The phfqit package[1]

Philippe Faist *philippe.faist@bluewin.ch*

August 15, 2016

[1] *This document corresponds to phfqit v1.0, dated 2016/08/15. It is part of the phfqitltx package suite, see https://github.com/phfaist/phfqitltx.*

---

phfqit—Utilities to typeset stuff in Quantum Information Theory (quite biased towards theory), in particular general mathematical symbols, operators, and shorthands for entropy measures.

---

1 Introduction **2**

2 Basic Usage **2**
2.1 Semantic vs. Syntactic Notation . . . 3
2.2 Size Specification . . . 3

3 General Symbols (and Math Operators) **3**
3.1 Math/Linear Algebra Operators . . . 4
3.2 Poly symbol . . . 4
3.3 Bits and Bit Strings . . . 4
3.4 Logical Gates . . . 5

4 Lie Groups and Algebras **5**

5 Bra-Ket Notation and Delimited Expressions **5**

6 Entropy Measures **6**
6.1 Entropy, Conditional Entropy . . . 6
6.2 Relative Entropy Measures . . . 7
6.3 Coherent Relative Entropy . . . 9

7 Implementation **9**
7.1 Simple Symbols and Shorthands . . . 9
7.1.1 General Symbols . . . 9
7.1.2 Math Operators . . . 10
7.1.3 Poly . . . 11
7.1.4 Bits and Bit Strings . . . 11
7.1.5 Logical Gates . . . 12
7.1.6 Lie Groups & Algebras . . . 12
7.2 Bra-Ket Notation . . . 12
7.3 Delimited Expressions . . . 13
7.4 Entropy Measures . . . 14
7.4.1 Some Internal Utilities . . . 14

7.4.2 Entropy, Conditional Entropy, and Entropy Function . . . 14
7.4.3 Relative Entropies . . . 18
7.4.4 Coherent Relative Entropy . . . 20

Change History **23**

Index **23**

1 Introduction

This package provides some useful definitions, mainly for notation of mathematical expressions which are used in quantum information theory (at least by me).

Are included utilities for:

- General symbols and mathematical expressions (identity operator, trace, rank, diagonal, ...) (section 3)
- Formatting of bits and bit strings (subsection 3.3)
- Formatting of names of logical gates (subsection 3.4)
- Typesetting the names of Lie groups and algebras, for example $\mathrm{su}(N)$ (section 4)
- Bra-ket notation, and delimited expressions such as average, norm, ... (section 5)
- Typesetting entropy measures, including the Shannon/von Neumann entropy, the smooth entropies, relative entropies, as well as my coherent relative entropy

2 Basic Usage

This package is straightforward to use. There are no package options.

```
\usepackage{phfqit}
```

[TODO: In the future, use package options to control which definitions we want?]

2.1 Semantic vs. Syntactic Notation

The macros in this package are meant to represent a *mathematical quantity*, independently of its final *notation*. For example, `\Hmaxf` indicates corresponds to the "new-style" max-entropy defined with the fidelity,[1] independently of the notation. Then, if the default notation "$H_{\max}$" doesn't suit your taste, you may then simply redefine this command to display whatever you like (see for example instructions in subsection 6.1). This allows to keep better distinction between different measures which may share the same notation in different works of literature. It also allows to switch notation easily, even in documents which use several quantities whose notation may be potentially conflicting.

2.2 Size Specification

Many of the macros in this package allow their delimiters to be sized according to your taste. For example, if there is a large symbol in an entropy measure, say

$$H_{\min}(\bigotimes_i A_i \,|\, B)\ , \tag{1}$$

then it may be necessary to tune the size of the parenthesis delimiters.

This is done with the optional size specification ⟨*size-spec*⟩. The ⟨*size-spec*⟩, whenever it is accepted, is always optional.

The ⟨*size-spec*⟩ starts with the backtick character "`", and is followed by a single token which may be a star `*` or a size modifier macro such as `\big`, `\Big`, `\bigg` and `\Bigg`. If the star is specified, then the delimiters are sized with `\left` and `\right`; otherwise the corresponding size modifier is used. When no size specification is present, then the normal character size is used.

For example:

`\Hmin{\bigotimes_i A_i}[B]` gives $H_{\min}(\bigotimes_i A_i \,|\, B)$,

`\Hmin‘\Big{\bigotimes_i A_i}[B]` gives $H_{\min}\Big(\bigotimes_i A_i \,\Big|\, B\Big)$, and

`\Hmin‘*{\bigotimes_i A_i}[B]` gives $H_{\min}\left(\bigotimes_i A_i \,\middle|\, B\right)$.

■ 3 General Symbols (and Math Operators)

`\Hs` Hilbert space = $\mathscr{H}$.

`\Ident` Identity operator = $\mathbb{1}$.

[1] see Marco Tomamichel, Ph. D., ETH Zurich (2012) arXiv:1203.2142

`\IdentProc` Identity process. Possible usage syntax is:

`\IdentProc[A][A']{\rho}`	$\mathrm{id}_{A\to A'}(\rho)$
`\IdentProc[A]{\rho}`	$\mathrm{id}_{A}(\rho)$
`\IdentProc[A][A']{}`	$\mathrm{id}_{A\to A'}$
`\IdentProc[A]{}`	id_{A}
`\IdentProc{}`	id
`\IdentProc{\rho}`	$\mathrm{id}(\rho)$
`\IdentProc‘\big[A]{\rho}`	$\mathrm{id}_{A}\big(\rho\big)$

This macro accepts a size specification with the backtick (‘‘), see subsection 2.2.

`\ee^X` A macro for the exponential. Type the LaTeX code as if `\ee` were just the symbol, i.e. as `\ee^{<ARGUMENT>}`. The ideas is that this macro may be redefined to change the appearance of the e symbol, or even to change the notation to `\exp{<ARGUMENT>}` if needed for inline math.

3.1 Math/Linear Algebra Operators

`\tr` `\supp` `\rank` `\linspan` `\spec` `\diag` Provide some common math operators. The trace tr, the support supp, the rank rank, the linear span span, the spectrum spec and the diagonal matrix diag. (Note that `\span` is already defined by LaTeX, so that we resort to `\linspan`.)

`\Re` `\Im` Also, redefine `\Re` and `\Im` (real and imaginary parts of a complex number), to the more readable $\mathrm{Re}(z)$ and $\mathrm{Im}(z)$. (The original symbols were $\Re(z)$ and $\Im(z)$.)

3.2 Poly symbol

`\poly` Can be typeset in $\mathrm{poly}(n)$ time.

3.3 Bits and Bit Strings

`\bit` Format a bit value, for example `\bit{0}` or `\bit0` gives `0` or `1`. This command works both in math mode and text mode.

`\bitstring` Format a bit string. For example `\bitstring{01100101}` is rendered as $\overline{01100101}$. This command works both in math mode and text mode.

3.4 Logical Gates

`\gate` Format a logical gate. Essentially, this command typesets its argument in small-caps font. For example, with `\gate{C-not}` you get C-NOT. (The default formatting ignores the given capitalization, but if you redefine this command you could exploit this, e.g. by making the "C" in "Cnot" larger than the "not".)

This command works both in math mode and in text mode.

`\AND` Some standard gates. These typeset respectively as AND, XOR, C-NOT, NOT, and
`\XOR` NO-OP.
`\CNOT`
`\NOT`
`\NOOP`

■ 4 Lie Groups and Algebras

`\uu(N)` Format some common Lie groups and algebras.
`\UU(N)`
`\su(N)` `\SN(N)` is the symmetric group of N items, and formats by default as S_N.
`\SU(N)`
`\so(N)`
`\SO(N)`
`\SN(N)`

■ 5 Bra-Ket Notation and Delimited Expressions

All commands here work in math mode only. They all accept an optional argument, which is a size modifier. Use the starred form to enclose the delimiters with `\left...\right` and have the size determined automatically. Usage for example is:

`\ket{\psi}`	$\ket{\psi}$
`\ket[\big]{\psi}`	$\big\|\psi\big\rangle$
`\ket[\Big]{\psi}`	$\Big\|\psi\Big\rangle$
`\ket[\bigg]{\psi}`	$\bigg\|\psi\bigg\rangle$
`\ket[\Bigg]{\psi}`	$\Bigg\|\psi\Bigg\rangle$
`\ket*{\displaystyle\sum_k \psi_k}`	$\left\|\displaystyle\sum_k \psi_k\right\rangle$

`\ket` Typeset a quantum mechanical ket. `\ket{\psi}` gives $|\psi\rangle$.

`\bra` Typeset a bra. `\bra{\psi}` gives $\langle\psi|$.

`\braket` Typeset a bra-ket inner product. `\braket{\phi}{\psi}` gives $\langle\phi|\psi\rangle$.

`\ketbra` Typeset a ket-bra outer product. `\ketbra{\phi}{\psi}` gives $|\phi\rangle\langle\psi|$.

`\proj` Typeset a rank-1 projector determined by a ket. `\proj{\psi}` gives $|\psi\rangle\langle\psi|$.

`\matrixel` Typeset a matrix element. `\matrixel{\phi}{A}{\psi}` gives $\langle\phi|A|\psi\rangle$.

`\dmatrixel` Typeset a diagonal matrix element of an operator. `\dmatrixel{\phi}{A}` gives $\langle\phi|A|\phi\rangle$.

`\innerprod` Typeset an inner product using the mathematicians' notation. `\innerprod{\phi}{\psi}` gives $\langle\phi, \psi\rangle$.

There are also some further delimited expressions defined, for convenience.

`\abs` The absolute value of an expression. `\abs{A}` gives $|A|$.

`\avg` The average of an expression. `\avg[\big]{\sum_k A_k}` gives $\left\langle\sum_k A_k\right\rangle$.

`\norm` The norm of an expression. `\norm{A_k}` gives $\|A_k\|$. (You can add subscripts, e.g. `\norm{A_k}_\infty` is $\|A_k\|_\infty$.)

`\intervalc` A closed interval. `\intervalc{x}{y}` gives $[x, y]$.

`\intervalo` An open interval. `\intervalo{x}{y}` gives $]x, y[$.

`\intervalco` A semi-open interval, closed on the lower bound and open on the upper bound. `\intervalco{x}{y}` gives $[x, y[$.

`\intervaloc` A semi-open interval, open on the lower bound and closed on the upper bound. `\intervaloc{x}{y}` gives $]x, y]$.

6 Entropy Measures

6.1 Entropy, Conditional Entropy

The entropy measures (except for `\Hfunc`) all share the same syntax. This syntax is only described for the min-entropy `\Hmin`, but the other entropy measures enjoy the same features.

The name of the macros are chosen such that they identify the *abstract entropy measure*, and not necessarily the way one uses to write it down in a specific context. For example, for the "old" max-entropy $H_{\max,\mathrm{old}}(X)_\rho = \log \operatorname{rank} \rho$, you should use `\Hzero` independently of whether it should be denoted by H_0, $H_{\max}$ or $H_{\max,\mathrm{old}}$. This allows you to change the notation by redefining the command `\Hzero`, while making sure that the correct quantity is addressed.

(You might have both "old"-style and "new"-style max-entropy in the same paper.) The macros \Hmin, \Hzero, \Hmaxf and \HH may be redefined to change the subscript by using the following code (change "\mathrm{max},0" to your favorite subscript text):

```
\makeatletter
\renewcommand\Hzero{\@HHbase{\HHSym}{\mathrm{max},0}}
\makeatother
```

These commands are robust, meaning they can be used for example in figure captions and section headings.

\Hmin Min-entropy. The general syntax is \Hmin[⟨*state*⟩][⟨*epsilon*⟩]{⟨*target system*⟩}[⟨*conditioning system*⟩]. For example:

\Hmin{X}	$H_{\min}(X)$
\Hmin[\rho]{X}	$H_{\min}(X)_{\rho}$
\Hmin[\rho][\epsilon]{X}[Y]	$H^{\epsilon}_{\min}(X\,\|\,Y)_{\rho}$
\Hmin[\rho|\rho][\epsilon]{X}[Y]	$H^{\epsilon}_{\min}(X\,\|\,Y)_{\rho\|\rho}$
\Hmin[][\epsilon]{X}[Y]	$H^{\epsilon}_{\min}(X\,\|\,Y)$

\HH Shannon/von Neumann entropy. This macro has the same arguments as for \Hmin (even though, of course, there is no real use in smoothing the Shannon/von Neumann entropy...). For example, \HH[\rho]{X}[Y] gives $H(X\,|\,Y)_{\rho}$.

\Hzero Rényi-zero max-entropy. This macro has the same arguments as for \Hmin. For example, \Hzero[][\epsilon]{X}[Y] gives $H^{\epsilon}_{\max,0}(X\,|\,Y)$.

\Hmaxf The max-entropy. This macro has the same arguments as for \Hmin. For example, \Hmaxf[][\epsilon]{X}[Y] gives $H^{\epsilon}_{\max}(X\,|\,Y)$.

\Hfunc The entropy, written as a mathematical function. It is useful to write, e.g., $H(p_1\rho_1 + p_2\rho_2)$ (code: \Hfunc(p_1\rho_1 + p_2\rho_2)). Sizing specifications also work, e.g. \Hfunc‘\big(x) or \Hfunc‘*(x). However there is neither support for an epsilon-like superscript nor for a conditioning system.

\HHSym You may redefine this macro if you want to change the "H" symbol of all entropy measures. For example, with \renewcommand\HHSym{\spadesuit}, \Hmin{A}[B] would give $\spadesuit_{\min}(A\,|\,B)$.

6.2 Relative Entropy Measures

Relative entropies also have a corresponding set of commands.

\DD Generic relative entropy. The syntax of this command is either of the following: \DD⟨*size-spec*⟩{⟨*state*⟩}{⟨*relative-to state*⟩},

`\DD_{`⟨*subscript*⟩`}`⟨*size-spec*⟩`{`⟨*state*⟩`}{`⟨*relative-to state*⟩`}`,
`\DD_{`⟨*subscript*⟩`}^{`⟨*superscript*⟩`}`⟨*size-spec*⟩`{`⟨*state*⟩`}{`⟨*relative-to state*⟩`}`,
`\DD^{`⟨*superscript*⟩`}`⟨*size-spec*⟩`{`⟨*state*⟩`}{`⟨*relative-to state*⟩`}`.

In all cases, the argument is typeset as: $\big(\langle \textit{state} \rangle \big\| \langle \textit{relative-to state} \rangle\big)$. The size of the delimiters can be set with a size specification using the standard backtick syntax, as for the other entropies and as described in subsection 2.2.

Examples:

`\DD{\rho}{\sigma}`	$D(\rho\,\|\,\sigma)$
`\DD*{\rho}{\sigma}`	$D(\rho\,\|\,\sigma)$
`\DD‘\big{\rho}{\sigma}`	$D\big(\rho\,\big\|\,\sigma\big)$

You can also play around with subscripts and superscripts, but it is recommended to use the macros `\Dminf`, `\Dminz` and `\Dmax` directly. Specifying the subscripts and superscripts to `\DD` should only be done within new custom macros to define new relative entropy measures.

`\DD_{\mathrm{Rob}}^{\epsilon}{\rho}{\sigma}`	$D^{\epsilon}_{\mathrm{Rob}}(\rho\,\|\,\sigma)$
`\DD^{sup}{\rho}{\sigma}`	$D^{sup}(\rho\,\|\,\sigma)$

`\Dmax` The max-relative entropy. The syntax is `\Dmax[`⟨*epsilon*⟩`]`⟨*size-spec*⟩`{`⟨*state*⟩`}{`⟨*relative-to state*⟩`}`

For example `\Dmax[\epsilon]{\rho}{\sigma}` gives $D^{\epsilon}_{\max}(\rho\,\|\,\sigma)$ and `\Dmax[\epsilon]‘\big{\rho}{\sigma}` gives $D^{\epsilon}_{\max}\big(\rho\,\big\|\,\sigma\big)$.

The size-spec is as always given using the backtick syntax described in subsection 2.2.

`\Dminz` The “old” min-relative entropy, based on the Rényi-zero relative entropy. The syntax is the same as for `\Dmax`.

`\Dminf` The “new” min-relative entropy, defined using the fidelity. The syntax is the same as for `\Dmax`.

`\Dr` The Rob-relative entropy. The syntax is the same as for `\Dmax`.

`\DHyp` The hypothesis testing relative entropy. The syntax is the same as for `\Dmax`, except that by default the optional argument is `\eta`. That is, `\DHyp{\rho}{\sigma}` gives $D^{\eta}_{\mathrm{H}}(\rho\,\|\,\sigma)$. (This is because this quantity is directly defined with a η (or ϵ) built in, and it is not a zero-error quantity which is smoothed with the purified distance.)

`\DDSym` The symbol to use to denote a relative entropy. You may redefine this command to change the symbol. (This works like `\HHSym` above.)

6.3 Coherent Relative Entropy

A macro for a new quantity, the coherent relative entropy, is also available.

`\DCoh` Typeset a coherent relative entropy. The syntax is `\DCoh[`⟨*epsilon*⟩`]`⟨*size-spec*⟩`{`⟨*rho*⟩`}{`⟨*A*⟩`}{`⟨*B*⟩`}{`⟨*Gamma-1*⟩`}{`⟨*Gamma-2*⟩`}`.

For example, `\DCoh[\epsilon]{\rho}{A}{B}{\Gamma_A}{\Gamma_B}` gives $\bar{D}^{\epsilon}_{A\to B}(\rho_{BA} \,\|\, \Gamma_A, \Gamma_B)$.

The subscript `BA` is automatically added to the ⟨*rho*⟩ argument. If this is not desired, then begin the ⟨*rho*⟩ argument with a star. For example, `\DCoh{*\sigma_A\otimes\rho_B}{A}{B}{\Gamma_A}{\Gamma_B}` gives $\bar{D}_{A\to B}(\sigma_A \otimes \rho_B \,\|\, \Gamma_A, \Gamma_B)$.

The ⟨*size-spec*⟩ is of course optional and follows the same syntax as everywhere else (subsection 2.2).

`\emptysystem` Use the `\emptysystem` macro to denote a trivial system. For example, `\DCoh{\rho}{X}{\emptysystem}{\Gamma}{1}` gives $\bar{D}_{X\to\emptyset}(\rho_X \,\|\, \Gamma, 1)$.

`\DCSym` The symbol to use to denote a coherent relative entropy. You may redefine this command to change the symbol. (This works like `\HHSym` and `\DDSym` above.)

■ 7 Implementation

First, load dependent packages. Toolboxes, fonts and so on.

```
\RequirePackage{calc}
\RequirePackage{etoolbox}
\RequirePackage{amsmath}
\RequirePackage{dsfont}
\RequirePackage{mathrsfs}
\RequirePackage{mathtools}
```

7.1 Simple Symbols and Shorthands

7.1.1 General Symbols

These symbols are documented in section 3.

`\Hs` Hilbert space.

```
\newcommand{\Hs}{\mathscr{H}}
```

`\Ident` Identity operator, $\mathbb{1}$.

```
\newcommand{\Ident}{\mathds{1}}
```

\IdentProc Identity process.

```
\def\IdentProc{%
  \phfqit@parsesizearg\phfqit@IdentProc@maybeA%
}
\newcommand\phfqit@IdentProc@maybeA[1][]{%
  \def\phfqit@IdentProc@val@A{#1}%
  \phfqit@IdentProc@maybeB%
}
\newcommand\phfqit@IdentProc@maybeB[1][]{%
  \def\phfqit@IdentProc@val@B{#1}%
  \phfqit@IdentProc@arg%
}
\def\phfqit@IdentProc@arg#1{%
  \def\phfqit@IdentProc@val@arg{#1}%
```

At this point, prepare the three arguments, each expanded exactly as they were when given to these macros, and delegate the formatting to `\phfqit@IdentProc@do`.

```
  \edef\@tmp@args{%
    {\expandonce{\phfqit@IdentProc@val@A}}%
    {\expandonce{\phfqit@IdentProc@val@B}}%
    {\expandonce{\phfqit@IdentProc@val@arg}}%
  }%
  \expandafter\phfqit@IdentProc@do\@tmp@args%
}
\def\phfqit@IdentProc@do#1#2#3{%
  \operatorname{id}_{#1\notblank{#2}{\to #2}{}}%
  \notblank{#3}{\expandafter\phfqit@inner@parens\phfqit@val@sizearg{#3}}{}%
}
```

\ee^... Macro for the exponential.

```
\def\ee^#1{e^{#1}} % we could imagine that in inlines, we replace this by exp()...
```

7.1.2 Math Operators

See user documentation in subsection 3.1.

\tr \supp \rank \linspan \spec \diag Some common math operators. Note that `\span` is already defined by LaTeX, so we resort to `\linspan` for the linear span of a set of vectors.

```
\DeclareMathOperator{\tr}{tr}
\DeclareMathOperator{\supp}{supp}
\DeclareMathOperator{\rank}{rank}
\DeclareMathOperator{\linspan}{span}
\DeclareMathOperator{\spec}{spec}
\DeclareMathOperator{\diag}{diag}
```

\Re \Im

Also, alter the appearance of \Re and \Im to something more readable.

```
\let\phfqit@Re\Re
\DeclareMathOperator{\phfqit@Realpart}{Re}%
\renewcommand{\Re}{\phfqit@Realpart}
\let\phfqit@Im\Im
\DeclareMathOperator{\phfqit@Imagpart}{Im}%
\renewcommand{\Im}{\phfqit@Imagpart}
```

7.1.3 Poly

\poly

Poly symbol.

```
\DeclareMathOperator{\poly}{poly}
```

7.1.4 Bits and Bit Strings

See documentation in subsection 3.3

\bit \bitstring

Bits and bit strings.

```
\newcommand\bit[1]{\texttt{#1}}
\newcommand\bitstring[1]{\phfqit@bitstring{#1}}
```

The implementation of \bitstring needs some auxiliary internal macros.

```
\def\phfqit@bitstring#1{%
  \begingroup%
  \setlength{\phfqit@len@bit}{\maxof{\widthof{\bit{0}}}{\widthof{\bit{1}}}}%
  \phfqitBitstringFormat{\phfqit@bitstring@#1\phfqit@END}%
  \endgroup%
}
```

The internal \phfqit@bitstring@ macro picks up the next bit, and puts it into a LaTeX \makebox on its own with a fixed width.

```
\def\phfqit@bitstring@#1#2\phfqit@END{%
  \makebox[\phfqit@len@bit][c]{\phfqitBitstringFormatBit{#1}}%
  \if\relax\detokenize\expandafter{#2}\relax%
  \else%
```

If there are bits left, then recurse for the rest of the bitstring:

```
    \phfqitBitstringSep\phfqit@bitstring@#2\phfqit@END%
  \fi%
}
\newlength\phfqit@len@bit
```

\phfqitBitstringSep
\phfqitBitstringFormat

Redefine these to customize the bit string appearance.

```
\newcommand\phfqitBitstringSep{\hspace{0.3ex}}
\newcommand\phfqitBitstringFormat[1]{\ensuremath{\underline{\overline{#1}}}}
\def\phfqitBitstringFormatBit{\bit}
```

7.1.5 Logical Gates

See user documentation in subsection 3.4.

\gate

Generic macro to format a gate name.

```
\DeclareRobustCommand\gate[1]{\ifmmode\textsc{\lowercase{#1}}%
  \else{\rmfamily\textsc{\lowercase{#1}}}\fi}
```

\AND
\XOR
\CNOT
\NOT
\NOOP

Some common gates.

```
\newcommand{\AND}{\gate{And}}
\newcommand{\XOR}{\gate{Xor}}
\newcommand{\CNOT}{\gate{C-Not}}
\newcommand{\NOT}{\gate{Not}}
\newcommand{\NOOP}{\gate{No-Op}}
```

7.1.6 Lie Groups & Algebras

\uu(N)
\UU(N)
\su(N)
\SU(N)
\so(N)
\SO(N)
\SN(N)

Some Lie Groups & Algebras. See section 4

```
\def\uu(#1){\phfqit@fmtLieAlgebra{u}(#1)}
\def\UU(#1){\phfqit@fmtGroup{U}(#1)}
\def\su(#1){\phfqit@fmtLieAlgebra{su}(#1)}
\def\SU(#1){\phfqit@fmtGroup{SU}(#1)}
\def\so(#1){\phfqit@fmtLieAlgebra{so}(#1)}
\def\SO(#1){\phfqit@fmtGroup{SO}(#1)}
\def\SN(#1){\mathrm{S}_{#1}}
```

\phfqit@fmtLieAlgebra
\phfqit@fmtLieGroup

Override these to change the appearance of the group names or algebra names. The argument is the name of the group or algebra (e.g. su or SU).

```
\def\phfqit@fmtLieAlgebra#1{\mathrm{#1}}
\def\phfqit@fmtGroup#1{\mathrm{#1}}
```

7.2 Bra-Ket Notation

\ket
\bra
\braket
\ketbra
\proj
\matrixel
\dmatrixel
\innerprod

Bras, kets, norms, some delimiter stuff. User documentation in section 5.

```
\DeclarePairedDelimiterX\ket[1]{\lvert}{\rangle}{{#1}}
\DeclarePairedDelimiterX\bra[1]{\langle}{\rvert}{{#1}}
\DeclarePairedDelimiterX\braket[2]{\langle}{\rangle}{%
  {#1}\hspace*{0.2ex}\delimsize\vert\hspace*{0.2ex}{#2}%
}
\DeclarePairedDelimiterX\ketbra[2]{\lvert}{\rvert}{%
  {#1}\delimsize\rangle\hspace*{-0.25ex}\delimsize\langle{#2}%
}
\DeclarePairedDelimiterX\proj[1]{\lvert}{\rvert}{%
  {#1}\delimsize\rangle\hspace*{-0.25ex}\delimsize\langle{#1}%
}
\DeclarePairedDelimiterX\matrixel[3]{\langle}{\rangle}{%
  {#1}\hspace*{0.2ex}\delimsize\vert\hspace*{0.2ex}{#2}%
  \hspace*{0.2ex}\delimsize\vert\hspace*{0.2ex}{#3}%
}
\DeclarePairedDelimiterX\dmatrixel[2]{\langle}{\rangle}{%
  {#1}\hspace*{0.2ex}\delimsize\vert\hspace*{0.2ex}{#2}%
  \hspace*{0.2ex}\delimsize\vert\hspace*{0.2ex}{#1}%
}
\DeclarePairedDelimiterX\innerprod[2]{\langle}{\rangle}{%
  {#1},\hspace*{0.2ex}{#2}%
}
```

7.3 Delimited Expressions

Delimited expressions are documented in section 5.

\abs
\avg
\norm

Other delimited expressions.

```
\DeclarePairedDelimiterX\abs[1]{\lvert}{\rvert}{{#1}}
\DeclarePairedDelimiterX\avg[1]{\langle}{\rangle}{{#1}}
\DeclarePairedDelimiterX\norm[1]{\lVert}{\rVert}{{#1}}
```

\phfqit@insideinterval

Format the contents of an interval. Utility for defining \intervalc and friends.

```
\def\phfqit@insideinterval#1#2{{#1\mathclose{},\mathopen{}#2}}
```

\intervalc
\intervalo
\intervalco
\intervaloc

Open/Closed/Semi-Open Intervals

```
\DeclarePairedDelimiterX\intervalc[2]{[}{]}{\phfqit@insideinterval{#1}{#2}}
\DeclarePairedDelimiterX\intervalo[2]{]}{[}{\phfqit@insideinterval{#1}{#2}}
\DeclarePairedDelimiterX\intervalco[2]{[}{[}{\phfqit@insideinterval{#1}{#2}}
\DeclarePairedDelimiterX\intervaloc[2]{]}{]}{\phfqit@insideinterval{#1}{#2}}
```

7.4 Entropy Measures

7.4.1 Some Internal Utilities

`\phfqit@parsesizearg` Internal utility to parse size argument with the backtick specification (subsection 2.2).

Parses a size argument, if any, and stores it into `\phfqit@val@sizearg`. The value stored can directly be expanded as an optional argument to a `\DeclarePairedDelimiter`-compatible command (see mathtools package).

`#1` should be a command token. It is the next action to take, after argument has been parsed.

```
\def\phfqit@parsesizearg#1{%
  \begingroup%
  \mathcode`\`="0060\relax%
  \gdef\phfqit@val@sizearg{}%
  \@ifnextchar`{\phfqit@parsesizearg@withsize{#1}}{\endgroup#1}%
}
\def\phfqit@parsesizearg@withsize#1`#2{%
  \def\@tmp@arg{#2}%
  \def\@tmp@star{*}%
  \def\@tmp@endgroupandcontinue{\endgroup#1}%
  \ifx\@tmp@arg\@tmp@star\relax%
    \gdef\phfqit@val@sizearg{*}%
    \expandafter\@tmp@endgroupandcontinue%
  \else%
    \gdef\phfqit@val@sizearg{[#2]}%
    \expandafter\@tmp@endgroupandcontinue%
  \fi%
}
```

`\phfqit@inner@parens` Simple parenthesis-delimited expression, with `\DeclarePairedDelimiter`-compatible syntax. For example,

`\phfqit@inner@parens{`⟨*content*⟩`}` → `( `⟨*content*⟩` )`

`\phfqit@inner@parens*{`⟨*content*⟩`}` → `\left( `⟨*content*⟩` \right)`

`\phfqit@inner@parens[\big]{`⟨*content*⟩`}` → `\bigl( `⟨*content*⟩` \bigr)`

```
\DeclarePairedDelimiterX\phfqit@inner@parens[1]{(}{)}{#1}
```

7.4.2 Entropy, Conditional Entropy, and Entropy Function

See user documentation in subsection 6.1.

\HHSym The symbol used to designate an entropy measure (not relative).

```
\newcommand\HHSym{H}
```

\@HHbase Base macro for entropy macros.

USAGE: `\@HHbase{`⟨*H-symbol*⟩`}{`⟨*subscript*⟩`}{`⟨*superscript*⟩`}`⟨*size-spec*⟩ `[`⟨*state*⟩`] [`⟨*epsilon*⟩`]{`⟨*target system*⟩`}[`⟨*conditioning system*⟩`]`

The argument ⟨*size-spec*⟩ is optional, and is documented in subsection 2.2. For example ⟨*size-spec*⟩ = `‘*` or `‘\Big`.

This command is robust.

Examples:

`\@HHbase{\hat{H}}{\mathrm{max}}[\rho][\epsilon]{E}[X’]` →
$\hat{H}^{\epsilon}_{\max}(E\,|\,X')_{\rho}$

`\@HHbase{\hat{H}}{\mathrm{max}}‘*[\rho][\epsilon]{\bigotimes_i E}[X’]`
→ $\hat{H}^{\epsilon}_{\max}\left(\bigotimes_i E \,\middle|\, X'\right)_{\rho}$

`\@HHbase{\hat{H}}{\mathrm{max}}‘\big[\rho][\epsilon]{E}[X’]`
→ $\hat{H}^{\epsilon}_{\max}\big(E\,\big|\,X'\big)_{\rho}$

```
\def\@HHbase#1#2{%
  #1_{#2}%
  \@HHbase@parsesize%
}
\robustify\@HHbase
```

TODO: use our generic size parser, don’t duplicate code.... this is historical and I don’t dare change it without thorough testing:

```
\def\@HHbase@parsesize{%
  \begingroup\mathcode‘\‘="0060\relax%
  \gdef\HH@tmp@sizearg{}%
  \@ifnextchar‘\@HHbase@withsize\@HHbase@endgroupandparseinner%
}
\def\@HHbase@withsize‘#1{%
  \def\@tmp@arg{#1}%
  \def\@tmp@star{*}%
  \ifx\@tmp@arg\@tmp@star\relax%
    \gdef\HH@tmp@sizearg{*}%
    \expandafter\@HHbase@endgroupandparseinner%
  \else%
    \gdef\HH@tmp@sizearg{[#1]}%
    \expandafter\@HHbase@endgroupandparseinner%
  \fi%
```

```
}
\def\@HHbase@endgroupandparseinner{\endgroup\@HHbase@parseinner}
\newcommand\@HHbase@parseinner[1][]{%  arg: state
  \def\HH@tmpstore@state{#1}%
  \@HHbase@parseinner@%
}
\newcommand\@HHbase@parseinner@[2][]{% arg: epsilon and target system
  \def\HH@tmpstore@epsilon{#1}%
  \def\HH@tmpstore@system{#2}%
  \@HHbase@parseinner@@%
}
\newcommand\@HHbase@parseinner@@[1][]{% arg: conditioning system
  \def\HH@tmpstore@condsys{#1}%
  \@HHbase@do@inner%
}
\newtoks\HH@tmp@toks
\def\HH@addtoks#1\@HH@END@ADD@TOKS{\HH@tmp@toks=\expandafter{\the\HH@tmp@toks#1}}%
```

`\@HHbase@do@inner` Format the entropy measure. All information is stored in macros of the form `\HH@tmpstore@<FIELD>`. The base string (entropy symbol and subscript) have already been typeset.

```
\def\@HHbase@do@inner{%
```

Add the superscript:

```
  ^{\HH@tmpstore@epsilon}%
```

If system is blank, we just want the symbol itself with no argument. (`\notblank` is from the etoolbox package.) Otherwise, add the rest:

```
  \expandafter\notblank\expandafter{\HH@tmpstore@system}{%
```

Construct the parenthetic argument to the entropy, which we will store in the token register `\HH@tmp@toks`:

```
    \HH@tmp@toks={}%
```

… add system name:

```
    \expandafter\HH@addtoks\HH@tmpstore@system\@HH@END@ADD@TOKS%
```

… add conditional system, if specified:

```
    \expandafter\notblank\expandafter{\HH@tmpstore@condsys}{%
      \HH@addtoks\mathclose{}\,\delimsize\vert\,\mathopen{}\@HH@END@ADD@TOKS%
      \expandafter\HH@addtoks\HH@tmpstore@condsys\@HH@END@ADD@TOKS%
    }{}%
```

The tokens are ready now. Prepare the argument to the \phfqit@inner@parens command, and go:

```
    \edef\tmp@args{\expandonce{\HH@tmp@sizearg}{\the\HH@tmp@toks}}%
    \expandafter\phfqit@inner@parens\tmp@args%
```

Finally, add the state as subscript, if any:

```
    _{\HH@tmpstore@state}%
    %
  }{}%
  %
}
```

Now, we have the proper entropy commands.

\HH
\Hzero
\Hmin
\Hmaxf

The definition of individual entropy macros just delegates to \@HHbase with the relevant subscript.

```
\newcommand\HH{\@HHbase{\HHSym}{}}
\newcommand\Hzero{\@HHbase{\HHSym}{\mathrm{max},0}}
\newcommand\Hmin{\@HHbase{\HHSym}{\mathrm{min}}}
\newcommand\Hmaxf{\@HHbase{\HHSym}{\mathrm{max}}}
```

\Hfunc

Entropy function. Usage: \Hfunc(x), \Hfunc‘*(x), \Hfunc‘\big(x).

TODO: Use our generic size-specification parser! Don’t duplicate code!

```
\DeclareRobustCommand\Hfunc{%
  \begingroup\mathcode‘\‘="0060\relax%
  \gdef\Hfunc@tmp@sizearg{}%
  \@ifnextchar‘\Hfunc@withsize\Hfunc@next%
}
\def\Hfunc@withsize‘#1{%
  \def\@tmp@arg{#1}%
  \def\@tmp@star{*}%
  \ifx\@tmp@arg\@tmp@star\relax%
    \gdef\Hfunc@tmp@sizearg{*}%
    \endgroup%
    \expandafter\Hfunc@inner%
  \else%
    \gdef\Hfunc@tmp@sizearg{[#1]}%
    \endgroup%
    \expandafter\Hfunc@inner%
  \fi%
}
\def\Hfunc@next{\endgroup\Hfunc@inner}
\def\Hfunc@inner(#1){%
  \HHSym%  ({#1})%
  \expandafter\phfqit@inner@parens\Hfunc@tmp@sizearg{#1}%
}
```

7.4.3 Relative Entropies

User documentation in subsection 6.2

`\DDSym` Symbol to use to denote a relative entropy.

```
\newcommand\DDSym{D}
```

`\@DDbase@inner` Internal macro to format the inner contents of a relative entropy.

`\@DDbase@inner{\rho}{\Gamma}` $\rightarrow$ $\boxed{(\rho \,\|\, \Gamma)}$

You can also specify the optional size specifier compatible with the `\DeclarePariedDelimiter` syntax: `\@DDbase@inner*{\rho}{\Gamma}` and `\@DDbase@inner[\big]{\rho}{\Gamma}`, for example.

```
\DeclarePairedDelimiterX\@DDbase@inner[2]{(}{)}{%
  #1\mathclose{}\,\delimsize\Vert\,\mathopen{}#2%
}
```

`\@DDbase` Base macro for relative entropy macros.

USAGE: `\@DDbase{`⟨*D-symbol*⟩`}{`⟨*subscript*⟩`}{`⟨*superscript*⟩`}`⟨*size-spec*⟩`{`⟨*state*⟩`}{`⟨*relative to state*⟩`}`

The ⟨*size-spec*⟩ may be either a backtick-style specification, or a star or an optional argument ("`[\big]`").

Examples:

`\@DDbase{DSYMBOL}{subscript}{superscript}{\rho}{\Gamma}` $\rightarrow$ $DSYMBOL^{superscript}_{subscript}(\rho \,\|\, \Gamma)$, and similarly
`\@DDbase{DSYMBOL}{subscript}{superscript}*{\rho}{\Gamma}`,
`\@DDbase{DSYMBOL}{subscript}{superscript}[\big]{\rho}{\Gamma}`,
`\@DDbase{D-symbol}{subscript}{superscript}‘*{\rho}{\Gamma}`,
`\@DDbase{D-symbol}{subscript}{superscript}‘\big{\rho}{\Gamma}`.

This command is robust.

```
\def\@DDbase#1#2#3{%
  #1_{#2}^{#3}%
  \@DDbase@parsesize%
}
\robustify\@DDbase
\def\@DDbase@parsesize{%
  \@ifnextchar‘\@DDbase@withsize\@DDbase@inner%
}
\def\@DDbase@withsize‘#1{%
  \def\@tmp@arg{#1}%
  \def\@tmp@star{*}%
```

```
  \ifx\@tmp@arg\@tmp@star\relax%
    \def\tmp@cmd{\@DDbase@inner*}%
    \expandafter\tmp@cmd%
  \else%
    \def\tmp@cmd{\@DDbase@inner[#1]}%
    \expandafter\tmp@cmd%
  \fi%
}
```

`\DD` (Usual) quantum relative entropy. Actually this is more versatile, because you can also specify subscript and superscript.

```
\DeclareRobustCommand\DD{%
  \def\DD@tmp@sub{}%
  \def\DD@tmp@sup{}%
  \DD@%
}
\def\DD@{%
  \@ifnextchar_\DD@parsesub\DD@@%
}
\def\DD@@{%
  \@ifnextchar^\DD@parsesup\DD@@@%
}
\def\DD@@@{% sub/super-scripts have been parsed, move on to rest of command
  \@DDbase{\DDSym}{\DD@tmp@sub}{\DD@tmp@sup}%
}
\def\DD@parsesub_#1{%
  \def\DD@tmp@sub{#1}%
  \DD@% continue parsing maybe another sub or superscript
}
\def\DD@parsesup^#1{%
  \def\DD@tmp@sup{#1}%
  \DD@% continue parsing maybe another sub or superscript
}
```

`\Dminz` “Old” min-relative entropy, based on the Rényi-zero relative entropy.

```
\DeclareRobustCommand\Dminz[1][]{%
  \@DDbase{\DDSym}{\mathrm{min,0}}{#1}%
}
```

`\Dminf` Min-relative entropy (“new” version).

```

%
% \Dminf{\rho}{\sigma}
% \Dminf[\epsilon]{\rho}{\sigma}
% \Dminf<states-spec>
% \Dminf[\epsilon]<states-spec>
%
```

```
% Where <states-spec>  =  <size-spec>{\rho}{\sigma}
%
% Where optional  <size-spec>  =  "`*"  or  "`\Big"
%
\DeclareRobustCommand\Dminf[1][]{%
  \@DDbase{\DDSym}{\mathrm{min}}{#1}%
}
```

`\Dmax` Max-relative entropy.

```
\DeclareRobustCommand\Dmax[1][]{%
  \@DDbase{\DDSym}{\mathrm{max}}{#1}%
}
```

`\Dr` Rob-relative entropy.

```
\DeclareRobustCommand\Dr[1][]{%
  \@DDbase{\DDSym}{\mathrm{r}}{#1}%
}
```

`\DHyp` Hypothesis testing relative entropy.

```
\DeclareRobustCommand\DHyp[1][\eta]{%
  \@DDbase{\DDSym}{\mathrm{H}}{#1}%
}
```

7.4.4 Coherent Relative Entropy

See user documentation in subsection 6.3.

`\DC@inner` Format the contents of the coherent relative entropy. This is simply a `\DeclareParied Delimiter`-style command. The syntax is `\DC@inner{`$\langle$*rho*$\rangle$`}{`$\langle$*Gamma1*$\rangle$`}{`$\langle$*Gamma2*$\rangle$`}`, and this typesets as $\left(\langle \mathit{rho} \rangle \middle\| \langle \mathit{Gamma1} \rangle, \langle \mathit{Gamma2} \rangle\right)$.

```
\DeclarePairedDelimiterX\DC@inner[3]{(}{)}{%
  #1\mathclose{}\,\delimsize\Vert\,\mathopen{}#2\mathclose{},\mathopen{}#3%
}
```

`\DCSym` Symbol to use for the coherent relative entropy

```
\newcommand\DCSym{\bar\DDSym}
```

`\emptysystem` Designates the trivial system (uses symbol for empty set). It is important to this, because of the automatic indexes set on the “rho” argument.

```
\def\emptysystem{\ensuremath{\emptyset}}
```

\DCoh The Coherent Relative Entropy.

TODO: Use our generic size parser, don't duplicate code!

First part: read the first few arguments (epsilon superscript, optional size specification).

```
\newcommand\DCoh[1][]{%
  \def\DC@tmp@sup{#1}%
  %\message{*********|\detokenize{#1}|*********}%
  \begingroup\mathcode`\`="0060\relax
  \DC@parsesize%
}
\def\DC@parsesize#1{%
  \gdef\DC@tmp@sizeargs{}%
  \ifstrequal{#1}{`}\DC@withsize{\endgroup\DC@rest{#1}}%
}
\def\DC@withsize#1{%
  %\message{*********\detokenize{#1}********}%
  \def\@tmp@arg{#1}%
  \def\@tmp@star{*}%
  \ifx\@tmp@arg\@tmp@star\relax%
    \gdef\DC@tmp@sizeargs{*}%
    \endgroup%
    \expandafter\DC@rest%
  \else%
    \gdef\DC@tmp@sizeargs{[#1]}%
    \endgroup%
    \expandafter\DC@rest%
  \fi%
}
```

Read the rest and typeset the output. #1=rho, #2=system-in, #3=system-out, #4=Gamma-in, #5=Gamma-out:

```
\def\DC@rest#1#2#3#4#5{%
  %\message{*********\detokenize{#1}|\detokenize{#2}|\detokenize{#3}%
  %    |\detokenize{#4}|\detokenize{#5}|********}%
  \def\DC@tmp@rho{\DC@fmtrhosub#1\DC@ENDSTATE{#2}{#3}}%
  \DCSym_{#2\to #3}^{\DC@tmp@sup}%
  \expandafter\DC@inner\DC@tmp@sizeargs{\DC@tmp@rho}{#4}{#5}%
}
```

Read the following tokens until the marker \DC@ENDSTATE, and format this as a state with or without the automatic system subscripts (depending on if the argument starts with a '*').

```
\def\DC@fmtrhosub{%
  \@ifnextchar*\DC@fmtrhosub@nosub\DC@fmtrhosub@wsub%
}
\def\DC@fmtrhosub@nosub*#1\DC@ENDSTATE#2#3{%
```

```
  #1%
}
\def\DC@fmtrhosub@wsub#1\DC@ENDSTATE#2#3{%
  \begingroup%
    \let\emptysystem\relax%
    #1_{#3#2}%
  \endgroup%
}
```

Change History

v1.0

General: Initial version 1

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.

Symbols

\, ... 175, 214, 284
\@DDbase 216, 247, 258, 272, 275, 278, 281
\@DDbase@inner 213, 222, 228, 231
\@DDbase@parsesize 218, 221
\@DDbase@withsize 222, 224
\@HH@END@ADD@TOKS . 168, 173, 175, 176
\@HHbase 132, 185, 186, 187, 188
\@HHbase@do@inner 165, 169
\@HHbase@endgroupandparseinner 140, 147, 150, 153
\@HHbase@parseinner 153, 154
\@HHbase@parseinner@ 156, 158
\@HHbase@parseinner@@ 161, 163
\@HHbase@parsesize 134, 137
\@HHbase@withsize 140, 142
\@ifnextchar 116, 140, 192, 222, 241, 244, 320
\@tmp@arg 119, 122, 143, 145, 195, 197, 225, 227, 300, 302
\@tmp@args 22, 27
\@tmp@endgroupandcontinue 121, 124, 127
\@tmp@star 120, 122, 144, 145, 196, 197, 226, 227, 301, 302
\‘ 114, 138, 190, 291

A

\abs *6*, 104
\AND *5*, 68
\avg *6*, 104

B

\bar 286
\begingroup 50, 113, 138, 190, 291, 326
\Big 269
\bit *4*, 47, 65
\bitstring *4*, 47
\bra *6*, 82
\braket *6*, 82

C

\CNOT *5*, 68

D

\DC@ENDSTATE 315, 322, 325
\DC@fmtrhosub 315, 319
\DC@fmtrhosub@nosub 320, 322
\DC@fmtrhosub@wsub 320, 325
\DC@inner 283, 317
\DC@parsesize 292, 294
\DC@rest 296, 305, 309, 312
\DC@tmp@rho 315, 317
\DC@tmp@sizeargs . . 295, 303, 307, 317
\DC@tmp@sup 289, 316
\DC@withsize 296, 298
\DCoh *9*, 288
\DCSym *9*, 286, 316
\DD *7*, 235
\DD@ 238, 240, 251, 255
\DD@@ 241, 243
\DD@@@ 244, 246
\DD@parsesub 241, 249
\DD@parsesup 244, 253
\DD@tmp@sub 236, 247, 250

\DD@tmp@sup 237, 247, 254
\DDSym *8*, 212, 247, 258, 272, 275, 278, 281, 286
\DeclareMathOperator 34, 35, 36, 37, 38, 39, 41, 44, 46
\DeclarePairedDelimiterX 82, 83, 84, 87, 90, 93, 97, 101, 104, 105, 106, 108, 109, 110, 111, 130, 213, 283
\DeclareRobustCommand 66, 189, 235, 257, 271, 274, 277, 280
\delimsize 85, 88, 91, 94, 95, 98, 99, 175, 214, 284
\detokenize 57, 290, 299, 313, 314
\DHyp *8*, 280
\diag *4*, 34
\dmatrixel *6*, 82
\Dmax *8*, 274
\Dminf *8*, 260
\Dminz *8*, 257
\Dr *8*, 277

E

\ee 33
\ee^... 33
\ee^X *4*
\emptyset 287
\emptysystem *9*, 287, 327
\endgroup 53, 116, 121, 153, 199, 203, 207, 296, 304, 308, 329
\ensuremath 64, 287
\epsilon 263, 265
\eta 280
etoolbox 16
\expandafter 27, 31, 57, 124, 127, 147, 150, 168, 171, 173, 174, 176, 179, 200, 204, 210, 229, 232, 305, 309, 317
\expandonce 23, 24, 25, 178

G

\gate *5*, 66, 68, 69, 70, 71, 72

H

\Hfunc *7*, 189
\Hfunc@inner 200, 204, 207, 208
\Hfunc@next 192, 207
\Hfunc@tmp@sizearg 191, 198, 202, 210
\Hfunc@withsize 192, 194
\HH *7*, 185
\HH@addtoks 168, 173, 175, 176
\HH@tmp@sizearg ... 139, 146, 149, 178
\HH@tmp@toks 167, 168, 172, 178
\HH@tmpstore@condsys . 164, 174, 176
\HH@tmpstore@epsilon 159, 170
\HH@tmpstore@state 155, 180
\HH@tmpstore@system .. 160, 171, 173
\HHSym ... *7*, 131, 185, 186, 187, 188, 209
\Hmaxf *7*, 185
\Hmin *7*, 185
\Hs *3*, 7
\hspace 63, 85, 88, 91, 94, 95, 98, 99, 102
\Hzero *7*, 185

I

\Ident *3*, 8
\IdentProc *4*, 9
\ifmmode 66
\ifstrequal 296
\Im *4*, 40
\innerprod *6*, 82
\intervalc *6*, 108
\intervalco *6*, 108
\intervalo *6*, 108
\intervaloc *6*, 108

K

\ket *6*, 82
\ketbra *6*, 82

L

\langle .. 83, 84, 88, 91, 93, 97, 101, 105
\let 40, 43, 327
\linspan *4*, 34
\lowercase 66, 67
\lVert 106
\lvert 82, 87, 90, 104

M

\makebox 56
\mathclose 107, 175, 214, 284
\mathcode 114, 138, 190, 291
\mathds 8
\mathopen 107, 175, 214, 284
\mathrm 79, 80, 81, 186, 187, 188, 258, 272, 275, 278, 281
\mathscr 7
mathtools 14
\matrixel *6*, 82
\maxof 51
\message 290, 299, 313

N

\newlength 62
\newtoks 167
\NOOP *5*, 68
\norm *6*, 104
\NOT *5*, 68
\notblank 30, 31, 171, 174

O

\operatorname 30
\overline 64

P

packages:
  etoolbox 16
  mathtools 14
  phfqit 1
  phfqitltx 1
phfqit 1
\phfqit@bitstring 48, 49
\phfqit@bitstring@ 52, 55, 59
\phfqit@END 52, 55, 59
\phfqit@fmtGroup 74, 76, 78, 81
\phfqit@fmtLieAlgebra 73, 75, 77, 80
\phfqit@fmtLieGroup 80
\phfqit@IdentProc@arg 18, 20
\phfqit@IdentProc@do 27, 29
\phfqit@IdentProc@maybeA ... 10, 12
\phfqit@IdentProc@maybeB ... 14, 16
\phfqit@IdentProc@val@A 13, 23
\phfqit@IdentProc@val@arg .. 21, 25
\phfqit@IdentProc@val@B 17, 24
\phfqit@Im 43
\phfqit@Imagpart 44, 45
\phfqit@inner@parens 31, 130, 179, 210
\phfqit@insideinterval .....
..... 107, 108, 109, 110, 111
\phfqit@len@bit 51, 56, 62
\phfqit@parsesizearg 10, 112
\phfqit@parsesizearg@withsize
..... 116, 118
\phfqit@Re 40
\phfqit@Realpart 41, 42
\phfqit@val@sizearg 31, 115, 123, 126
\phfqitBitstringFormat 52, 63
\phfqitBitstringFormatBit .. 56, 65
\phfqitBitstringSep 59, 63
phfqitltx 1
\poly *4*, 46
\proj *6*, 82

R

\rangle .. 82, 84, 88, 91, 93, 97, 101, 105
\rank *4*, 34
\Re *4*, 40
\relax 57, 114, 122, 138,
145, 190, 197, 227, 291, 302, 327
\renewcommand 42, 45
\RequirePackage 1, 2, 3, 4, 5, 6
\rho 262, 263, 267
\rmfamily 67
\robustify 136, 220
\rVert 106
\rvert 83, 87, 90, 104

S

\setlength 51
\sigma 262, 263, 267
\SN 79
\SN(N) *5*, 73
\SO 78
\so 77
\SO(N) *5*, 73
\so(N) *5*, 73
\spec *4*, 34
\SU 76
\su 75
\SU(N) *5*, 73
\su(N) *5*, 73
\supp *4*, 34

T

\textsc 66, 67
\texttt 47
\the 168, 178
\tmp@args 178, 179
\tmp@cmd 228, 229, 231, 232
\to 30, 316
\tr *4*, 34

U

\underline 64
\UU 74
\uu 73
\UU(N) *5*, 73
\uu(N) *5*, 73

V

\Vert 214, 284
\vert 85, 94, 95, 98, 99, 175

W

\widthof 51

X

\XOR *5*, 68