\textbf{unicode-math-input} — Allow entering Unicode symbols in math formulas*

user202729

Released 2023/05/12

\abstract
Allow entering Unicode symbols in math formulas.

1 Introduction

This package allows entering Unicode symbols in math formulas.

1.1 Existing packages

There are several existing packages, but other than \texttt{unicode-math} (which also changes the output encoding) they do not cover a lot of characters and/or does not handle several issues well.

We compare the situation with several existing packages:

- \texttt{unixode}:
  - defines \texttt{'} to be \texttt{\textbackslash prime} which is big and not usable, it should be \texttt{^\textbackslash prime} similar to \texttt{'}s definition.
  - defines \texttt{–} (en dash) to be nothing, which breaks the character even in text mode.
  - does not define \texttt{×} or \texttt{±} (they’re already valid in text mode in \LaTeX{}, but will be silently omitted in math mode)
  - does not handle consecutive superscript/subscript characters.
  - you need to manually patch the source code a bit in order to make it work with PDF\LaTeX{}. And even after that it will raise lots of warnings about redefining Unicode characters.

- \texttt{utf8x}:
  - incompatible with lots of packages.
  - does not define \texttt{\bigoplus} \texttt{(\bigoplus)}
  - also does not handle consecutive superscript/subscript characters.

See also \url{https://tex.stackexchange.com/a/628285}.

\*This file describes version v0.0.0, last revised 2023/05/12.
1.2 Features

\LaTeX{}'s implementation of input encoding and font encoding is very complicated, necessitated by the fact that non-Unicode \TeX{} engines handle each UTF-8 character as multiple tokens and enc\TeX{} extension is not enabled in \LaTeX{}.

There's a few other issues that we don't really need to deal with, because they are in the next layer:

- **What is the use of the command \texttt{\textbackslash iec}?**

  - [https://tex.stackexchange.com/a/239575/250119](https://tex.stackexchange.com/a/239575/250119)

  We don’t need to deal with \texttt{\textbackslash iec} as since \TeX{} Live 2019, the mechanism is no longer used and the Unicode character itself is written to auxiliary files.

  We need to get the following things correct:

  - \texttt{\textbackslash left}\{

    In \Llua{}\LaTeX{} in order to implement this we need to hard code the \texttt{\textbackslash unicode} of the character, so if \texttt{\textbackslash angle} is redefined, the change will not follow.

    An alternative is to overwrite the definition of \texttt{\textbackslash left} built-in, but this is not used.

  - \texttt{\textbackslash big}( in \texttt{amsmath} package or outside)

    In PDF\LaTeX{} there's an issue of argument-grabbing (\texttt{\big} etc. is a macro so they will only grab the first octet of the \texttt{\langle} character), so the macro must be patched.

    Furthermore, the patching is done \texttt{\textbackslash AtBeginDocument} in case \texttt{amsmath} etc. is loaded after this package.

    We handle \texttt{\big} \texttt{\Big} \texttt{\bigg} \texttt{\Bigg} and the \texttt{\bigl}, \texttt{\bigr} variants etc.

    Pass the option \texttt{ignore-patch-delimiter-commands} to disable the behavior in case of package clash.

  - in \texttt{unicode-math}, \texttt{a}\texttt{'} renders as \texttt{a}\texttt{\textasciicircum}\texttt{(backprime)} i.e. \texttt{a}'. We will not modify the default behavior i.e. \texttt{a'} in this package.

  - \texttt{\textbackslash section}\texttt{(\$1 \times 2\$)} (for writing to auxiliary file in table of contents) – as mentioned above, since \TeX{} Live 2019 this is correct by default.

  - Some characters such as \texttt{\texttimes} or \texttt{\textfrac{1}{2}} in PDF\LaTeX{} are already usable outside math mode, we try to not break the compatibility.

  - The symbol should work correctly when appear at the start of an alignment entry, e.g., the start of an \texttt{align*} cell.

  - \texttt{\$2^{3+4}\$} (consecutive Unicode characters for superscript/subscript, refer to [https://tex.stackexchange.com/q/344160/250119](https://tex.stackexchange.com/q/344160/250119).) Also need to handle \texttt{'} similarly.

  - This packages does modify the default definition of \texttt{'} to allow \texttt{G'^\textbackslash group 123\textbackslash group} to work however. Pass the option \texttt{ignore-patch-prime} to disable the behavior in case of package clash.

  - The original implementation of \texttt{'} is somewhat interesting that it allows sequences such as \texttt{G'^\textbackslash group} to work, we will not emulate it here.

\[1\] Refer to [https://tex.stackexchange.com/a/266282/250119](https://tex.stackexchange.com/a/266282/250119) for a way to force-enable enc\TeX{} extension in \LaTeX{} if you’re interested.
• Also need to handle Unicode prime symbols ′, ″ etc.

• To minimize errors, we make ⨆ default to \nequiv, but fallback to \not\equiv if the former is not available.

We should also take care of aliases – for example, ⨆ should check \nle and \nleq before fallback to \not\nle or \not\nleq.

Note that by default (or with amsmath or amsymb), \not does not smartly check the following symbol, however with some packages such as unicode-math, txfonts the \not does do that – in particular, it checks for the presence of control sequences named \notXXX and \nXXX where XXX stands for the original control sequence/character.

It would be beneficial for amsymb to make \not smart, as for example \not\exists looks worse than \nexists, however the package does not touch \not.

• Similarly, ″ default to ^{\dprime} if available, else fallback to ^{\prime\prime}.

• Whenever possible, we do not make the symbols have active catcode, only change the mathcode, that way usage of the symbols in places such as fancyverb environment is minimally affected. (see test files for an example)

• We try to make minimum assumptions about the internal implementation details of \LaTeX packages; nevertheless this is not always possible.

• Combining modifiers (such as U+00305 COMBINING OVERLINE in ā, which corresponds to \overbar) are difficult to support (although with whole-file scanning + rescan-sync or Lua\TeX’s process_input_buffer callback it’s not impossible; an alternative is to use Lua\TeX callback to modify the math list after it’s constructed, see https://github.com/wspr/unicode-math/issues/555#issuecomment-1045207378 for an example), plus unicode-math does not support them anyway, so they will not be supported. They’re difficult to support because normally the modifier appear after the character that it modifies but \LaTeX requires the command (e.g. \overbar) to appear before the character that it modifies.

As a special case, the 4 commands \enclosecircle \enclosesquare \enclosediamond and \enclosetriangle are supported (simply because the \LaTeX command can appear after the character it modifies)

• The fraction slash U+2044 FRACTION SLASH, as in 1⁄2 rendering 1⁄2, is also not implemented because of similar difficulty as above.

• Symbols such as √ or ⁜ will be equivalent to \sqrt command (taking an argument to draw a square root) instead of \surd (the symbol itself), unlike unicode-math. While sequences such as √(67) may feasibly be supported without breaking too many things, implementation is difficult and we don’t see much use for it.

• Similarly, one might expect that √U+23DF BOTTOM CURLY BRACKET get mapped to \underbrace, but the behavior of such command would be a bit unexpected (you need to write _{123}_(456) to get 123, so this will not be the default.
• the Unicode character is mapped indirectly to the control sequence, so that when
the user/some package redefines a control sequence such as \pi, the corresponding
Unicode character (π) will also change. This will incur a small loss in efficiency
however.

(modulo the issue with \Udelcode mentioned above)

There are some issues however:

• U+1D7D8 MATHEMATICAL DOUBLE-STRUCK DIGIT ZERO gets translated to \mathbb{0},
but this is incorrect by default unless the blackboard bold font happens to have
such a character.

(nevertheless, it’s difficult to change math font in the middle of the document
anyway. Refer to https://tex.stackexchange.com/q/30049/250119.)

• In the unicode-math source code there’s this remark:

    The catcode setting is to work around (strange?) behaviour in LuaT\TeX
    in which catcode 11 characters don’t have italic correction for maths. We
don’t adjust ascii chars, however, because certain punctuation should not
have their catcodes changed.

    This feature is currently unimplemented.

• At the moment, following a Unicode superscript character, double superscript will
not be defined – that is, G^{234} will just display as G^{(234)} – while this is fixable,
we don’t see much point in detecting this error.

2 Usage

Simply include the package:

\usepackage{unicode-math-input}

Because by default the unicode-math package will already allow entering Unicode
symbols in math formulas, this package will raise an error if the other package is already
loaded.
3 Advanced commands and options

\umiMathbf {...}
\umiMathit {...}

These functions are not to be used directly. But you can redefine them to customized behavior of bold/italic/etc. Unicode characters.

For example you can \renewcommand\umiMathbf[1]{\mathbf{#1}} which is the default behavior.

More usefully, you may want to \renewcommand\umiMathbf{\bm} to make entered characters such as \textalpha appear bold italic in the output, remember to load package bm if you want to do so (which is unicode-math behavior with \bold-style=ISO package option).

\umiFrac \umiDefineMathChar {α} {\alpha}

Not to be used directly, but you can redefine it such as \let\umiFrac\tfrac (or more clearly, \renewcommand\umiFrac[2]{\tfrac{#1}{#2}}) to customize the appearance of Unicode characters like ½.

If you want to customize the appearance of individual symbols, consider using \umiDefineMathChar.

\umiDefineMathDelimiter
\umiRefreshDelimiterList

You should normally not need this command.

As mentioned before, in Lua\LaTeX once a command is redefined, the Unicode character does not automatically update.

This command will check all the normal delimiter Unicode characters. In PDF\LaTeX this command does nothing.

Another way is to use \umiDefineMathDelimiter to manually refresh individual Unicode characters, this is also useful if you define an Unicode character that is not “normally” a delimiter.
Package option. \texttt{\umiRefreshDelimiterList} will be run \texttt{\AtBeginDocument}. Pass this to disable it running.

Only needed if there’s some package clash or if there’s spurious warning on “not determined to be a delimiter” etc.

\begin{verbatim}
\sheetTable
\end{verbatim}

\texttt{\umiPatchCmdUnicodeArg \sqrt}
\texttt{\umiUnpatchCmdUnicodeArg \sqrt}

After executing this command, the command specified in the argument (\texttt{\sqrt} in this example) can be called with one argument being an Unicode character without needing a brace.

(i.e. you can write \texttt{\sqrt a} instead of \texttt{\sqrt\{a\}}.)

Because of implementation detail,

\begin{itemize}
  \item \texttt{\sqrt a} (without the space between \texttt{\sqrt} and \texttt{a}) works in PDF\TeX{} but not Lua\TeX{}. (so this form is not recommended.)
  \item \texttt{\sqrt a} works in Lua\TeX{} without needing the patch. In other words, the patch does nothing in Unicode engines.
\end{itemize}

The command being patched must take at least one mandatory argument as the first argument, and it only affect that first argument. In other words, \texttt{\sqrt[3]{a}} cannot be patched this way unless you do e.g. \texttt{\newcommand\cbrt[1]{\sqrt[3]{#1}}} then \texttt{\umiPatchCmdUnicodeArg\cbrt}, then \texttt{\cbrt a} works (but \texttt{\sqrt[3]{a}} still doesn’t).

\begin{verbatim}
\sheetTable
\end{verbatim}

\texttt{\umiPatchCmdUnicodeTwoArgs \frac}
\texttt{\umiUnpatchCmdUnicodeTwoArgs \frac}

Similar to above, but for commands with (at least) two mandatory arguments.

Only affects these 2 mandatory arguments.

\begin{verbatim}
\sheetTable
\end{verbatim}

\texttt{\umiPatchCmdUnicodeArgExtraGroup \Big}
\texttt{\umiUnpatchCmdUnicodeArgExtraGroup \Big}

Don’t use this command unless you know exactly what you’re doing.

Similar to \texttt{\umiPatchCmdUnicodeArg}, but open an implicit group before executing anything and close the group after.

The command being patched must take exactly one argument.

This is useful because some \TeX{} primitives such as \texttt{\^} or \texttt{\mathopen} requires either a single “character” or a group braced with \texttt{\{...\}} / \texttt{\bgroup...\egroup}.

Package option. Pass this to avoid patching \texttt{\Big} etc. with the command above (only needed if there’s some package clash).
In the example above, after some steps of execution of \TeX, the state will be
\texttt{abc... {α}xyz...}.

Formally: if the character following the first argument to \texttt{\umiBraceNext} is not rep-
resentable in a single byte and the engine is not Unicode, the character will be braced,
otherwise nothing happens. Then the argument is put back in the input stream.

This is an internal command mainly useful for defining the command above, for
example after
\begin{verbatim}
1 \let\oldbig\big
2 \def\big{\umiBraceNext{\oldbig}}
\end{verbatim}

then \texttt{\big} will eventually execute \texttt{\oldbig{\{} which is the desired behavior (that \texttt{\oldbig}
expects one braced argument).

\texttt{\ignorepatchprime} Do not patch the default definition of ‘ in math mode.  

\begin{itemize}
\item By default it’s patched to allow \texttt{G^+} and \texttt{G^-} to work. Only use this when there’s
some package clash.
\end{itemize}

\texttt{\umipatchprime \umipatchprime} \texttt{\umipatchprime}

As mentioned above, by default \texttt{\umipatchprime} is run \texttt{\AtBeginDocument}. But it can be
patched and unpatched manually.

Note that it’s undefined behavior if some package modifies the definition of active ‘
while it’s patched. To resolve conflict, unpatch ‘, load the package, then patch again.

4 Compatibility

This package should have tested with various \TeX distribution versions on Overleaf.

Index

The italic numbers denote the pages where the corresponding entry is described, numbers
underlined point to the definition, all others indicate the places where it is used.

\begin{verbatim}
I
ignore-patch-delimiter-commands ........ 6
ignore-patch-prime .................... 7
ignore-refresh-delimiter-list .......... 6

U
\umibracenext ......................... 7
\umidefinemathchar .................. 5
\umidefinemathdelimiter ............. 5
\umifrac ................................ 5
\umimathbb ............................ 5
\umimathbb bit ........................ 5
\umimathbfit ........................ 5
\umimathbf .................................. 5
\umimathbf frak ........................ 5
\umimathbf scr ............................ 5
\umimathbf frak .......................... 5
\umimathfrak ............................ 5
\umimathit ............................... 5
\umimathscr ................................ 5
\umimathsf ................................ 5
\umimathsfbf ............................ 5
\umimathsfbfit .......................... 5
\umimathsfit ............................. 5
\end{verbatim}
<table>
<thead>
<tr>
<th>Command</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>\umiMathtt</td>
<td>5</td>
</tr>
<tr>
<td>\umiPatchCmdUnicodeArg</td>
<td>6</td>
</tr>
<tr>
<td>\umiPatchCmdUnicodeArgExtraGroup</td>
<td>6</td>
</tr>
<tr>
<td>\umiPatchCmdUnicodeTwoArgs</td>
<td>6</td>
</tr>
<tr>
<td>\umiRefreshDelimiterList</td>
<td>5</td>
</tr>
<tr>
<td>\umiUnpatchCmdUnicodeArg</td>
<td>6</td>
</tr>
<tr>
<td>\umiUnpatchPrime</td>
<td>7</td>
</tr>
</tbody>
</table>