THE REC-THY PACKAGE

PETER M. GERDES (GERDES@INVARIANT.ORG)

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, c.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

Date: February 14, 2018: Version 2.4.

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. 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. A notable exception to this rule are the formula classes where square brackets are used to indicate an oracle to be placed in the superscript, e.g., `\pizn[X]{2}` yields $\Pi^{0,X}_2$, so as not to generate confusion with the alternative notion $\Pi^0_2(X)$. Also a lowercase first letter in a formula class indicates the lightface version while a capital first letter indicates the boldface version.

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

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

`\PriorityTree`	$\mathbb{T}$	Truepath.
`\tpath`	$\mathbf{f}$	Truepath.
`\tpath[s]`	$\mathbb{f}_s$	Approximate Truepath.
`\xi \leftof \eta`	$\xi <_L \eta$	The left of relation for priority tree arguments

`\Astages{\xi}`	$\mathcal{S}^{\xi}_{\text{act}}$	Set of stages at which ξ is active.
`\Vstages{\xi}`	$\mathcal{S}^{\xi}$	Stages where $\xi \prec \mathbb{f}_s$ even if links skip ξ.
`\reqof{\xi}`	$\mathbb{r}(\xi)$	Requirement implemented by a node ξ.
`\ball[X]{y}{s}`	$\boldsymbol{\alpha}^{X}(y, s)$	Location of ball y (headed to X) at s
`\ball{y}{s}`	$\boldsymbol{\alpha}(y, s)$	Location of ball y at s

3.2. **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{e}[Y]`	ϕ^{Y}_{e}	
`\recf[s]{e}[Y]`	$\phi^{Y}_{e,s}$	
`\recf{e}[Y](x)`	$\phi^{Y}_{e,s}(x)$	
`\recf[s]{e}[Y](x)`	$\phi^{Y}_{e,s}(x)$	
`\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}`	$\Phi_e(x)$	
`\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`	$\Phi_e(x)\downarrow_{-NoValue-}$	Convergence
`\recfnl{e}{}{x}\conv[s]`	$\Phi_e(x)\downarrow_{s}$	

`\recfnl{e}{}{x}\nconv`	$\Phi_e(x)\!\!\nwarrow\ \downarrow_{-NoValue-}$	Divergence
`\recfnl{e}{}{x}\nconv[s]`	$\Phi_e(x)\!\!\nwarrow\ \downarrow_s$	
`\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}`	$W_{e,s}^X$	
`\iREAop{e}(\eset)` `\reaop*{e}(\eset)`	$J_e(\varnothing)$	1-REA operator
`\alphaREAop{\alpha}(\eset)` `\reaop{\alpha}(\eset)`	$\mathcal{J}^\alpha(\varnothing)$	α-REA operator
`\alphaREAop[f]{\alpha}(\eset)` `\reaop[f]{\alpha}(\eset)`	$\mathcal{J}_f^\alpha(\varnothing)$	with particular witness to uniformity

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

`\Tdeg{d}`	$\mathbf{d}$	Turing degree
`\Tjump{X}` `\jump{X}`	X'	Turing jump
`\jjump{X}`	X''	
`\jumpn{X}{n}`	$X^{(n)}$	
`\Tzero`	$\mathbb{0}$	Computable degree
`\zeroj`	$\mathbb{0}'$	
`\zerojj`	$\mathbb{0}''$	
`\zerojjj`	$\mathbb{0}'''$	
`\zeron{n}`	$\mathbb{0}^{(n)}$	
`X \Tequiv Y` `X \Teq Y`	$X \equiv_{\mathbf{T}} Y$	Turing equivalence
`X \nTequiv Y` `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`	$X \not\leq_{\mathbf{T}} Y$	
`X \nTgeq Y`	$X \not\geq_{\mathbf{T}} Y$	
`\Tdeg{d} \Tdegjoin \Tdeg{d'}`	$\mathbf{d} \vee_{\mathbf{T}} \mathbf{d}'$	Join of degrees
`\Tdeg{d} \Tdegmeet \Tdeg{d'}` `\Tdeg{d} \Tmeet \Tdeg{d'}`	$\mathbf{d} \wedge_{\mathbf{T}} \mathbf{d}'$	Meet of degrees (when defined)
`X \Tplus Y` `X \Tjoin Y` `\TPlus_{i \in \omega} X_i` `\TJoin_{i \in \omega} X_i`	$X \oplus Y$ $\bigoplus_{i\in\omega} X_i$	Effective join of sets
`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`	$X \not\leq_{\mathrm{tt}} Y$	
`X \ttngeq Y`	$X \not\geq_{\mathrm{tt}} Y$	

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

We also introduce the following two enviornments for introducing requirements. The requirement enviornment 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

$$\mathscr{R}^{*}{}_{r,j}\colon\ \Phi_r(B) = W_j \implies \exists[k]\left(\Upsilon^{j}_{k}(C \oplus W_j) = B) \lor W_j \leq_{\mathbf{T}} \mathbb{0}\right)$$

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

```
\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

$$\mathscr{R}_i\colon \qquad \Phi_i(B) = W_i \implies \exists[k]\left(\Upsilon^{i}_{k}(C \oplus W_i) = B) \lor W_j \leq_{\mathbf{T}} \mathbb{0}\right)$$

3.5. **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

`x \meet y`
`\Meet_{i\in \omega} x_i`
	$x \wedge y$
	$\bigwedge_{i\in\omega} x_i$
		Meet operation

`x \join y`
`\Join_{i\in \omega} x_i`
	$x \vee y$
	$\bigotimes_{i\in\omega} x_i$
		Join operation

`\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\map{X}{Y}`
`\functo{f}{X}{Y}`
	$f : X \mapsto Y$
	$f : X \mapsto Y$
		Function specification

`f \compfunc g`
`f \funcomp g`
`f \compose g`
	$f \circ g$	Function composition

`\( \ensuretext{blah} \)`
`\ensuretext{blah}`
	blah	Types argument in text mode

3.6. **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}`	$\{-NoValue-\|(x,y)\}\ x > y$	Set notation
`\set{(x,y)}{}`	$\{-NoValue-\|(x,y)\}$	
`\set{(x,y)}`	$\{-NoValue-\|(x,y)\}$	
`\card{X}`	$\|X\|$	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
`\eset`	$\varnothing$	Emptyset abbreviation
`x \nin A`	$x \notin A$	not an element
`\setcmp{X}`	$\overline{X}$	Set compliment
`X \setdiff Y`	$X - Y$	Set difference
`X \symdiff Y`	$X \Delta Y$	Symmetric difference
`\interior X`	$\operatorname{int} X$	Interior
`\closure X`	$\circlearrowright X$	Closure

3.7. **Delimiters.** To disable these commands pass the option `nodelim`.

`\gcode{\phi}` `\godelnum{\phi}` `\cornerquote{\phi}`	$\ulcorner\phi\urcorner$	Godel Code/Corner Quotes
`\llangle x,y,z \rrangle`	$\langle\!\langle x,y,z \rangle\!\rangle$	Properly spaced double angle brackets

3.8. **Recursive vs. Computable.** To disable these commands pass the option `nonames`. To use recursive, r.e. and recursively enumerable everywhere pass the option `re`. To use computable, c.e. and computably enumerable everywhere pass the option `ce`. To force REA and CEA use the options `rea` and `cea`. If none of these options are passed the macros will expand as below. All macros in this section work in both text and math modes.

`\re`	r.e.
`\ce`	c.e.
`\REA`	REA
`\CEA`	CEA
`\recursive`	recursive
`\computable`	computable
`\recursivelyEnumerable`	recursively enumerable
`\computablyEnumerable`	computably enumerable
`\Recursive`	Recursive
`\Computable`	Computable
`\RecursivelyEnumerable`	Recursively enumerable
`\ComputablyEnumerable`	Computably enumerable

3.9. **Quantifiers & Connectives.** 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`	$\exists*$	
`\exists*[x < y]`	$\exists * [x < y]$	
`\exists*(x < y)`	$\exists * (x < y)$	
`\nexists[x < y]`	$\nexists[x < y]$	
`\nexists(x < y)`	$\nexists(x < y)$	
`\nexists*` `\nexistsinf`	$\nexists*$	
`\nexists*[x < y]`	$\nexists * [x < y]$	
`\nexists*(x < y)`	$\nexists * (x < y)$	
`\forall[x < y]`	$\forall[x < y]$	
`\forall(x < y)`	$\forall(x < y)$	
`\forall*` `\forallae`	$\forall*$	For almost all.
`\forall*[x < y]`	$\forall * [x < y]$	
`\forall*(x < y)`	$\forall * (x < y)$	
`\True`	$\top$	
`\False`	$\bot$	
`\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

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

`\bstrs`	$2^{<\omega}$	Finite binary strings
`\wstrs`	$\omega^{<\omega}$	Finite sequences of integers
`\cantor`	2^{ω}	Cantor space
`\baire`	ω^{ω}	Baire space
`\Baire`	$\mathcal{N}$	Alternate baire space

3.11. **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 conatination symbol pass `concatsym=macroname`, e.g., if you specify `\def\plus{+}` and then pass `concatsym=plus` the concatenation symbol would be changed to +.

`\str{1,0,1}` `\code{5,8,13}`	$\langle\langle 1,0,1\rangle\rangle$ $\langle\langle 5,8,13\rangle\rangle$	Strings/Codes for strings
`\EmptyStr` `\estr`	$\langle\langle\rangle\rangle$ $\langle\langle\rangle\rangle$	Empty string
`\decode{\sigma}{3}`	$(\sigma)_3$	Alternate notation for $\sigma(3)$
`\sigma\concat\tau` `\sigma\concat[0]`	$\sigma\,\hat{}\,\tau$ $\sigma\,\hat{}\,\langle\langle 0\rangle\rangle$	Concatenation
`\strpred{\sigma}`	σ^-	The immediate predecessor of σ
`\lh{\sigma}`	$\lvert\sigma\rvert$	Length of σ
`\sigma \incompat \tau` `\sigma \incomp \tau`	$\sigma \mid \tau$ $\sigma \mid \tau$	Incompatible strings
`\sigma \compat \tau`	$\sigma \nmid \tau$	Compatible strings
`\pair{x}{y}`	$\langle x, y\rangle$	Code for the pair (x, y)
`\setcol{X}{n}`	$X^{[n]}$	$\{-NoValue-\lvert y\}\ \langle n, y\rangle \in X$
`\setcol{X}{\leq n}`	$X^{[\leq n]}$	$\{-NoValue-\lvert\langle x, y\rangle\}\ \langle x, y\rangle \in X \wedge x \leq n$

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

`f \subfun g`	$f \prec g$	
`f \supfun g`	$f \succ g$	
`f \nsubfun g`	$f \nprec g$	~~Varities of the function extension relation~~
`f \nsupfun g`	$f \nsucc g$	
`f \subfuneq g`	$f \preceq g$	
`f \supfuneq g`	$f \succeq g$	
`f \nsubfuneq g`	$f \npreceq g$	
`f \nsupfuneq g`	$f \nsucceq g$	

3.13. **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}`	$T^{\langle\infty\rangle}$	$\{-NoValue-\mid\sigma\in T\}\ \exists(g)(g\in[T]\land\sigma\subset g)$
`\hgt{T}`	$\lVert T\rVert$	

3.14. **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 \subseteq* Y`
	$X\subset *Y$
	$X\subseteq *Y$
		All but finitely much of X is in Y

`X \supset* Y`
`X \supseteq* Y`
	$X\supset *Y$
	$X\supseteq *Y$
		All but finitely much of Y is in X

`X \eq Y`	$X=Y$	Macro for `=`

`X \eq* Y`
`X \eqae Y`
	$X =^* Y$	Equal mod finite

`X \infsubset Y`	$X\subset_\infty Y$	$X\subset Y\land\lvert Y\setminus X\rvert=\omega$
`X \infsubset* Y`	$X\subset^*_\infty Y$	$X\subset *Y\land\lvert Y\setminus X\rvert=\omega$
`X \infsupset Y`	$X\supset_\infty Y$	$Y\subset X\land\lvert X\setminus Y\rvert=\omega$
`X \infsupset* Y`	$X\supset^*_\infty Y$	$Y\subset *X\land\lvert X\setminus Y\rvert=\omega$
`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

3.15. **Ordinal Notations.** To disable these commands pass the option `noordinalnotations`.

`\wck`	ω_1^{ck}	First non-computable ordinal
`\ordzero`	0	Notation for ordinal 0
`\abs{\alpha}`	$\lvert\alpha\rvert$	Ordinal α denotes
`\kleeneO` `\ordNotations`	$\mathcal{O}$	Set of ordinal notations
`\kleeneO*` `\uniqOrdNotations` `\kleeneOuniq`	$\overline{\mathcal{O}}$	Unique set of ordinal notations
`\kleeneO(X)`	$\mathcal{O}^X$	Relativized ordinal notations
`\kleeneO[\alpha]`	$\mathcal{O}_{\lvert\alpha\rvert}$	Ordinal notations for ordinals $< \lvert\alpha\rvert$
`\kleeneO*(X)[\alpha]`	$\overline{\mathcal{O}}^X_{\lvert\alpha\rvert}$	
`\alpha \kleeneless \beta`	$\alpha <_{\mathcal{O}} \beta$	Ordering on notations
`\alpha \kleenel    \beta`	$\alpha <_{\mathcal{O}} \beta$	
`\alpha \kleeneleq  \beta`	$\alpha \leq_{\mathcal{O}} \beta$	
`\alpha \kleenegtr  \beta`	$\alpha >_{\mathcal{O}} \beta$	
`\alpha \kleenegeq  \beta`	$\alpha \geq_{\mathcal{O}} \beta$	
`\alpha \kleenePlus \beta`	$\alpha +_{\mathcal{O}} \beta$	Effective addition of notations
`\alpha \kleeneMul  \beta`	$\alpha \cdot_{\mathcal{O}} \beta$	Effective multiplication of notations
`\kleenelim{\lambda}{n}`	$\lambda_{[n]}$	The n-th element in effective limit defining notation λ
`\kleenepred{\alpha}`	α^-	Predecessor of α if defined
`\kleenehgt{R}` `\hgtO{R}`	$\lVert R\rVert_{\mathcal{O}}$	Heigh of computable relation R

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

`\sigma \forces \phi` `\sigma \frc \phi`	$\sigma \Vdash \phi$	σ forces ϕ
`\sigma \forces(X) \phi`	$\sigma \Vdash^X_T \phi$	ϕ is formula relative to X
`\sigma \forces[T] \phi`	$\sigma \Vdash^X_T \phi$	Local forcing on T
`\sigma \forces* \phi`	$\sigma \Vdash^* \phi$	Strong forcing

3.17. **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. Capitalizing the first letter gives the boldface version in all cases (except the computable infinitary formulas as this doesn't make sense). Not all formulas/abbreviations are demonstrated below given the huge number but the enough are included to make it clear what command is required to generate the desired formula class, e.g., substituting pi for delta does what you think it does.

To change the syntax for the computable infinitary formulas you can pass the options `cdeltasym=macroname`, `csigmasym=macroname` and `cpisym=macroname` where macroname is the name (without the leading
) of the macro giving the desired symbol to use for the relevant class.

`\Cdeltan[X]{\alpha}`	${}^{\mathcal{C}}\Delta^{X}_{2}$	The computable δ^{X}_{α} formulas
`\deltan{2}`	Δ_2	
`\deltan[X]{2}`	Δ^{X}_{2}	
`\deltaZeroN[X]{2}` `\deltazn[X]{2}`	$\Delta^{0,X}_{2}$	
`\deltaZeroOne[X]` `\deltazi[X]`	$\Delta^{0,X}_{1}$	
`\sigmaZeroTwo[X]` `\sigmazii[X]`	$\Delta^{0,X}_{2}$	
`\deltaZeroThree[X]` `\deltaziii[X]`	$\Delta^{0,X}_{3}$	
`\deltaOneN[X]{2}` `\deltaIn[X]{2}`	$\Delta^{1,X}_{2}$	
`\deltaOneOne[X]` `\deltaIi[X]`	$\Delta^{1,X}_{1}$	
`\deltaOneTwo[X]` `\deltaIii[X]`	$\Delta^{1,X}_{2}$	
`\deltaOneThree[X]` `\deltaIiii[X]`	$\Delta^{1,X}_{3}$	
`\pizi`	Π^{0}_{1}	
`\pizn[X]{n}`	$\Pi^{0,X}_{n}$	
`\Deltan{2}`	$\boldsymbol{\Delta}_2$	
`\DeltaOneN[X]{n}`	$\boldsymbol{\Delta}^{1,X}_{n}$	
`\logic{\omega_1}{\omega}`	$\mathcal{L}_{\omega_1,\omega}$	Indicates the kind of infinitary logic

3.18. **Proof Cases.**

Proof. The pfcases enviornment provides a numbered, referenceable division of a proof segment into cases.

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

CASE 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

We can now 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*

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 $x = z \land z > q \land z < r \land x + z = r$ and cleverref that case $x = z \land z > q \land z < r \land x + z = r$. 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).

The above is accomplished with the following code.

```
\begin{proof}
\begin{pfcases}
\case[\( x = y \)]\label{case:first} ...
\case[\( x = z \land z > q \land z < r \land x + z = r \)] \label{case:second} ..
\begin{pfcases}
\case[\( x=2 \)] \label{case:second:sub1} ..
\case[\( x = 3 \)] \label{case:second:sub2}..
\end{pfcases}
\end{pfcases}
.. \ref{case:second}.. \cref{case:second} ..  \ref{case:second:sub1} .. \cref{case:se
\begin{pfcases*}
\case[\( x = y \)]\label{case*:first} ..
\case[\( x = z \land z > q \land z < r \land x + z = r \)] \label{case*:second} ..
\begin{pfcases*}
\case[\( x=2 \)] \label{case*:second:sub1} ..
\case[\( x = 3 \)] \label{case*:second:sub2}
\end{pfcases*}
.. \ref{case*:second} .. \cref{case*:second} .. \ref{case*:second:sub1} .. \cref{case
\end{proof}
```

3.19. **MRref.** Finally to enable the mrref helper macros pass the option `mrref`.

These macros normalize the formating of mathscinet references for supported bibliography styles and ensure the MR numbers link to the mathscinet page of the article. Unless you have a good reason (like journal formatting guidelines) there is no reason not to always pass this option. Note this option requires the hyperref package.

4. Release Notes

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 XƎLaTeX

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