THE REC-THY PACKAGE

PETER M. GERDES (GERDES@INVARIANT.ORG)

2024/03/29: Version 4.0

ABSTRACT. The rec-thy package is designed to help mathematicians publishing papers in the area of recursion theory (aka Computability Theory) easily use standard notation. This includes easy commands to denote Turing reductions, Turing functionals, r.e. sets, stagewise computations, forcing and syntactic classes.

1. INTRODUCTION

This package aims to provide a useful set of LaTeX macros covering basic computability theory notation. Given the variation in usage in several areas this package had to pick particular notational conventions. The package author would like to encourage uniformity in these conventions but has included a multitude of package options to allow individual authors to choose alternative conventions or exclude that part of the package. Some effort has been made to align the semantic content of documents created with this package with the LaTeX source. The author hopes that eventually this package may be incorporated into some larger package for typesetting papers in mathematical logic.

While computability theory is now the more popular name for the subject also known as recursion theory the author deliberately choose to title this package rec-thy to avoid confusion with the proliferation of packages for typesetting computer science related disciplines. While the subject matter of computability theory and theoretical computer science overlap significantly the notational conventions often differ.

Comments, patches, suggestions etc.. are all welcome. This project is hosted on github at https://github.com/TruePath/Recursion-Theory-Latex-Package.

2. USAGE

Include the package in your document by placing `\usepackage{rec-thy}` into your preamble after placing rec-thy.sty somewhere TeX can find it. The commands in this package have been divided into related groups. The commands in a given section can be disabled by passing the appropriate package option. For instance to disable the commands in the general mathematics section and the delimiters section you would include the following in your preamble `\usepackage[nomath,nodelim]{rec-thy}`. The commands in each subsection along with their results are listed below and the

options to disable the commands in each grouping or modify their behavior are listed in that subsection. Aliases and variants of a command are listed below the initial version of a command and aliases are indented.

Significant use is made in this package of optional arguments delimited either by square brackets or parenthesis. Users of the package should take care to wrap arguments that may themselves include brackets or parenthesis in braces. For example `\REset(\REset(X){e}){i}` should be fixed to `\REset({\REset(X){e}}){i}`.

3. Alternate Symbols

While the symbols used by default in the package are suggested for adoption to achieve greater consistency users may wish to specify their own symbols and options have been provided to enable this. The following parameters may be specified.

modulescr Sets the script used to typeset the `\module` command. Default is mathcal.

reqscr Sets the script used to typeset requirements. Default is mathscr.

pfcasefont Sets the script used to typeset Case in the cases helper.

emptystr Sets the empty string symbol. Default is $\langle\rangle$

concatsym Sets the concat symbol. Default is ˆ

recfnlsym Sets the symbol used for recursive functionals. Default is Φ.

recfsym Sets the symbol used for recursive functions. Default is ϕ.

usesym Sets the symbol used for the use operator. Default is $\mathfrak{u}$ where this is printed using `\symbfrak` if unicode-math is loaded and with `\mathfrak` otherwise.

lstrdelim,rstrdelim Left and right delimiters for `\str`

lcodedelim,rcodedelim Left and right delimiters for `\code`

lpairdelim,rpairlim,pairsup Left and right delimiters for `\pair` and superscript.

altcompat Uses the `\succprec` and `\nsuccprec` commands for `\compat` and `\incompat` respectively.

4. Commands

A few general conventions are usually followed in the commands. Whenever an operator can be used as a binary operator (as in $X \cup Y$) and as an operation on some collection $\bigcup_{i \in \omega} X_i$ the binary operator will begin with a lowercase letter `\union` and the operation on the collection will begin with a capital letter `\Union`. If the first letter is already capitalized then the second letter is used instead.

Objects that have a natural stagewise approximation generally admit an optional argument in brackets to specify a stage. For instance `\REset[s]{e}` yields $W_{e,s}$. An

optional argument in parenthesis is used for relativization. For instance `\REset(X){e}`
produces W_e^X. For the formula classes square brackets are used to indicate an oracle
to be placed in the superscript, e.g., `\pizn[X]{2}` yields $\Pi_2^{0,X}$ while parenthesis
place the value in parenthesises after the symbol as in $\Pi_2^0(X)$. To give boldcase versions of formulas classes a star should be used immediately following the command
as in $\mathbf{\Pi}_2^0(X)$.

Unless indicated otherwise all macros are to be used inside math mode. Indented
commands indicate an alias for the command on the line above.

4.1. **Priority Trees.** Note the commands in this section double as suggestions for
standardized notation.

`\PriorityTree`	$\mathbb{T}$	Priority Tree
`\tpath`	$\mathbb{f}$	Truepath.
`\tpath[s]`	$\mathbb{f}_s$	Approximate Truepath.
`\xi \leftof \eta`	$\xi <_L \eta$	The left of relation for priority tree arguments
`\module{R}[X]{i,j}`	$\mathcal{R}_{i,j}^X$	Module in tree construction (superscript optional arg)

4.2. **Computations.** To disable these commands pass the option `nocomputations`.
To specify an alternative symbol for either recursive functionals or recursive functions pass `recfnlsym=macroname` or `recfsym=macroname`. For instance, to use
ψ for recursive functions and Ψ for recursive functionals pass the options `recfsym=psi, recfnlsym=Psi`.

`\murec{x}{f(x)>1}`	$\mu x\,(f(x) > 1)$	Least x satisfying condition.
`\recf{e}`	ϕ_e	
`\recf[s]{e}`	$\phi_{e,s}$	Computable functions
`\recf{e}(x)`	$\phi_e(x)$	
`\recf[s]{e}(x)`	$\phi_{e,s}(x)$	
`\recf(Y){e}`	ϕ_e^Y	
`\recf[s](Y){e}`	$\phi_{e,s}^Y$	
`\recf(Y){e}(x)`	$\phi_e^Y(x)$	
`\recf[s](Y){e}(x)`	$\phi_{e,s}^Y(x)$	
`\recfnl{e}{Y}{x}`	$\Phi_e(Y; x)$	
`\recfnl[s]{e}{Y}{x}`	$\Phi_{e,s}(Y; x)$	Computable functionals

`\recfnl{e}{Y}{}`	$\Phi_e(Y)$	
`\recfnl{e}{}{x}`	Φ_e	
`\recfnl{e}{}{}`	Φ_e	
`\recfnl{e}{}{} \cequiv \recfnl{i}{}{}`	$\Phi_e \simeq \Phi_i$	Equivalent computations
`\recfnl{e}{}{} \ncequiv \recfnl{i}{}{}`	$\Phi_e \not\simeq \Phi_i$	Inequivalent computations
`\recfnl{e}{}{x}\conv`	$\Phi_e\downarrow$	Convergence
`    \recfnl{e}{}{x}\conv[s]`	$\Phi_e\downarrow_s$	
`\recfnl{e}{}{x}\nconv`	$\Phi_e\not\downarrow$	Divergence
`    \recfnl{e}{}{x}\nconv[s]`	$\Phi_e\not\downarrow_s$	
`\use{\recfnl{e}{Y}{x}}`	$\mathbf{u}\left[\Phi_e(Y;x)\right]$	Use of a computation.
`\REset{e}`	W_e	c.e. sets
`\REset[s]{e}`	$W_{e,s}$	
`\REset(X){e}`	W_e^X	
`\REset[s](X){e}`	$W_{e,s}^X$	
`\Hop{e}(X)`	$\mathcal{H}_e(X)$	Hop sets
`\Hop[s](X){e}`	$\mathcal{H}_{e,s}(X)$	
`\Hop[s]{e}`	$\mathcal{H}_{e,s}(\varnothing)$	
`\Hop{e}`	$\mathcal{H}_e(\varnothing)$	
`\REAop{e}{\alpha}`	$\mathcal{J}_e^\alpha$	

4.3. **Degrees.** To disable these commands pass the option `nodegrees`.

`\Tdegrees` `    \strcD`	$\mathcal{D}$	The structure of $\mathcal{P}(\omega)$ under $\leq_{\mathbf{T}}$ alternate command
`\REdegrees` `    \strcR`	$\mathcal{R}$	The structure of r.e sets under $\leq_{\mathbf{T}}$ alternate command
`\Adegrees` `    \strcDa`	$\mathcal{D}_{\mathbf{a}}$	The structure of $\mathcal{P}(\omega)$ under $\leq_{\mathbf{a}}$ alternate command
`\AREAdegrees` `    \strcRa`	FOO	The structure of the ω-REA sets under $\leq_{\mathbf{a}}$ alternate command
`\Tdeg{d}`	$\mathbf{d}$	indicated that the variable/entity is a Turing degree
`\Tjump{X}`	X'	Turing jump

`\jump{X}`		
`\jjump{X}`	X''	
`\jumpn{X}{n}`	$X^{(n)}$	
`\Tzero`	$\mathbf{0}$	Computable degree
`\zeroj`	$\mathbf{0}'$	
`\zerojj`	$\mathbf{0}''$	
`\zerojjj`	$\mathbf{0}'''$	
`\zeron{n}`	$\mathbf{0}^{(n)}$	
`X \Teq Y`	$X \equiv_{\mathbf{T}} Y$	Turing equivalence
`X \nTeq Y`	$X \not\equiv_{\mathbf{T}} Y$	Turing inequivalence
`X \Tlneq Y`	$X \lneq_{\mathbf{T}} Y$	
`X \Tleq  Y`	$X \leq_{\mathbf{T}} Y$	
`X \Tgneq Y`	$X \gneq_{\mathbf{T}} Y$	
`X \Tgeq  Y`	$X \geq_{\mathbf{T}} Y$	
`X \Tgtr  Y`	$X >_{\mathbf{T}} Y$	
`X \Tless Y`	$X <_{\mathbf{T}} Y$	
`X \nTleq Y`	$X \nleq_{\mathbf{T}} Y$	
`X \nTgeq Y`	$X \ngeq_{\mathbf{T}} Y$	
`X \Tincompat Y`	$X \mid_{\mathbf{T}} Y$	Turing incompatibility
`X \Tcompat Y`	$X \cancel{\mid}_{\mathbf{T}} Y$	Turing compatibility
`\Tdeg{d} \join \Tdeg{d'}` `\Join_{i \in \omega} \Tdeg{d_i}`	$\mathbf{d} \vee \mathbf{d}'$ $\bigvee_{i \in \omega} \mathbf{d_i}$	Join of degrees
`\Tdeg{d} \meet \Tdeg{d'}` `\Meet_{i \in \omega} \Tdeg{d_i}`	$\mathbf{d} \wedge \mathbf{d}'$ $\bigwedge_{i \in \omega} \mathbf{d_i}$	Meet of degrees (when defined)
`X \Tplus Y`	$X \oplus Y$	Effective join of sets (alias for `\oplus`)
`\TPlus_{i \in \omega} X_i` `\Oplus_{i \in \omega} X_i`	$\bigoplus_{i \in \omega} X_i$	Effective join of sets alternative name
`X \ttlneq Y`	$X \lneq_{\mathrm{tt}} Y$	Truth table reducibilities
`X \ttleq  Y`	$X \leq_{\mathrm{tt}} Y$	
`X \ttgneq Y`	$X \gneq_{\mathrm{tt}} Y$	
`X \ttgeq  Y`	$X \geq_{\mathrm{tt}} Y$	
`X \ttgtr  Y`	$X >_{\mathrm{tt}} Y$	
`X \ttless Y`	$X <_{\mathrm{tt}} Y$	
`X \ttnleq Y`	$X \nleq_{\mathrm{tt}} Y$	
`X \ttngeq Y`	$X \ngeq_{\mathrm{tt}} Y$	
`\Azero`	$\mathbf{0_a}$	Arithmetic degree

`\Azeroj`	$\mathbf{0_a}'$	
`\Azerojj`	$\mathbf{0_a}''$	
`\Azerojjj`	$\mathbf{0_a}'''$	
`\Azeron{n}`	$\mathbf{0_a}^{(n)}$	
X `\Aeq` Y	$X \equiv_a Y$	Arithmetic equivalence
X `\nAeq` Y	$X \not\equiv_a Y$	Arithmetic inequivalence
X `\Alneq` Y	$X \lneq_a Y$	
X `\Aleq` Y	$X \leq_a Y$	
X `\Agneq` Y	$X \gneq_a Y$	
X `\Ageq` Y	$X \geq_a Y$	
X `\Agtr` Y	$X >_a Y$	
X `\Aless` Y	$X <_a Y$	
X `\nAleq` Y	$X \nleq_a Y$	
X `\nAgeq` Y	$X \ngeq_a Y$	
X `\Aincompat` Y	$X \mid_a Y$	Turing incompatibility
X `\Acompat` Y	$X \!\not\mid_a Y$	Turing compatibility

4.4. **Requirement Assistance.** To disable these commands pass the option `noreqhelper`.
To disable the hyperlinked requirements pass `nohyperreqs`
Math mode is not required for `\req{R}{e}`

`\req{R}{e}` `\req{R}[\nu]{e}`	$\mathscr{R}_e$ $\mathscr{R}_e^\nu$	Requirement
`\req*{R}{e}` `\req*{R}[\nu]{e}`	$\mathscr{R}_e$ $\mathscr{R}_e^\nu$	Requirement without hyperlinks

We also introduce the following environments for introducing requirements. The requirement environment is used as follows

```
\begin{requirement}{\req{R^{*}}{r,j}}
    \recfnl{r}{B}{} = \REset{j} \implies
        \exists[k] \left( \Upsilon^{j}_k(
      C \Tplus \REset{j}) = B) \lor \
      REset{j} \Tleq \Tzero \right)
\end{requirement}
```

Giving output

$$\mathscr{R}_{r,j}^*: \quad \Phi_r(B) = W_j \implies [\exists k]\Big(\Upsilon_k^j(C \oplus W_j) = B) \vee W_j \leq_{\mathbf{T}} \mathbf{0}\Big)$$

The require environment merges the `\req[\nu]{R}{e}` command directly into the environment arguments. It also creates an automatic label which makes use of the 1st and 2nd arguments but assumes the third argument contains only indexes whose names are subject to change. Unless `nohyperreqs` is passed the `\req[\nu]{R}{e}` automatically links to the defining require environment.

```
\begin{require}{R}{i}
        \recfnl{i}{B}{} = \REset{i} \implies    \exists
           [k] \left( \Upsilon^{i}_k(C \Tplus \REset{i
           }) = B) \lor \REset{j} \Tleq \Tzero \right)
\end{require}
```

Giving output

$$\mathscr{R}_i: \quad \Phi_i(B) = W_i \implies [\exists k]\Big(\Upsilon_k^i(C \oplus W_i) = B) \vee W_j \leq_{\mathbf{T}} \mathbf{0}\Big)$$

To list multiple requirements at once without introducing unnecessary spaces we also introduce the requirements environment which can be used as follows. Note that one may *not* end the final line with a linebreak or an undesired tag will appear.

```
\begin{requirements}
\require{P}[A]{e} A \neq \recfnl{i}{}{} \\
\require{P}[B]{e} B \neq \recfnl{i}{}{} \\
\require{R}{i,\hat{i},j,\hat{j}} \recfnl{i}{A}{} = \
   hat{A} \land \recfnl{\hat{i}}{\hat{A}}{} = A \land
    \recfnl{j}{B}{} = \hat{B} \land \recfnl{\hat{j}}{\
   hat{B}}{} = B \implies \Gamma(\hat{A} \symdiff \hat
   {B}) = A \Tplus B
\end{requirements}
```

$$\mathscr{P}_e^A: \quad A \neq \Phi_i$$
$$\mathscr{P}_e^B: \quad B \neq \Phi_i$$
$$\mathscr{R}_{i,\hat{i},j,\hat{j}}: \quad \Phi_i(A) = \hat{A} \wedge \Phi_{\hat{i}}(\hat{A}) = A \wedge \Phi_j(B) = \hat{B} \wedge \Phi_{\hat{j}}(\hat{B}) = B \implies \Gamma(\hat{A} \mathbin{\triangle} \hat{B}) = A \oplus B$$

4.5. **General Math Commands.** To disable these commands pass the option `nomath`.

`\eqdef`	$\overset{\text{def}}{=}$	Definitional equals
`\iffdef`	$\overset{\text{def}}{\iff}$	Definitional equivalence
`\aut`	Aut	Automorphisms of some structure
`\Ord`	Ord	Set of ordinals
`\abs{x}`	$\lvert x \rvert$	Absolute value
`\dom`	dom	Domain
`\rng`	rng	Range
`f\restr{X}`	$f \restriction_X$	Restriction
`\ordpair{x}{y}`	(x, y)	Ordered Pair
`f \compfunc g` `    f \compose g`	$f \circ g$	Function composition
`f: X \pmapsto Y`	$f : X \rightharpoonup Y$	partial function from X to Y.
`f: Y \pmapsfrom X`	$f : Y \leftharpoonup X$	partial function from X to Y.
`f: X \fpmapsto Y`	$f : X \rightharpoonup_{<\infty} Y$	finite partial function from X to Y.
`f: Y \fpmapsfrom X`	$f : Y \leftharpoonup_{<\infty} X$	finite partial function from X to Y.
`\ParFuncs{X}{Y}`	$Y^{\subset X}$	set of partial functions from X to Y.
`\FinParFuncs{X}{Y}`	$Y^{\subset_{<\infty} X}$	set of finite partial functions from X to Y.
`\( \ensuretext{blah} \)` `    \ensuretext{blah}`	blah	Types argument in text mode
`\quotient{X}{Y}`	X / Y	

4.6. **Operators.** Misc operators used in logic and computability. To disable these commands pass the option `nooperators`.

`x \meet y`	$x \wedge y$	Meet operation
`\Meet_{i\in \omega} x_i`	$\bigwedge_{i\in\omega} x_i$	
`x \join y`	$x \vee y$	Join operation
`\Join_{i\in \omega} x_i`	$\bigvee_{i\in\omega} x_i$	
`x \xor y`	$x \text{ xor } y$	
`\Land \phi_i`	$\bigwedge \phi_i$	Operator form of and
`\Lor \phi_i`	$\bigvee \phi_i$	Operator form of or
`\LLand \phi_i`	$\bigwedge\!\!\!\bigwedge \phi_i$	Infinitary conjunction
`\LLor \phi_i`	$\bigvee\!\!\!\bigvee \phi_i$	Infinitary disjunction

4.7. **Set Notation.** To disable these commands pass the option `nosets`.

Note that `\Cross` and `\cross` overwrite the existing commands saving them as `\CrossOrig` and `\crossOrig` respectively as these commands have varying meanings between different font packages. However, the other commands avoid overwriting existing commands with that name

`\set{(x,y)}{x > y}`	$\{(x, y) \mid x > y\}$	Set notation
`\set{(x,y)}`	$\{(x, y)\}$	
`\card{X}`	$\lvert X \rvert$	Cardinality
`X \union Y`	$X \cup Y$	Union
`\Union_{i \in \omega} X_i`	$\bigcup_{i \in \omega} X_i$	
`X \isect Y`	$X \cap Y$	Intersection
`\Isect_{i \in \omega} X_i`	$\bigcap_{i \in \omega} X_i$	
`X \cross Y`	$X \times Y$	Cartesian product (Cross Product)
`\Cross_{i \in \omega} X_i`	$\Pi_{i \in \omega} X_i$	
`\powset{\omega}`	$\mathcal{P}(\omega)$	Powerset
`\powset[\alpha]{\omega}`	$\mathcal{P}_\alpha(A)$	Subsets of A of size $< \alpha$
`\eset`	$\varnothing$	Emptyset abbreviation
`x \nin A`	$x \notin A$	not an element
`\setcmp{X}`	$\overline{X}$	Set compliment
`X \symdiff Y`	$X \,\Delta\, Y$	Symmetric difference

4.8. **Improved Symbols and MnSymbol Imports.** Unless the option `nosymb` is passed a number of symbols are (re)defined for better typesetting. To prevent only redefinition of `\llangle` and `\rrangle` (e.g. to use the XITS unusual default) pass the option `nodoubleangles`. A couple new symbols which can't be easily created without the MnSymbol font are also defined and are listed below.

`\Searrow`	$\searrow\!\!\!\searrow$	double searrow
`\nSearrow`	$\nsearrow\!\!\!\searrow$	negated double searrow

4.9. **Quantifiers.** To disable these commands pass the option `noquants`. The commands `\exists` and `\forall` are standard but the package extends them.

`\exists[x < y]`	$[\exists x < y]$	
`\exists(x < y)`	$(\exists x < y)$	
`\exists*` `\existsinf`	$\exists^\infty$	
`\exists*[x < y]`	$\left[\exists^\infty x < y\right]$	
`\exists*(x < y)`	$(\exists^\infty x < y)$	
`\nexists[x < y]`	$\left[\nexists x < y\right]$	
`\nexists(x < y)`	$\left(\nexists x < y\right)$	
`\nexists*` `\nexistsinf`	$\nexists^\infty$	
`\nexists*[x < y]`	$\left[\nexists^\infty x < y\right]$	
`\nexists*(x < y)`	$\left(\nexists^\infty x < y\right)$	
`\forall[x < y]`	$[\forall x < y]$	
`\forall(x < y)`	$(\forall x < y)$	
`\forall*` `\forallae`	$\forall^*$	For almost all.
`\forall*[x < y]`	$\left[\forall^* x < y\right]$	
`\forall*(x < y)`	$\left(\forall^* x < y\right)$	

4.10. **Spaces.** To disable these commands pass the option `nospaces`.

`\bstrs`	$2^{<\omega}$	Finite binary strings
`\cbstrs`	$2^{<\omega<\omega}$	Finite sequence of binary strings (regarded as columnise specification)
`\wstrs`	$\omega^{<\omega}$	Finite sequences of integers
`\cantor`	2^ω	Cantor space
`\baire`	ω^ω	Baire space
`\Baire`	$\mathcal{N}$	Alternate baire space
`\bpfuncs`	$\{0, 1, \uparrow\}^{<\omega}$	Finite binary partial functions

4.11. **Strings.** To disable these commands pass the option `nostrings`.

To specify an alternative symbol for the empty string pass `emptystr=macroname`.
For instance, to use λ as the empty string pass `emptystr=lambda`. Similarly, to

specify an alternate value for the concatenation symbol pass `concatsym=macroname`, e.g., if you specify `\def\plus{+}` and then pass `concatsym=plus` the concatenation symbol would be changed to +.

To use `\succprec` and `\nsuccprec` for `\compat` and `\incompat` pass the option `altcompat`

`\str{1,0,1}` `\code{5,8,13}`	$\langle 1,0,1 \rangle$ $\langle\langle 5,8,13 \rangle\rangle$	Strings/Codes for strings
`\EmptyStr` `\estr`	$\langle\rangle$ $\langle\rangle$	Empty string
`\godelnum{\phi}` `\cornerquote{\phi}`		Godel Code/Corner Quotes
`\decode{\sigma}{3}`	$(\sigma)_3$	Alternate notation for $\sigma(3)$
`\sigma\concat\tau` `\sigma\concat[0]`	$\sigma^\frown \tau$ $\sigma^\frown \langle 0 \rangle$	Concatenation
`\strpred{\sigma}`	σ^-	The immediate predecessor of σ
`\lh{\sigma}`	$\|\sigma\|$	Length of σ
`\sigma \incompat \tau` `\sigma \compat \tau`	$\sigma \mid \tau$ $\sigma \nmid \tau$	Incompatible strings Compatible strings
`\sigma \nsuccprec \tau` `\sigma \succprec \tau`	$\sigma \nsuccapprox \tau$ $\sigma \succapprox \tau$	Alternate incompatible strings Alternate compatible strings
`\pair{x}{y}`	$\langle x,y \rangle$	Code for the pair (x,y)
`\setcol{X}{n}`	$X^{[n]}$	$\{ y \mid \langle n,y \rangle \in X \}$
`\setcol{X}{\leq n}`	$X^{[\leq n]}$	$\{ \langle x,y \rangle \mid \langle x,y \rangle \in X \land x \leq n \}$

4.12. **Subfunctions.** To disable these commands pass the option `nosubfuns`.

`f \subfun g`	$f \prec g$	Varieties of the function extension relation
`f \supfun g`	$f \succ g$	
`f \nsubfun g`	$f \nprec g$	
`f \nsupfun g`	$f \nsucc g$	
`f \subfuneq g`	$f \preceq g$	
`f \subfunneq g`	$f \precneqq g$	
`f \supfuneq g`	$f \succeq g$	
`f \supfunneq g`	$f \succneqq g$	
`f \nsubfuneq g`	$f \npreceq g$	
`f \nsupfuneq g`	$f \nsucceq g$	

4.13. **Trees.** To disable these commands pass the option `notrees`.

`\CBderiv{T}`	$T^{\langle 1 \rangle}$	Cantor-Bendixson Derivative
`\CBderiv[\alpha]{T}`	$T^{\langle \alpha \rangle}$	
`\pruneTree{T}`	$T^{\langle \infty \rangle}$	$\{\sigma \in T \mid (\exists g)(g \in [T] \wedge \sigma \subset g)\}$
`\hgt{T}`	$\|T\|$	Tree Height
`\TreeMod{T}{\sigma}`	T/σ	$\{\tau \mid \sigma \hat{\ } \tau \in T\}$
`\sigma \TreeMul T`	$\sigma * T$	$\{\sigma \hat{\ } \tau \mid \tau \in T\}$

4.14. **Set Relations.** To disable these commands pass the option `nosetrels`.
Note that many of these commands are extensions of existing commands.

`X \subset* Y`	$X \subset^* Y$	All but finitely much of X is in Y
`X \subseteq* Y`	$X \subseteq^* Y$	
`X \supset* Y`	$X \supset^* Y$	All but finitely much of Y is in X
`X \supseteq* Y`	$X \supseteq^* Y$	
`X \eq Y`	$X = Y$	Macro for =
`X \eq* Y` `X \eqae Y`	$X =^* Y$	Equal mod finite
`X \infsubset Y`	$X \subset_\infty Y$	$X \subset Y \wedge |Y \smallsetminus X| = \omega$
`X \infsubset* Y`	$X \subset_\infty^* Y$	$X \subset^* Y \wedge |Y \smallsetminus X| = \omega$
`X \infsupset Y`	$X \supset_\infty Y$	$Y \subset X \wedge |X \smallsetminus Y| = \omega$
`X \infsupset* Y`	$X \supset_\infty^* Y$	$Y \subset^* X \wedge |X \smallsetminus Y| = \omega$
`X \majsubset Y`	$X \subset_m Y$	X is a major subset of Y
`X \majsupset Y`	$X \supset_m Y$	Y is a major subset of X

4.15. **Ordinal Notations.** To disable these commands pass the option `noordinalnotations`. Note the name for addition is `\Oadd` rather than `\Oplus` to avoid a collusion with existing commands.

`\wck`	ω_1^{CK}	First non-computable ordinal
`\Oabs{\alpha}` `    \Ohgt{\alpha}`	$\|\alpha\|_{\mathcal{O}}$	The height of the notation or well-ordering α (i.e. ordinal it den Alternative command
`\kleeneO`	$\mathcal{O}$	Set of ordinal notations
`    \kleeneO-`	$\vec{\mathcal{O}}$	Set of limit notations
`    \kleeneO+`	$^{+}\mathcal{O}$	Set of successor notations
`    \kleeneO* or \Ouniq`	$\mathcal{O}_1$	Canonical unique set of ordinal notations
`    \kleeneO(X)`	$\mathcal{O}^X$	Relativized ordinal notations
`    \kleeneO[\alpha]`	$\mathcal{O}_\alpha$	Ordinal notations for ordinals $< |\alpha|$
`    \kleeneO*(X)[\alpha]`	$\mathcal{O}^X_{1,\,\alpha}$	Multiple options work together
`\alpha \Oless \beta` `    \alpha \Oleq  \beta`	$\alpha <_{\mathcal{O}} \beta$ $\alpha \leq_{\mathcal{O}} \beta$	less than on the ordering on notations

`    \alpha \Ogtr   \beta`
 `    \alpha \Ogeq   \beta`
 `    \alpha \Onless \beta`
 `    \alpha \Onleq  \beta`
	$\alpha >_{\mathcal{O}} \beta$
	 $\alpha \geq_{\mathcal{O}} \beta$
	 $\alpha \not<_{\mathcal{O}} \beta$
	 $\alpha \not\leq_{\mathcal{O}} \beta$
		less than on the ordering on notations

`    \alpha \Ongtr   \beta` `    \alpha \Ongeq   \beta`	$\alpha \not>_{\mathcal{O}} \beta$ $\alpha \not\geq_{\mathcal{O}} \beta$	
`\alpha \Oadd \beta`	$\alpha +_{\mathcal{O}} \beta$	Effective addition of notations
`\alpha \Omul  \beta`	$\alpha \times_{\mathcal{O}} \beta$	Effective multiplication of notations
`\Olim{\lambda}{n}`	$\lambda_{[n]}$	The n-th element in effective limit defining notation λ
`\Opred{\alpha}`	α^-	Predecessor of α if defined
`\Ofunc{\gamma}(m)`	$\{\gamma\}^{\mathcal{O}}(m)$	The m-th element in the effectively given limit defining the limi

4.16. **Forcing.** To disable these commands pass the option `noforcing`.

`\sigma \forces \phi`	$\sigma \Vdash \phi$	σ forces ϕ
`\sigma \forces(X) \phi`	$\sigma \Vdash_T^X \phi$	ϕ is formula relative to X
`\sigma \forces(X)[T] \phi`	$\sigma \Vdash_T^X \phi$	Local forcing on T relative to X
`\sigma \forces* \phi`	$\sigma \Vdash^* \phi$	Strong forcing
`\sigma \wforces \phi`	$\sigma \Vdash^w \phi$	weak forcing (takes optional paren and bracket arguments as abo

4.17. **Syntax.** To disable these commands pass the option `nosyntax`.
All syntax classes can be relativized with an optional argument in square brackets
even when not listed below. Only the Δ formula classes are listed below since the
syntax is identical for Σ and Π. Adding a star after the command produces the bold-
face version. Do to the common usage of the zero and one versions of the formulas
an abbreviation is provided unless the option `noshortsyntax` is passed.

`\DeltaN{2}`	Δ_2	
`\DeltaN(X){2}`	$\Delta_2(X)$	
`\DeltaN[X]{2}`	Δ_2^X	
`\DeltaN[X]{2}`	$\boldsymbol{\Delta}_2^X$	boldface
`\DeltaZeroN{2}`	Δ_2^0	
`\DeltaZeroN(X){2}`	$\Delta_2^0(X)$	
`\DeltaZeroN[X]{2}`	$\Delta_2^{0,X}$	
`\DeltaZeroN[X]{2}`	$\boldsymbol{\Delta}_2^{0,X}$	boldface
`\deltazn(X){2}`	$\Delta_2^0(X)$	Abbreviated form
`\DeltaOneN{2}`	Δ_2^1	
`\DeltaOneN(X){2}`	$\Delta_2^1(X)$	
`\DeltaOneN[X]{2}`	$\Delta_2^{1,X}$	
`\DeltaOneN[X]{2}`	$\boldsymbol{\Delta}_2^{1,X}$	boldface
`\deltain(X){2}`	$\Delta_2^1(X)$	Abbreviated form
`\logic{\omega_1}{\omega}`	$\mathcal{L}_{\omega_1,\omega}$	Indicates the kind of infinitary logic

4.18. **Proof Cases.** The pfcases enviornment provides a numbered, referenceable division of a proof segment into cases. Pass the option nopfcases to disable loading of this feature. Note these features are not available if you are using the package in a beamer document (not likely to be an issue).

To see how these commands work consider the following code.

```
\begin{proof}
\begin{pfcases}
\case[\( x = y \)]\label{case:first} %
 Lorem ipsum dolor sit amet, consectetur adipiscing elit. In et enim eget nisl luctus

 Vestibulum semper eget velit ut lobortis. In vel finibus est. Nullam tellus dolor, p

\case[\( x = z \land z > q \land z < r \land x + z = r \)] \label{case:second} %
 Quisque consectetur, felis non congue dictum, mauris mi suscipit sem, vel laoreet ju

\begin{pfcases}
\case[\( x=2 \)] \label{case:second:sub1} %
 ipsum dolor sit amet, consectetur adipiscing elit. In et enim eget nisl luctus venen
\case[\( x = 3 \)] \label{case:second:sub2} %
 consectetur adipiscing elit. In et enim eget nisl luctus venenatis. Pellentesque sed
\end{pfcases}

\end{pfcases}
\end{proof}
We can now reference the case number \ref{case:second} and cleveref \cref{case:second
```

This produces the following output.

Proof. CASE 1 $x = y$**:**   Lorem ipsum dolor sit amet, consectetur adipiscing elit. In et enim eget nisl luctus venenatis. Pellentesque sed erat sodales, tincidunt quam non, eleifend risus. Fusce aliquam dignissim pharetra. Integer id dui ac libero tincidunt consectetur. Sed laoreet nunc nec semper laoreet.

Vestibulum semper eget velit ut lobortis. In vel finibus est. Nullam tellus dolor, pellentesque sed orci sed, ornare pretium diam. Nam vel tincidunt tellus. Nulla at mi nisl.

CASE 2 $x = z \land z > q \land z < r \land x + z = r$**:**   Quisque consectetur, felis non congue dictum, mauris mi suscipit sem, vel laoreet justo ipsum in tellus. Suspendisse blandit malesuada velit faucibus pulvinar.

CASE 2a $x = 2$**:**   ipsum dolor sit amet, consectetur adipiscing elit. In et enim eget nisl luctus venenatis. Pellentesque sed erat sodales, tincidunt quam non, eleifend risus. Fusce aliquam dignissim pharetra. Integer id dui ac libero tincidunt consectetur. Sed laoreet nunc nec semper laoreet.

CASE 2b $x = 3$**:**   consectetur adipiscing elit. In et enim eget nisl luctus venenatis. Pellentesque sed erat sodales, tincidunt quam non, eleifend

$\square$

We may reference the case number 2 and cleveref case 2 as well as case 2a and cleveref case 2b.

To skip numbering and instead reference with the argument to \case use pfcases*. For instance, consider the following code.

```
\begin{proof}
\begin{pfcases*}
\case[\( x = y \)]\label{case*:first}  Lorem ipsum dolor sit amet, consectetur
\adipiscing elit. In et enim eget nisl luctus venenatis. Pellentesque sed erat
\sodales, tincidunt quam non, eleifend risus. Fusce aliquam dignissim pharetra.
\Integer id dui ac libero tincidunt consectetur. Sed laoreet nunc nec semper
\laoreet.

Vestibulum semper eget velit ut lobortis. In vel finibus est. Nullam tellus
dolor, pellentesque sed orci sed, ornare pretium diam. Nam vel tincidunt tellus.
Nulla at mi nisl.

\case[\( x = z \land z > q \land z < r \land x + z = r \)] \label{case*:second}
\Quisque consectetur, felis non congue dictum, mauris mi suscipit sem, vel
\laoreet justo ipsum in tellus. Suspendisse blandit malesuada velit faucibus
\pulvinar.

\begin{pfcases*}
```

```
\case[\( x=2 \)] \label{case*:second:sub1} Lorem ipsum dolor sit amet,
\consectetur adipiscing elit. In et enim eget nisl luctus venenatis.
\Pellentesque sed erat sodales, tincidunt quam non, eleifend risus. Fusce
\aliquam dignissim pharetra. Integer id dui ac libero tincidunt consectetur. Sed
\laoreet nunc nec semper laoreet.

\case[\( x = 3 \)] \label{case*:second:sub2}  consectetur adipiscing elit. In et
\enim eget nisl luctus venenatis. Pellentesque sed erat sodales, tincidunt quam
\non, eleifend
\end{pfcases*}
\end{pfcases*}
\end{proof}
```

```
Again we can ref case \ref{case*:second} and cleverref that \cref{case*:second}.  Als
```

Proof. CASE $x = y$**:**   Lorem ipsum dolor sit amet, consectetur adipiscing elit. In
et enim eget nisl luctus venenatis. Pellentesque sed erat sodales, tincidunt quam
non, eleifend risus. Fusce aliquam dignissim pharetra. Integer id dui ac libero
tincidunt consectetur. Sed laoreet nunc nec semper laoreet.

Vestibulum semper eget velit ut lobortis. In vel finibus est. Nullam tellus dolor,
pellentesque sed orci sed, ornare pretium diam. Nam vel tincidunt tellus. Nulla
at mi nisl.

CASE $x = z \wedge z > q \wedge z < r \wedge x + z = r$**:**   Quisque consectetur, felis non congue
dictum, mauris mi suscipit sem, vel laoreet justo ipsum in tellus. Suspendisse
blandit malesuada velit faucibus pulvinar.

CASE $x = 2$**:**   Lorem ipsum dolor sit amet, consectetur adipiscing elit. In
et enim eget nisl luctus venenatis. Pellentesque sed erat sodales, tincidunt
quam non, eleifend risus. Fusce aliquam dignissim pharetra. Integer id dui
ac libero tincidunt consectetur. Sed laoreet nunc nec semper laoreet.

CASE $x = 3$**:**   consectetur adipiscing elit. In et enim eget nisl luctus venenatis.
Pellentesque sed erat sodales, tincidunt quam non, eleifend

□

Again we can ref case $x = z \wedge z > q \wedge z < r \wedge x + z = r$ and cleverref that case $x = z \wedge z > q \wedge z < r \wedge x + z = r$. Also case $x = 2$ and cleveref subcase $x = 3$.

If the choice of cleveref labels aren't to your taste please let me know and I will consider changing them. Also, note that by passing the options `pfcasefont=textit` one can change the font used to typeset Case to italics (or whatever other font command you choose).

4.19. **Steps.** To enable the steps environment you *must* pass the option `steps` when loading the package. This enables use of the steps environment to typeset

presentation of procedures. The command \begin{steps} starts a list environ-
ment with an optional argument which is passed along unchanged as the optional
argument to the underlying enumitem based list. Inside the steps environment the
command \step[steptitle] prints a numbered step with steptitle displayed in
bold. An example of the use of the steps environment is provided below.

```
\begin{steps}
    \step If \( x=5 \) fails to hold end the stage without acting.  Otherwise the mod

    \step If there is \( k < m  \) with \(  s_{k} - 1 \approx_i s - 1  \) perform the
        \begin{steps}
            \step Choose \( a \nin \setcol{A}{1} \) large.
            \step Set \( c = c_k, b = b_k, \hat{s}_0 = s_{k} - 1, \hat{s}_1 = s - 1 \
            \step Enumerate \( b \) into \( \setcol{A}{2} \) and set \( R_{j,e}(s) =
        \end{steps}
    \step If there is no such \( k \) we instead enumerate \( c_m \) into  \( \setcol

\end{steps}
```

This yields the following output:

Step 1   If $x = 5$ fails to hold end the stage without acting. Otherwise the module
          acts by executing the subsequent steps.

Step 2   If there is $k < m$ with $s_k - 1 \approx_i s - 1$ perform the following steps and end
          the stage.

  Step 2a   Choose $a \notin A^{[1]}$ large.

  Step 2b   Set $c = c_k, b = b_k, \hat{s}_0 = s_k - 1, \hat{s}_1 = s - 1$.

  Step 2c   Enumerate b into $A^{[2]}$ and set $R_{j,e}(s) = 2$

Step 3   If there is no such k we instead enumerate c_m into $A^{[3]}$ and execute the
          procedure for expansionary stages.

5. Font Notes

The package is designed to work with both XITS fonts using unicode-math un-
der lualatex and xelatex as well as standard fonts under pdflatex. The package will
assume that XITS or a font with similar features is loaded if the unicode-math pack-
age is loaded. To properly represent symbols the package needs access to different
mathcal and mathscr fonts. By default, if unicode-math isn't loaded the package
will load mathrsfs. To disable this behavior pass the option norsfs. Furthermore,
it also needs access to a blackboard bold that works for lowercase symbols. This
isn't an issue using XITS but if unicode-math isn't loaded the package will load the
mathbbm package to define certain symbols. This behavior can be overridden by
passing the option nobbm and the package will do the best it can.

6. RELEASE NOTES

4.0 Stripped out non-working compatibility code and removed previously depreciated features. Removed mref helper. Removed the ball command. Removed undocumented hyphenation. Removed the recursive/computably terminology commands. Removed lots of unnecessary syntax commands and reworked the code to be simpler. Added option not to load rsfs. Added the commands `\succprec`, `\precsucc`, `\csuccprec`, `\cprecsucc` for stacked succ/prec symbols and added the option altcompat to use these instead of the standard mid and not mid compatibility symbols. Added the `\Searrow` symbol from MnSymbol and changed option from nodelim to nosymb. Changed commands for `\StrcR` and `\StrcD` to be `\strcR` and `\strcD` for consistency and added `\strcRa` and `\strcDa` as synonyms for `\Adegrees` and `\AREAdegrees`. Added `\Adegrees` and `\AREAdegrees` and the relations for arithmetic degrees. Merged the degree symbols into the degrees option and depreciated some alternate spellings. Added `\TreeMul`, `\TreeMod`, `\TreeExt`, `\wforces`, `\HYP`, depreciated the computable classes of sentences and simplified the standard ones. Fixed `\nconv` to be the symbol from MnSymbol if the nosymb option isn't passed, fixed `\concat` to not be so far above the line. Adjusted spacing for `\compat`, added abbreviations. Complete renaming and of the commands for ordinal notations and extra options to indicate limit, successor and unique sets of notations. Added `\Ofunc` as the effective limit for a notation. Fixed errors in the pfcases environment and eliminated some old compatibility code that wasn't being used by anyone. Added `\maps` helper for normal math stuff. Fixed BeamerRequirements. Added the `\bpfuncs` command and `\cbstrs`. Removed dependency on mathtools for compatibility with asl.cls

Adjusted a number of other command names/conventions and deleted redundant commands. Large scale simplification and deduplication. Made substantial changes to documentation to reflect this.

3.8.2 Removed option to put the set at the end of an `\REset` operation to avoid capturing later parenthesized arguments, e.g. `\REset{i}(A)` no longer works to avoid confusion with `\REset{i}(x)`. Fixed failed pdf doc update.

3.8.1 Fixed issues displaying the prime for jump operations.

3.8 Adjusted `\Tdeg` to be more beamer friendly and fixed it not to dumbly underline `0^n`. Fixed BeamerRequirements to work with differing values of `\abovedisplayskip`. Removed a few typos in docs for the requirements assistance. Added BeamerRequire and BeamerRequire* that put the requirements in a block and offer overlay specifications.

3.7 Fixed cases environment (both prettier and no problem with creating newline). Major re-factor to fix all options. Fixed bug with `\set` display. Added `\st` command inside the `\set` command to depreciate second argument. Added `\finSsets` (alt `\ssetsOfsize`), `\finsets` and optional

argument to `\powset`. Changed `\REAop` so the hat is on the starred version as intended. Added the steps (experimental) environment which requires an option to enable and fixed some problems with existing options. Reimplemented `\REset` to use xparse to avoid some errors in unusual contexts. Fixed the display of `\tpath`. Depreciated `\iREAop` and `\oneREAop`. Fixed `\Join`.

3.6 Fixed `\REA[n]` so that dash is shorter. Added BeamerRequirements to add a block for requirements in beamer and fixed the requirements environment for beamer. Fixed `\req*` error. Improved syntax for `\recfnl` so it can accept a parenthesis delimited argument as the oracle. Improved `\setcol` to allow it to be used in a nested fashion without typesetting bugs. Changed the `\code` and `\pair` commands to use only a single angle bracket.

3.5 Added `\Hop` and misc code cleanup.

3.4 Eliminated dependence on undertilde which is missing from texlive

3.3.1 Fixed typo causing error under pdflatex.

3.3 - Fixed/added tweak to overline so it looks correct. Also added real symbols so that `\subfunneq` and `\supfunneq` can be defined appropriately. Added `\floor` and `\ceil`. Note these aren't yet shown off in package doc. Fixed incorrect use of tiny in math mode.

3.2 - Removed `\reaop`, `\alphaREAop`, `\aREAop` in favor of using the single form `\REAop`. Removed `\functo`, `\map` and `funcomp`, `\hgtO` as useless synonyms and removed `\KleeneOBelow` and `KleeneOLess` as beyond what the package should define. Added package option compat31 to ensure package compatibility with version 3.1. An optional parentheses delimited argument specifying the base has been added to `\REAop`. Both `\REAop` and the pair `\REA/\CEA` have been updated to ignore order of optional arguments. The square brackets used to delimit the argument to the use command are now auto-sized. Added `\pmapsto`, `\pmapsfrom`, `\kleeneZero`, `\kleeneNum`, `\entersat`. Also Misc typesetting fixes.

3.1 02/26/2019 - Fixed `\wck` to be ω_1^{CK}, i.e., have capitalized roman CK.

3.01 02/17/2019 - Fixed `\RE` `\CE` `\Re` and `\Ce` for the various capitalized versions. Fixed weird bug with `\recfnl` no longer working based on let. Removed `\interior` and `\closure` as not really appropriate commands for the package and having bugs. Also fixed package to have correct version.

3.0 02/16/2019 - Added requirements environment for multiple requirements. Changed the `\req` and `\require` commands to take their optional argument after the first mandatory arguments as well as before. Added the commands `\module` and `\modof` and `\xor`. Improved the corner quotes. Added `\leftofeq`, `\rightof`, `\rightofeq`. Added `\RE`, `\CE`, `\Ce`, `\Re` and `\Tincompat`, `\Tincomp`, `\Tcompat`. Changed the way strings are

symbolized and coded. Fixed suffix commands to work with unicode-math. Also added `\require*` inside `\requirements`. Added `\nleftofeq`, `\nrightof`, `\nrightofeq`, `\nleftof`. The commands `\ancestor`, `\descendant` etc... `\reqof`, `\Astages` and `\Vstagesnow` require the option suppPriorityTrees be passed to the class to use and should be viewed as depreciated. Fixed the options system so different symbols can be correctly passed to the class. Changed the way `\recf` works to comply with the usual syntax.

2.4.3 11/29/2018 - Rendered compatible with beamer by removing enumitem requirement if beamer is loaded.

2.4.2 11/29/2018 - Fixed horrible bugs introduced in last version and fixed many symbols to work even in pdflatex mode. Also have everything compiling again.

2.4.1 2/14/2018 - Moved to using xparse to define the case macros and several other macros to allow nested brackets for optional arguments. Added the recf command and cleaned up some option processing. Also worked around the mathtools/unicode-math font bug described here

2.4 1/17/2018 - Added priority tree helpers. Should be more robust with respect to existing definitions of common commands.

2.3 12/31/2017 - Added proof cases helper. Also fixed the issue with `\ncequiv` in X&LATEX

2.2 11/14/2017 - Fixed `\Tdeg` so it works different on symbols and vars and added `\Tdegof` and `\Tvarof`. Added `\subfunneq` and `\supfunneq`.

2.1 10/05/2017 - Fixed way packages are required so rec-thy can be loaded in a flexible order. Also fixed one or two bugs.

2.0 09/26/2017 - Added support for introducing requirements, the subfunction relation and probably other undocumented features

1.3 06/20/2012 - Added abbreviations for computable infinitary formulas and made a few minor fixes.

1.2 01/01/2011 - Fixed awful option processing bug preventing most options from being recognized and added mrref option.

1.0 10/15/2010 - Initial public release