

The rec-thy Package

Peter M. Gerdes (gerdes@invariant.org)

2011-01-01: Version 2.1

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 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 $\Pi_2^{0,X}$, so as not to generate confusion with the alternative notion $\Pi_2^0(X)$. 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 **u**

3.1 Computations

To disable these commands pass the option `nocomputations`.

<code>\murec{x}{f(x)>1}</code>	$\mu x (f(x) > 1)$	Least x satisfying condition.
<code>\recfnl{e}{Y}{x}</code>	$\Phi_e(Y; x)$	Computable functions/functionals
<code>\recfnl[s]{e}{Y}{x}</code>	$\Phi_{e,s}(Y; x)$	
<code>\recfnl{e}{Y}{}</code>	$\Phi_e(Y)$	
<code>\recfnl{e}{}{x}</code>	$\Phi_e(x)$	
<code>\recfnl{e}{}{}</code>	Φ_e	
<code>\recfnl{e}{}{} \simeq \recfnl{i}{}{}</code>	$\Phi_e \simeq \Phi_i$	Equivalent computations
<code>\recfnl{e}{}{} \not\simeq \recfnl{i}{}{}</code>	$\Phi_e \not\simeq \Phi_i$	Inequivalent computations
<code>\recfnl{e}{}{x} \downarrow</code> <code>\recfnl{e}{}{x} \downarrow [s]</code>	$\Phi_e(x) \downarrow$ $\Phi_e(x) \downarrow_s$	Convergence
<code>\recfnl{e}{}{x} \not\downarrow</code> <code>\recfnl{e}{}{x} \not\downarrow [s]</code>	$\Phi_e(x) \not\downarrow$ $\Phi_e(x) \not\downarrow_s$	Divergence
<code>\use{\recfnl{e}{Y}{x}}</code>	$u[\Phi_e(Y; x)]$	Use of a computation.
<code>\REset{e}</code>	W_e	c.e. sets
<code>\REset[s]{e}</code>	$W_{e,s}$	
<code>\REset(X){e}</code>	W_e^X	
<code>\REset[s](X){e}</code>	$W_{e,s}^X$	
<code>\iREAop{e}(\set)</code> <code>\reaop*{e}(\set)</code>	$J_e(\emptyset)$	1-REA operator
<code>\alphaREAop{\alpha}(\set)</code> <code>\reaop{\alpha}(\set)</code>	$\mathcal{J}^\alpha(\emptyset)$	α -REA operator
<code>\alphaREAop[f]{\alpha}(\set)</code> <code>\reaop[f]{\alpha}(\set)</code>	$\mathcal{J}_f^\alpha(\emptyset)$	with particular witness to uniformity

3.2 Degrees

To disable these commands pass the option `nodegrees`.

$\backslash\text{Tdeg}\{d\}$	\mathbf{d}	Turing degree
$\backslash\text{Tjump}\{X\}$ $\backslash\text{jump}\{X\}$	X'	Turing jump
$\backslash\text{jjump}\{X\}$	X''	
$\backslash\text{jumpn}\{X\}\{n\}$	$X^{(n)}$	
$\backslash\text{Tzero}$	$\not\prec$	Computable degree
$\backslash\text{zeroj}$	$\not\prec'$	
$\backslash\text{zerojj}$	$\not\prec''$	
$\backslash\text{zerojjj}$	$\not\prec'''$	
$\backslash\text{zeron}\{n\}$	$\not\prec^{(n)}$	
$X \backslash\text{Tequiv} Y$ $X \backslash\text{Teq} Y$	$X \equiv_{\mathbf{T}} Y$	Turing equivalence
$X \backslash\text{nTequiv} Y$ $X \backslash\text{nTeq} Y$	$X \not\equiv_{\mathbf{T}} Y$	Turing inequivalence
$X \backslash\text{Tlneq} Y$	$X \leq_{\mathbf{T}} Y$	
$X \backslash\text{Tleq} Y$	$X \leq_{\mathbf{T}} Y$	
$X \backslash\text{Tgneq} Y$	$X \geq_{\mathbf{T}} Y$	
$X \backslash\text{Tgeq} Y$	$X \geq_{\mathbf{T}} Y$	
$X \backslash\text{Tgtr} Y$	$X >_{\mathbf{T}} Y$	
$X \backslash\text{Tless} Y$	$X <_{\mathbf{T}} Y$	
$X \backslash\text{nTleq} Y$	$X \not\leq_{\mathbf{T}} Y$	
$X \backslash\text{nTgeq} Y$	$X \not\geq_{\mathbf{T}} Y$	
$\backslash\text{Tdeg}\{d\} \backslash\text{Tdegjoin} \backslash\text{Tdeg}\{d'\}$	$\mathbf{d} \vee_{\mathbf{T}} \mathbf{d}'$	Join of degrees
$\backslash\text{Tdeg}\{d\} \backslash\text{Tdegmeet} \backslash\text{Tdeg}\{d'\}$ $\backslash\text{Tdeg}\{d\} \backslash\text{Tmeet} \backslash\text{Tdeg}\{d'\}$	$\mathbf{d} \wedge_{\mathbf{T}} \mathbf{d}'$	Meet of degrees (when defined)
$X \backslash\text{Tplus} Y$ $X \backslash\text{Tjoin} Y$ $\backslash\text{TPlus}_{\{i \in \omega\}} X_i$ $\backslash\text{TJoin}_{\{i \in \omega\}} X_i$	$X \oplus Y$ $\bigoplus_{i \in \omega} X_i$	Effective join of sets

$X \not\leq_{tt} Y$	$X \leq_{tt} Y$	Truth table reducibilities
$X \leq_{tt} Y$	$X \leq_{tt} Y$	
$X \not\geq_{tt} Y$	$X \geq_{tt} Y$	
$X \geq_{tt} Y$	$X \geq_{tt} Y$	
$X >_{tt} Y$	$X >_{tt} Y$	
$X <_{tt} Y$	$X <_{tt} Y$	
$X \not\leq_{tt} Y$	$X \not\leq_{tt} Y$	
$X \not\geq_{tt} Y$	$X \not\geq_{tt} Y$	

3.3 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}`

<code>\req{R}{e}</code>	\mathcal{R}_e	
<code>\req[\nu]{R}{e}</code>	\mathcal{R}_e^ν	Requirement
<code>\req*{R}{e}</code>	\mathcal{R}_e	
<code>\req*[\nu]{R}{e}</code>	\mathcal{R}_e^ν	Requirement without hyperlinks

We also introduce the following two 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_k^j(C \oplus W_j) = B \right) \lor \left(
  C \oplus W_j = B \right)
\end{requirement}

```

Giving output

$$\mathcal{R}_{r,j}^*: \Phi_r(B) = W_j \implies [\exists k] \left(\Upsilon_k^j(C \oplus W_j) = B \right) \vee W_j \leq_{\mathbf{T}} \not\leq$$

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 are just 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

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

3.4 General Math Commands

To disable these commands pass the option `nomath`.

<code>\eqdef</code>	$\stackrel{\text{def}}{=}$	Definitional equals
<code>\iffdef</code>	$\stackrel{\text{def}}{\iff}$	Definitional equivalence
<code>\aut</code>	Aut	Automorphisms of some structure
<code>\Ord</code>	Ord	Set of ordinals
<code>x \meet y</code>	$x \wedge y$	Meet operation
<code>\Meet_{i \in \omega} x_i</code>	$\bigwedge_{i \in \omega} x_i$	
<code>x \join y</code>	$x \vee y$	Join operation
<code>\Join_{i \in \omega} x_i</code>	$\bigvee_{i \in \omega} x_i$	
<code>\abs{x}</code>	$ x $	Absolute value
<code>\dom</code>	dom	Domain
<code>\rng</code>	rng	Range
<code>f \restr{X}</code>	$f _X$	Restriction
<code>\ordpair{x}{y}</code>	(x, y)	Ordered Pair
<code>f \map{X}{Y}</code> <code>\functo{f}{X}{Y}</code>	$f : X \mapsto Y$ $f : X \mapsto Y$	Function specification
<code>f \compfunc g</code> <code>f \funcomp g</code> <code>f \compose g</code>	$f \circ g$	Function composition
<code>\(\ensuretext{blah} \)</code> <code>\ensuretext{blah}</code>	blah	Types argument in text mode

3.5 Set Notation

To disable these commands pass the option `nosets`.

<code>\set{(x,y)}{x > y}</code>	$\{(x, y) x > y\}$	Set notation
<code>\set{(x,y)}{\}</code>	$\{(x, y)\}$	
<code>\set{(x,y)}</code>	$\{(x, y)\}$	
<code>\card{X}</code>	$ X $	Cardinality
<code>X \union Y</code>	$X \cup Y$	Union
<code>\Union_{i \in \omega} X_i</code>	$\bigcup_{i \in \omega} X_i$	
<code>X \isect Y</code>	$X \cap Y$	Intersection
<code>\Isect_{i \in \omega} X_i</code>	$\bigcap_{i \in \omega} X_i$	
<code>X \cross Y</code>	$X \times Y$	Cartesian product (Cross Product)
<code>\Cross_{i \in \omega} X_i</code>	$\prod_{i \in \omega} X_i$	
<code>\powset{\omega}</code>	$\mathcal{P}(\omega)$	Powerset
<code>\eset</code>	\emptyset	Emptyset abbreviation
<code>x \nin A</code>	$x \notin A$	not an element
<code>\setcmp{X}</code>	\overline{X}	Set compliment
	$\sim X$	With option <code>minussetcmp</code>
<code>X \setminusminus Y</code>	$X - Y$	Set difference
<code>X \syndiff Y</code>	$X \Delta Y$	Symmetric difference
<code>\interior X</code>	$\text{int } X$	Interior
<code>\closure X</code>	$\text{cl } X$	Closure

3.6 Delimiters

To disable these commands pass the option `nodelim`.

<code>\gcode{\phi}</code> <code>\godelnum{\phi}</code> <code>\cornerquote{\phi}</code>	$\lceil \phi \rceil$	Godel Code/Corner Quotes
<code>\llangle x,y,z \rrangle</code>	$\langle\langle x,y,z \rangle\rangle$	Properly spaced double angle brackets

3.7 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.

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

3.8 Quantifiers & Connectives

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

<code>\exists[x < y]</code>	$[\exists x < y]$	
<code>\exists(x < y)</code>	$(\exists x < y)$	
<code>\exists*</code>	\exists^∞	
<code>\existssinf</code>		
<code>\exists*[x < y]</code>	$[\exists^\infty x < y]$	
<code>\exists*(x < y)</code>	$(\exists^\infty x < y)$	
<code>\nexists[x < y]</code>	$[\nexists x < y]$	
<code>\nexists(x < y)</code>	$(\nexists x < y)$	
<code>\nexists*</code>	\nexists^∞	
<code>\nexistssinf</code>		
<code>\nexists*[x < y]</code>	$[\nexists^\infty x < y]$	
<code>\nexists*(x < y)</code>	$(\nexists^\infty x < y)$	
<code>\forall[x < y]</code>	$[\forall x < y]$	
<code>\forall(x < y)</code>	$(\forall x < y)$	
<code>\forall*</code>	\forall^*	
<code>\forallallae</code>		For almost all.
<code>\forall*[x < y]</code>	$[\forall^* x < y]$	
<code>\forall*(x < y)</code>	$(\forall^* x < y)$	
<code>\True</code>	\top	
<code>\False</code>	\perp	
<code>\Land \phi_i</code>	$\bigwedge \phi_i$	Operator form of and
<code>\Lor \phi_i</code>	$\bigvee \phi_i$	Operator form of or
<code>\LLand \phi_i</code>	$\bigwedge \phi_i$	Infinitary conjunction
<code>\LLor \phi_i</code>	$\bigvee \phi_i$	Infinitary disjunction

3.9 Spaces

To disable these commands pass the option `nospaces`.

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

3.10 Strings

To disable these commands pass the option `nostrings`.

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

3.11 Subfunctions

To disable these commands pass the option `nosubfuns`.

<code>f \subfun g</code>	$f \prec g$	Varieties of the function extension relation
<code>f \supfun g</code>	$f \succ g$	
<code>f \nsubfun g</code>	$f \not\prec g$	
<code>f \nsupfun g</code>	$f \not\succ g$	
<code>f \subfuneq g</code>	$f \preceq g$	
<code>f \supfuneq g</code>	$f \succeq g$	
<code>f \nsubfuneq g</code>	$f \not\preceq g$	
<code>f \nsupfuneq g</code>	$f \not\succeq g$	

3.12 Trees

To disable these commands pass the option `notrees`.

<code>\CBderiv{T}</code>	$T^{(1)}$	Cantor-Bendixson Derivative
<code>\CBderiv[\alpha]{T}</code>	$T^{(\alpha)}$	
<code>\pruneTree{T}</code>	$T^{(\infty)}$	$\{\sigma \in T \mid (\exists g)(g \in [T] \wedge \sigma \subset g)\}$
<code>\hgt{T}</code>	$\ T\ $	

3.13 Set Relations

To disable these commands pass the option `nosetrels`.

Note that many of these commands are extensions of existing commands.

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

3.14 Ordinal Notations

To disable these commands pass the option `noordinalnotations`.

<code>\wck</code>	ω_1^{ck}	First non-computable ordinal
<code>\ordzero</code>	0	Notation for ordinal 0
<code>\abs{\alpha}</code>	$ \alpha $	Ordinal α denotes
<code>\kleene0</code> <code>\ordNotations</code>	\mathcal{O}	Set of ordinal notations
<code>\kleene0*</code> <code>\uniqOrdNotations</code> <code>\kleene0uniq</code>	$\overline{\mathcal{O}}$	Unique set of ordinal notations
<code>\kleene0(X)</code>	\mathcal{O}^X	Relativized ordinal notations
<code>\kleene0[\alpha]</code>	$\mathcal{O}_{ \alpha }$	Ordinal notations for ordinals $< \alpha $
<code>\kleene0*(X)[\alpha]</code>	$\overline{\mathcal{O}}_{ \alpha }^X$	
<code>\alpha \kleeneless \beta</code>	$\alpha <_{\mathcal{O}} \beta$	Ordering on notations
<code>\alpha \kleene1 \beta</code>	$\alpha <_{\mathcal{O}} \beta$	
<code>\alpha \kleeneleq \beta</code>	$\alpha \leq_{\mathcal{O}} \beta$	
<code>\alpha \kleenegtr \beta</code>	$\alpha >_{\mathcal{O}} \beta$	
<code>\alpha \kleenegeq \beta</code>	$\alpha \geq_{\mathcal{O}} \beta$	
<code>\alpha \kleenePlus \beta</code>	$\alpha +_{\mathcal{O}} \beta$	Effective addition of notations
<code>\alpha \kleeneMul \beta</code>	$\alpha \cdot_{\mathcal{O}} \beta$	Effective multiplication of notations
<code>\kleene1im{\lambda}{n}</code>	$\lambda_{[n]}$	The n -th element in effective limit defining notation λ
<code>\kleenepred{\alpha}</code>	α^-	Predecessor of α if defined
<code>\kleenehgt{R}</code> <code>\hgt0{R}</code>	$\ R\ _{\mathcal{O}}$	Heigh of computable relation R

3.15 Forcing

To disable these commands pass the option `noforcing`.

<code>\sigma \forces \phi</code>	$\sigma \Vdash \phi$	σ forces ϕ
<code>\sigma \frc \phi</code>		
<code>\sigma \forces(X) \phi</code>	$\sigma \Vdash_T^X \phi$	ϕ is formula relative to X
<code>\sigma \forces[T] \phi</code>	$\sigma \Vdash_T^X \phi$	Local forcing on T
<code>\sigma \forces* \phi</code>	$\sigma \Vdash^* \phi$	Strong forcing

3.16 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`, `csigmasym` and `cpisym` set equal to the command to produce your desired symbol. This is UNTESTED and quite likely doesn't work yet. If you desire this feature and it doesn't work send me a bug report.

<code>\Cdelta[X]{\alpha}</code>	${}^c\Delta_2^X$	The computable δ_α^X formulas
<code>\deltan{2}</code>	Δ_2	
<code>\deltan[X]{2}</code>	Δ_2^X	
<code>\deltaZeroN[X]{2}</code> <code>\deltazn[X]{2}</code>	$\Delta_2^{0,X}$	
<code>\deltaZeroOne[X]</code> <code>\deltazi[X]</code>	$\Delta_1^{0,X}$	
<code>\sigmaZeroTwo[X]</code> <code>\sigmazii[X]</code>	$\Delta_2^{0,X}$	
<code>\deltaZeroThree[X]</code> <code>\deltaziii[X]</code>	$\Delta_3^{0,X}$	
<code>\deltaOneN[X]{2}</code> <code>\deltaIn[X]{2}</code>	$\Delta_2^{1,X}$	

<code>\deltaOneOne[X]</code> <code>\deltaIi[X]</code>	$\Delta_1^{1,X}$	
<code>\deltaOneTwo[X]</code> <code>\deltaIii[X]</code>	$\Delta_2^{1,X}$	
<code>\deltaOneThree[X]</code> <code>\deltaIiii[X]</code>	$\Delta_3^{1,X}$	
<code>\pizi</code>	Π_1^0	
<code>\pizn[X]{n}</code>	$\Pi_n^{0,X}$	
<code>\Deltan{2}</code>	Δ_2	
<code>\DeltaOneN[X]{n}</code>	$\Delta_n^{1,X}$	
<code>\logic{\omega_1}{\omega}</code>	$\mathcal{L}_{\omega_1,\omega}$	Indicates the kind of infinitary logic

3.17 MRref

Finally to enable the mrref helper macros pass the option `mrref`.

These macros normalize the formatting 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

- 2.1 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