newcommand{\evolution}[2]{\m{#1}(0,T;\m{#2})}
\Macro{L20T}{\evolution{L2}}
\m{L20T}{H10Om}\cap\m{L20T}{L2Om}    L²(0,T;H¹₀(Ω)) ∩ L²(0,T;L²(Ω))
```

4.2 Matrix norms

A matrix norm with three bars may be defined by

```
\newnorm{matrix}{\delcasesize{#1}%
{\left|\left|\left| #2\right|\right|\right|}%
{|||#2|||}%
{#1|#1|#1|#2 #1|#1|#1|}}
```

so that

```
\norm{matrix}{A^2}\leq \norm{matrix}{A}^2    |||A²||| ≤ |||A|||²
\norm[]{matrix}{A^2}\leq \norm{matrix}{A}^2    |||A²||| ≤ |||A|||²
```

4.3 Sets

An example of code for defining sets was given with the documentation of `\deltwoarg`. Here is another possibility

```
\newcommand{\set}[3][auto]{%
\deltwoarg[#1]{#2\,}{\,#3}{\{}{\}}{|}}
\set{x>-1}{\frac{x}{1+x}\geq \frac{1}{2}}
```

$$\left\{x > -1 \;\middle|\; \frac{x}{1+x} \geq \frac{1}{2}\right\}$$

4.4 Duality products

The command `\casesize` may be used for defining a duality product, which is a bit tricky since we want the position of name of the dual space to be adjusted to the size of the delimitors. The command `\dual` defined by

```
\newcommand{\dual}[5][auto]{
\casesize{#1}{
\left.\vphantom{\left\langle #4;#5\right\rangle}%
\right._{\m{#2}}\left.%
\left\langle #4;#5\right\rangle\right._{\m{#3}}}%
{{}_{\m{#2}}\langle #4;#5\rangle_{\m{#3}}}%
{\bigl._{\m{#2}}\bigl\langle #4;#5\bigr\rangle_{\m{#3}}}%
{\Bigl._{\m{#2}}\Bigl\langle #4;#5\Bigr\rangle_{\m{#3}}}%
{\biggl._{\m{#2}}\biggl\langle #4;#5\biggr\rangle_{\m{#3}}}%
{\Biggl._{\m{#2}}\Biggl\langle #4;#5\Biggr\rangle_{\m{#3}}}}
```

may be used to produce

```
\Macro{H1}{\mathrm{H}^1}
\Macro{H-1}{\mathrm{H}^{-1}}
\dual{H-1}{H1}{f}{\frac{1}{g}}
\dual[]{H-1}{H1}{f}{\frac{1}{g}}
\dual[bigg]{H-1}{H1}{f}{\frac{1}{g}}
```

$$\begin{array}{l} {}_{\mathrm{H}^{-1}}\left\langle f;\frac{1}{g}\right\rangle_{\mathrm{H}^1} \\ {}_{\mathrm{H}^{-1}}\langle f;\frac{1}{g}\rangle_{\mathrm{H}^1} \\ {}_{\mathrm{H}^{-1}}\left\langle f;\frac{1}{g}\right\rangle_{\mathrm{H}^1} \end{array}$$

5 The code

5.1 Package heading

This package uses some code of the amsmath package. All the options of amsmath may be set as options of functan in order to avoir an option clash.

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{functan}[2004/07/03 v1.0, Macros for functional analysis]
\DeclareOption{leqno}{%
\PassOptionsToPackage{leqno}{amsmath}}
\DeclareOption{intlimits}{%
\PassOptionsToPackage{intlimits}{amsmath}}
\DeclareOption{nointlimits}{%
\PassOptionsToPackage{leqno}{amsmath}}
```

```
\DeclareOption{sumlimits}{%
\PassOptionsToPackage{sumlimits}{amsmath}}
\DeclareOption{nosumlimits}{%
\PassOptionsToPackage{leqno}{amsmath}}
\DeclareOption{namelimits}{%
\PassOptionsToPackage{namelimits}{amsmath}}
\DeclareOption{nonamelinits}{%
\PassOptionsToPackage{nonamelinits}{amsmath}}
\DeclareOption{reqno}{%
\PassOptionsToPackage{reqno}{amsmath}}
\DeclareOption{centertags}{%
\PassOptionsToPackage{centertags}{amsmath}}
\DeclareOption{tbtags}{%
\PassOptionsToPackage{tbtags}{amsmath}}
\DeclareOption{cmex10}{%
\PassOptionsToPackage{cmex10}{amsmath}}
\DeclareOption{fleqn}{%
\PassOptionsToPackage{fleqn}{amsmath}}
\ProcessOptions
\RequirePackage{amsmath}[2000/01/01 v2.0]
```

5.2 Defining macros

`\Macro` This commands defines a new macro with name ⟨*name*⟩, by creating a TEX command `functan@macro@`⟨*name*⟩. A warning message is sent if ⟨*name*⟩ has already been defined.

```
\newcommand{\Macro}[2]{\@ifundefined{functan@macro@#1}{}{%
\PackageWarning{functan}{The macro '#1' has already been defined}}
\@namedef{functan@macro@#1}{#2}}
```

`\m` `\mempty` This command allows to call the macro defined by ⟨*name*⟩. An error message is sent if ⟨*name*⟩ has not been defined, or if no argument is given. The macro `\mempty` provides no error message is the argument is empty. It is used to deal with `\conv` macros.

```
\newcommand{\m}[1]{\@ifempty{#1}{%
\PackageError{functan}{No macro's name given}{}
}{%
\@ifundefined{functan@macro@#1}{%
\PackageError{functan}{The macro `#1' is not defined}{}}{%
\@nameuse{functan@macro@#1}}}}
\newcommand{\mempty}[1]{\@ifempty{#1}{}{%
\@ifundefined{functan@macro@#1}{%
\PackageError{functan}{The macro `#1' is not defined}{}}{%
\@nameuse{functan@macro@#1}}}}
```

5.3 Dealing with size

\functan@casesize
\delcasesize

The macro `\delcasesize` is an alias for `\functan@casesize`. These macros require four arguments. This first one is the size among ∅ (empty argument for normal size) `auto` and one of the delimitors possible size (`big`, `Big`, `bigg`, `Bigg`). It `auto` is detected, then the second argument is executed. It no argument is given, then the third argument is executed. Otherwise, the fourth argument is executed. Another macro, `\casesize` (see below), is intended to execute a different code the the six distinct possibilities (∅, `auto`, `big`, `Big`, `bigg`, `Bigg`).

```
\global\def\functan@size@auto{auto}
\newcommand{\functan@casesize}[4]{\def\functan@size{#1}%
\ifx\functan@size\functan@size@auto #2\else%
\@ifempty{#1}{#3}{#4}\fi}
\let\delcasesize\functan@casesize
```

\casesize

The first argument is one of the delimitor's size specification (`auto`, ∅, `big`, `Big`, `bigg`, `Bigg`), and the six others are the different codes for each of the possible size in the order given just above.

```
\global\def\functan@size@big{big}
\global\def\functan@size@Big{Big}
\global\def\functan@size@bigg{bigg}
\global\def\functan@size@Bigg{Bigg}
\newcommand{\casesize}[7]{%
\def\functan@size{#1}
\@ifempty{#1}{#3}{%
\ifx\functan@size\functan@size@auto{#2}\else%
\ifx\functan@size\functan@size@big{#4}\else%
\ifx\functan@size\functan@size@Big{#5}\else%
\ifx\functan@size\functan@size@bigg{#6}\else%
\ifx\functan@size\functan@size@Bigg{#7}\else%
\PackageError{functan}{Size argument `#1' of `casesize' not valid}%
\fi\fi\fi\fi\fi}}
```

\delonearg
\deltwoarg

These two macros may be used for defining a pair of delimitors with automatic or manual size adjustment.

Nota: A `big-g-g` argument is transformed into TEX control sequence with the command `\@nameuse`. If the argument of nameuse is not a control sequence for an already defined control sequence, then `\@nameuse{`⟨...⟩`}` is transformed into an empty list.

```
\newcommand{\delonearg}[4][auto]{%
\functan@casesize{#1}{\left#3 #2 \right#4}{%
#3 #2 #4}{\@nameuse{#1l}#3 #2 \@nameuse{#1r}#4}}
\newcommand{\deltwoarg}[6][auto]{%
\def\functan@vert@bar{|}%
\edef\functan@arg@bar{#6}%
\ifx\functan@arg@bar\functan@vert@bar%
```

```
\functan@casesize{#1}{\left#4 #2\;\vrule\; #3\right#5}{%
#4 #2\;\vert\;#3 #5}{%
\@nameuse{#1l}#4 #2\;\@nameuse{#1}\vert\;#3 \@nameuse{#1r}#5}
\else
\functan@casesize{#1}{\left#4 #2 #6 #3 \right#5}{%
#4 #2 #6 #3  #5}{%
\@nameuse{#1l}#4  #2 #6 #3 \@nameuse{#1r}#5}
\fi}
```

5.4 Norms

`\functan@norm` This is the generic macro for the norms, which uses double bars ‖ as delimitors.

```
\newcommand{\functan@norm}[3]{%
\functan@casesize{#1}{\left\lVert #3\right\rVert%
\@ifnotempty{#2}{_{#2}}}{%
\lVert #3\rVert\@ifnotempty{#2}{_{#2}}}{%
\@nameuse{#1l}\lVert #3\@nameuse{#1r}\rVert%
\@ifnotempty{#2}{_{#2}}}}
```

`\functan@starred@norm` `\functan@nonstarred@norm` This two macros call the command `\functan@norm` either with (non-starred form) or without (starred form) expending the second argument ⟨*name*⟩ as the name of a macro defined by `\Macro`. Note that if the second argument is empty in `\functan@nonstarred@norm`, then no error message is produced. This allows to have a subscript just after the norm such as in `\norm{}{f}_1` without producing an error message of type "double subscript error". Note that the previous example may also be written `\norm*{1}{f}`. For the non-starred version, if a TEX macro `functan@named@norm@`⟨*name*⟩ exists (which is defined by `\newnorm`), then this macro is called instead of `\functan@norm`.

```
\newcommand{\functan@starred@norm}[3][auto]{%
\functan@norm{#1}{#2}{#3}}
\newcommand{\functan@nonstarred@norm}[3][auto]{%
\@ifundefined{functan@named@norm@#2}%
{\@ifempty{#2}{\functan@norm{#1}{}{#3}}{%
\functan@norm{#1}{\m{#2}}{#3}}}%
{\@nameuse{functan@named@norm@#2}{#1}{#3}}}
```

`\newnorm` This macro allows to override the behavior of the `\norm` macro by defining a TEX command `functan@named@norm@`⟨*name*⟩.

```
\newcommand{\newnorm}[1]{%
\@namedef{functan@named@norm@#1}##1##2}
```

`\norm` This macro is the one which is finally used. The presence of a star `*` is checked and either `\functan@starred@norm` or `\functan@nonstarred@norm` is called in consequence.

```
\newcommand{\norm}{\@ifstar{\functan@starred@norm}%
{\functan@nonstarred@norm}}
```

5.5 Convergence

Nota: It is for this set of macros that the compatibility with version 2.0 of `amsmath` or higher is important.

`\newconv` This macro allows to override the behavior of the `\conv` macro, by defining a TEX control sequence `functan@named@conv@`⟨*name*⟩.

```
\newcommand{\newconv}[1]{%
\@namedef{functan@named@conv@#1}##1}
```

`\conv` This macros stands for the usual convergence symbol (→). The presence of a star `*` is checked, and either `\functan@nonstarred@conv` or `\functan@starred@conv` is called in consequence. If a TEX command called `functan@named@conv@`⟨*name*⟩ exists, where ⟨*name*⟩ is the first argument of `\conv`, then this command is called instead of the others.

The command `\xrightarrow` is defined in the amsmath package.

```
\newcommand{\conv}{%
\@ifstar{\functan@starred@conv}{\functan@nonstarred@conv}}
\newcommand{\functan@nonstarred@conv}[2]{%
\@ifundefined{functan@named@conv@#1}%
{\xrightarrow[#2]{\mempty{#1}}}{\@nameuse{functan@named@conv@#1}{#2}}}
\newcommand{\functan@starred@conv}[2]{\xrightarrow[#2]{#1}}
```

`\xleftrightglobalarrow` `\Xleftrightglobalarrow` `\Xrightglobalarrow` `\Xleftglobalarrow` These commands are defined in a way similar to the one in `\xrightarrow` in the package amsmath. The first four arguments are lengths added on each side of the argument.

```
\newcommand{\xleftrightarrow}[2][]{%
\ext@arrow 9999\leftrightarrowfill@{#1}{#2}}
\newcommand{\Xrightarrow}[2][]{%
\ext@arrow 0359\Rightarrowfill@{#1}{#2}}
\newcommand{\Xleftarrow}[2][]{%
\ext@arrow 3095\Leftarrowfill@{#1}{#2}}
\newcommand{\Xleftrightarrow}[2][]{%
\ext@arrow 9999\Leftrightarrowfill@{#1}{#2}}
```

`\leftconv` `\leftrightconv` The commands `\leftconv` and `\leftrightconv` are defined as the command `\conv`, except that it is not possible to override them (however, any kind of arrow may be used with the command `\newconv`).

```
\newcommand{\leftconv}{%
\@ifstar{\functan@starred@leftconv}%
{\functan@nonstarred@leftconv}}
\newcommand{\functan@nonstarred@leftconv}[2]{%
\xleftarrow[#2]{\mempty{#1}}}
\newcommand{\functan@starred@leftconv}[2]{%
\xleftarrow[#2]{#1}}
\newcommand{\leftrightconv}{%
\@ifstar{\functan@starred@leftrightconv}%
```

```
{\functan@nonstarred@leftrightconv}}
\newcommand{\functan@nonstarred@leftrightconv}[2]{%
\xleftrightarrow[#2]{\mempty{#1}}}
\newcommand{\functan@starred@leftrightconv}[2]{%
\xleftrightarrow[#2]{#1}}
```

\Conv \Leftconv \Leftrightconv

These macros are defined like \conv.

```
\newcommand{\Conv}{%
\@ifstar{\functan@starred@Conv}{\functan@nonstarred@Conv}}
\newcommand{\functan@nonstarred@Conv}[2]{%
\Xrightarrow[#2]{\mempty{#1}}}
\newcommand{\functan@starred@Conv}[2]{%
\Xrightarrow[#2]{#1}}
\newcommand{\Leftconv}{%
\@ifstar{\functan@starred@Leftconv}%
{\functan@nonstarred@Leftconv}}
\newcommand{\functan@nonstarred@Leftconv}[2]{%
\Xleftarrow[#2]{\mempty{#1}}}
\newcommand{\functan@starred@Leftconv}[2]{%
\Xleftarrow[#2]{#1}}
\newcommand{\Leftrightconv}{%
\@ifstar{\functan@starred@Leftrightconv}%
{\functan@nonstarred@Leftrightconv}}
\newcommand{\functan@nonstarred@Leftrightconv}[2]{%
\Xleftrightarrow[#2]{\mempty{#1}}}
\newcommand{\functan@starred@Leftrightconv}[2]{%
\Xleftrightarrow[#2]{#1}}
```

\xrightharpoon \xleftharpoon \xleftrightharpoon

This type of arrow behaves as the other ones, except that an ⇀ is used at the end of the arrow.

```
\def\rightharpoonfill@{\arrowfill@\relbar\relbar\rightharpoonup}
\newcommand{\xrightharpoon}[2][]%
{\ext@arrow 0359\rightharpoonfill@{#1}{#2}}
\def\leftharpoonfill@{\arrowfill@\leftharpoonup\relbar\relbar}
\newcommand{\xleftharpoon}[2][]%
{\ext@arrow 3095\leftharpoonfill@{#1}{#2}}
\def\leftrightharpoonfill@%
{\arrowfill@\leftharpoonup\relbar\rightharpoonup}
\newcommand{\xleftrightharpoon}[2][]%
{\ext@arrow 3095\leftrightharpoonfill@{#1}{#2}}
```

\wkconv \leftwkconv \leftrightwkconv

These macro may be used for the weak convergence. Its definition is similar to the one of \conv.

```
\newcommand{\wkconv}{%
\@ifstar{\functan@starred@wkconv}%
{\functan@nonstarred@wkconv}}
\newcommand{\functan@nonstarred@wkconv}[2]{%
\xrightharpoon[#2]{\mempty{#1}}}
\newcommand{\functan@starred@wkconv}[2]{%
```

```
\xrightharpoon[#2]{#1}}
\newcommand{\leftwkconv}{%
\@ifstar{\functan@starred@leftwkconv}%
{\functan@nonstarred@leftwkconv}}
\newcommand{\functan@nonstarred@leftwkconv}[2]{%
\xleftharpoon[#2]{\mempty{#1}}}
\newcommand{\functan@starred@leftwkconv}[2]{%
\xleftharpoon[#2]{#1}}
\newcommand{\leftrightwkconv}{%
\@ifstar{\functan@starred@leftrightwkconv}%
{\functan@nonstarred@leftrightwkconv}}
\newcommand{\functan@nonstarred@leftrightwkconv}[2]{%
\xleftrightharpoon[#2]{\mempty{#1}}}
\newcommand{\functan@starred@leftrightwkconv}[2]{%
\xleftrightharpoon[#2]{#1}}
```

5.6 Scalar products

\scalprod The definition of the scalar product is similar to the definition of norms.

\newscalprod

```
\newcommand{\scalprod}{\@ifstar{\functan@starred@scalprod}%
{\functan@nonstarred@scalprod}}
\newcommand{\functan@starred@scalprod}[4][auto]{
\deltwoarg[#1]{#3}{#4}{\langle}%
{\rangle}{,}\@ifempty{#2}{}{_{#2}}}
\newcommand{\functan@nonstarred@scalprod}[4][auto]{
\@ifundefined{functan@scalprod@named@#2}{
\deltwoarg[#1]{#3}{#4}{\langle}%
{\rangle}{,}\@ifempty{#2}{}{_{\m{#2}}}}{%
\@nameuse{functan@scalprod@named@#2}{#1}{#3}{#4}}}
\newcommand{\newscalprod}[1]{%
\@namedef{functan@scalprod@named@#1}##1##2##3}
```

⟨/package⟩

Index

Numbers written in italic 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.