

The `lualatex-math` package*

Philipp Stephani
st_philipp@yahoo.de

2011/05/05

Contents

1	Introduction	1
2	Interface	2
3	Implementation of the <code>L^AT_EX 2_ε</code> package	2
3.1	Requirements	2
3.2	Messages	3
3.3	Initialization	3
3.4	Patching	3
3.5	<code>L^AT_EX 2_ε</code> kernel	4
3.6	<code>amsmath</code>	5
3.7	<code>mathtools</code>	8
4	Implementation of the <code>LuaL^AT_EX</code> module	9
5	Test files	10
5.1	Common definitions	10
5.2	<code>L^AT_EX 2_ε</code> kernel	13
5.3	<code>amsmath</code> and <code>mathtools</code>	14
5.4	<code>unicode-math</code>	16

1 Introduction

LuaT_EX brings major improvements to all areas of T_EX typesetting and programming. They are made available through new primitives or the embedded Lua interpreter, and combining them with existing L^AT_EX 2_ε package is not a task the average L^AT_EX user should have to care about. Therefore a multitude of L^AT_EX 2_ε packages have been written to bridge the gap between documents and the new features. The `lualatex-math` package focuses on the additional possibilities for mathematical typesetting. The most eminent of the new features is the ability to use Unicode and OpenType fonts, as provided by Will Robertson's `unicode-math` package. However, there is a smaller group of changes unrelated to Unicode: these are to be dealt with in this package. While in principle most T_EX documents written for traditional engines should work just fine with LuaT_EX, there is a small number of breaking changes that require the attention of package authors. The `lualatex-math` package tries to fix some of the issues encountered while porting traditional macro packages to LuaL^AT_EX.

*This document corresponds to `lualatex-math` v0.1, dated 2011/05/05.

The decision to write patches for existing macro packages should not be made lightly: monkey patching done by somebody different from the original package author ties the patching package to the implementation details of the patched functionality and breaks all rules of encapsulation. However, due to the lack of alternatives, it has become an accepted way of providing new functionality in \LaTeX . To keep the negative impact as small as possible, the `lualatex-math` package patches only the $\text{\LaTeX} 2_{\epsilon}$ kernel and a small number of popular packages. In general, this package should be regarded as a temporary kludge that should be removed once the math-related packages are updated to be usable with \LuaTeX . By its very nature, the package is likely to cause problems; in such cases, please refer to the issue tracker¹.

2 Interface

The `lualatex-math` package can be loaded with `\usepackage` or `\RequirePackage`, as usual. It has no options and no public interface; the patching is always done when the package is loaded and cannot be controlled. As a matter of course, the `lualatex-math` package needs \LuaTeX to function; it will produce error messages and refuse to load under other engines and formats. The package depends on the `expl3` bundle, the `etoolbox` package, the `luatexbase` bundle and the `filehook` package. The `lualatex-math` package is independent of the `unicode-math` package; the fixes provided here are valid for both Unicode and legacy math typesetting.

Currently patches for the $\text{\LaTeX} 2_{\epsilon}$ kernel and the `amsmath` and `mathtools` packages are provided. It is not relevant whether you load these packages before or after `lualatex-math`. They should work as expected (and ideally you shouldn't notice anything), but if you load other packages that by themselves overwrite commands patched by this package, bad things may happen, as it is usual with \LaTeX .

One user-visible change is that the new `\mathstyle` primitive (usually called `\luatexmathstyle` in \LuaTeX) should work in all cases after the `lualatex-math` package has been loaded, provided you use the high-level macros `\frac`, `\binom`, and `\genfrac`. The fraction-like \TeX primitives like `\over` or `\atopwithdelims` and the plain \TeX leftovers like `\brack` or `\choose` cannot be patched, and you shouldn't use them.

`\mathstyle`, `\luatexmathstyle`

`\frac`, `\binom`, `\genfrac`

3 Implementation of the $\text{\LaTeX} 2_{\epsilon}$ package

3.1 Requirements

```
1 <*package>
2 \NeedsTeXFormat{LaTeX2e}[2009/09/24]
3 \RequirePackage{expl3}[2011/02/17]
4 \ProvidesExplPackage{lualatex-math}{2011/05/05}{0.1}%
5   {Patches for mathematics typesetting with LuaLaTeX}
6 \RequirePackage { etoolbox } [ 2007/10/08 ]
7 \RequirePackage { luatexbase } [ 2010/05/27 ]
8 \RequirePackage { filehook } [ 2011/03/09 ]
9 \RequireLuaModule { lualatex-math } [ 2011/05/05 ]
```

`\lltxmath_restore_catcode:N` Executing the exhaustive expansion of `\lltxmath_restore_catcode:N` (*character token*) restores the category code of the (*character token*) to its current value.

```
10 \cs_new_nopar:Npn \lltxmath_restore_catcode:N #1 {
11   \char_set_catcode:nn { \int_eval:n { `#1 } }
12   { \char_value_catcode:n { `#1 } }
```

¹<https://github.com/phst/lualatex-math/issues>

```
13 }
```

We use the macro defined above to restore the category code of the dollar sign. There are packages that make the dollar sign active; hopefully they get loaded after the packages we are trying to patch.

```
14 \exp_args:Nx \AtEndOfPackage {
15   \ltxmath_restore_catcode:N \$
16 }
17 \char_make_math_shift:N \$
```

3.2 Messages

luatex-required Issued when not running under LuaTeX.

```
18 \msg_new:nnn { luatex-math } { luatex-required } {
19   The~ luatex-math~ package~ requires~ LuaTeX. \\
20   I~ will~ stop~ loading~ now.
21 }
```

macro-expected Issued when trying to patch a non-macro. The first argument must be the detokenized macro name.

```
22 \msg_new:nnn { luatex-math } { macro-expected } {
23   I've~ expected~ that~ #1~ is~ a~ macro,~ but~ it~ isn't.
24 }
```

wrong-meaning Issued when trying to patch a macro with an unexpected meaning. The first argument must be the detokenized macro name; the second argument must be the actual detokenized meaning; and the third argument must be the expected detokenized meaning.

```
25 \msg_new:nnn { luatex-math } { wrong-meaning } {
26   I've~ expected~ #1~ to~ have~ the~ meaning \\
27   #3, \\
28   but~ it~ has~ the~ meaning \\
29   #2.
30 }
```

patch-macro Issued when a macro is patched. The first argument must be the detokenized macro name.

```
31 \msg_new:nnn { luatex-math } { patch-macro } {
32   I'm~ going~ to~ patch~ macro~ #1.
33 }
```

3.3 Initialization

Unless we are running under LuaTeX, we issue an error and quit immediately. Loading the luatexbase module will already have produced an error, but we issue another one for clarity.

```
34 \luatex_if_engine:F {
35   \msg_error:nn { luatex-math } { luatex-required }
36   \endinput
37 }
```

3.4 Patching

\ltxmath_temp:w A scratch macro.

```
38 \chk_if_free_cs:N \ltxmath_temp:w
```

`\lltxmath_patch:NNnnn` The macro `\lltxmath_patch:NNnnn⟨command⟩⟨factory command⟩⟨parameter text⟩⟨expected replacement text⟩⟨new replacement text⟩` tries to patch `⟨command⟩`. If `⟨command⟩` is undefined, do nothing. Otherwise it must be a macro with the given `⟨parameter text⟩` and `⟨expected replacement text⟩`, created by the given `⟨factory command⟩` or equivalent. In this case it will be overwritten using the `⟨parameter text⟩` and the `⟨new replacement text⟩`. Otherwise issue a warning and don't overwrite.

```

39 \cs_new_protected_nopar:Npn \lltxmath_patch:NNnnn #1 #2 #3 #4 #5 {
40   \cs_if_exist:NT #1 {
41     \token_if_macro:NTF #1 {
42       \group_begin:
43       #2 \lltxmath_temp:w #3 { #4 }
44       \cs_if_eq:NNTF #1 \lltxmath_temp:w {
45         \msg_info:nnx { lualatex-math } { patch-macro }
46         { \token_to_str:N #1 }
47       \group_end:
48       #2 #1 #3 { #5 }
49     } {
50       \msg_warning:nnxxx { lualatex-math } { wrong-meaning }
51       { \token_to_str:N #1 } { \token_to_meaning:N #1 }
52       { \token_to_meaning:N \lltxmath_temp:w }
53     \group_end:
54   }
55 } {
56   \msg_warning:nnx { lualatex-math } { macro-expected }
57   { \token_to_str:N #1 }
58 }
59 }
60 }
61 \cs_generate_variant:Nn \lltxmath_patch:NNnnn { c }

```

3.5 L^AT_EX 2_ε kernel

LuaT_EX enables access to the current mathematical style via the `\mathstyle` primitive. For this to work, fraction-like constructs (e.g., `⟨numerator⟩ \over ⟨denominator⟩`) have to be enclosed in a `\Ustack` group. `\frac` can be patched to do this, but the plain T_EX remnants `\choose`, `\brack` and `\brace` should be discouraged.

`\luatexUstack` First we make sure that we can use the `\Ustack` primitive (under the name `\luatexUstack`).

```

62 \luatexbase@ensure@primitive { Ustack }

```

`\frac` Here we assume that nobody except `amsmath` redefines `\frac`. This is obviously not the case, but we ignore other packages (e.g., `nath`) for the moment. We only patch the L^AT_EX 2_ε kernel definition if the `amsmath` package is not loaded; the corresponding patch for `amsmath` follows below.

```

63 \AtEndPreamble {
64   \@ifpackageloaded { amsmath } { } {
65     \lltxmath_patch:NNnnn \frac \cs_set_nopar:Npn { #1 #2 } {
66       {
67         \begingroup #1 \endgroup \over #2
68       }
69     } {

```

To do: do we need the additional set of braces around `\Ustack`?

```

70     {

```

```

71      \luatexUstack { \group_begin: #1 \group_end: \over #2 }
72    }
73  }
74 }
75 }

```

3.6 amsmath

The popular `amsmath` package is subject to three LuaTeX-related problems:

- The `\mathcode` primitive is used several times, which fails for Unicode math characters. `\Umathcode` should be used instead.
- Legacy font dimensions are used for constructing stacks in the `\substack` command and the `subarray` environment. This doesn't work if a Unicode math font is selected.
- The fraction commands `\frac` and `\genfrac` don't use the `\Ustack` primitive.

`\luatexUmathcodenum` We need the extended versions of `\mathcode` and `\mathchardef`.

```

\luatexUmathchardef 76 \luatexbase@ensure@primitive { Umathcodenum }
                    77 \luatexbase@ensure@primitive { Umathchardef }

```

`\luatexalignmark` We use the primitives corresponding to the alignment mark (#) and to the inline
`\luatexUstartmath` math switches; this is more semantical and might lead to better error messages.

```

\luatexUstopmath 78 \luatexbase@ensure@primitive { alignmark }
                  79 \luatexbase@ensure@primitive { Ustartmath }
                  80 \luatexbase@ensure@primitive { Ustopmath }

```

`\luatexUmathstacknumup` Now we require the font parameters we will use.

```

\luatexUmathstackdenomdown 81 \luatexbase@ensure@primitive { Umathstacknumup }
\luatexUmathstackvgap      82 \luatexbase@ensure@primitive { Umathstackdenomdown }
                           83 \luatexbase@ensure@primitive { Umathstackvgap }

```

`\c_lltxmath_std_minus_mathcode_int` These constants contain the standard TeX mathematical codes for the minus and
`\c_lltxmath_std_equal_mathcode_int` the equal signs. We temporarily set the math codes to these constants before loading
the `amsmath` package so that it can request the legacy math code without error.

```

84 \int_const:Nn \c_lltxmath_std_minus_mathcode_int { "2200 }
85 \int_const:Nn \c_lltxmath_std_equal_mathcode_int { "303D }

```

`\lltxmath_set_mathchar:NN` The macro `\lltxmath_set_mathchar:NN⟨control sequence⟩⟨token⟩` defines the *⟨control sequence⟩* as an extended mathematical character shorthand whose mathematical code is given by the mathematical code of the character ``⟨token⟩`. Since there is no `\Umathcharnumdef` primitive, we have to extract the class, family, and slot numbers separately.

```

86 \cs_new_protected_nopar:Npn \lltxmath_set_mathchar:NN #1 #2 {
87   \luatexUmathchardef #1
88   \lua_now:x {
89     luatex.math.print_class_fam_slot( \int_eval:n { `#2 } )
90   }
91   \scan_stop:
92 }

```

`\lltxmath_char_dim:NN` The macro `\lltxmath_char_dim:NN⟨primitive⟩⟨token⟩` expands to a *⟨dimen⟩* whose value is the metric of the mathematical character corresponding to the character ``⟨token⟩` specified by *⟨primitive⟩*, which must be one of `\fontcharwd`, `\fontcharht` or `\fontchardp`, in the currently selected text style font.

```

93 \cs_new_nopar:Npn \lltxmath_char_dim:NN #1 #2 {
94   #1 \textfont
95   \lua_now:x {
96     lualatex.math.print_fam_slot( \int_eval:n { `#2 } )
97   }
98 }

```

`\l_lltxmath_minus_mathchar` `\l_lltxmath_equal_mathchar` These mathematical characters are saved before `amsmath` is loaded so that we can temporarily assign the \TeX values to the mathematical codes of the minus and equals signs. The `amsmath` package queries these codes, and if they represent Unicode characters, the package loading will fail. If `amsmath` has already been loaded, there is nothing we can do, therefore we use the non-starred version of `\AtBeginOfPackageFile`.

```

99 \chk_if_free_cs:N \l_lltxmath_minus_mathchar
100 \chk_if_free_cs:N \l_lltxmath_equal_mathchar
101 \AtBeginOfPackageFile { amsmath } {
102   \lltxmath_set_mathchar:NN \l_lltxmath_minus_mathchar \-
103   \lltxmath_set_mathchar:NN \l_lltxmath_equal_mathchar \=

```

Now we temporarily reset the mathematical codes.

```

104 \char_set_mathcode:nn { \- } { \c_lltxmath_std_minus_mathcode_int }
105 \char_set_mathcode:nn { \= } { \c_lltxmath_std_equal_mathcode_int }
106 \AtEndOfPackageFile { amsmath } {

```

`\std@minus` `\std@equals` The `amsmath` package defines the control sequences `\std@minus` and `\std@equal` as mathematical character shorthands while loading, but uses our restored mathematical codes, which must be fixed.

```

107 \cs_set_eq:NN \std@minus \l_lltxmath_minus_mathchar
108 \cs_set_eq:NN \std@equal \l_lltxmath_equal_mathchar

```

Finally, we restore the original mathematical codes of the two signs.

```

109 \luatexUmathcodenum \- \l_lltxmath_minus_mathchar
110 \luatexUmathcodenum \= \l_lltxmath_equal_mathchar
111 }
112 }

```

All of the following fixes work even if `amsmath` is already loaded.

`\@begindocumenthook` `amsmath` repeats the definition of `\std@minus` and `\std@equal` at the beginning of the document, so we also have to patch the internal kernel macro `\@begindocumenthook` which contains the hook code.

```

113 \AtEndOfPackageFile * { amsmath } {
114   \tl_replace_in:Nnn \@begindocumenthook {
115     \mathchardef \std@minus \mathcode \- \relax
116     \mathchardef \std@equal \mathcode \= \relax
117   } {
118     \lltxmath_set_mathchar:NN \std@minus \-
119     \lltxmath_set_mathchar:NN \std@equal \=
120   }

```

`\resetMathstrut@` `amsmath` uses the box `\Mathstrutbox@` for struts in mathematical mode. This box is defined to have the height and depth of the opening parenthesis taken from the current text font. The command `\resetMathstrut@` is executed whenever the mathematical fonts are changed and has to restore the correct dimensions. The original definition uses a temporary mathematical character shorthand definition whose meaning is queried to extract the family and slot. We can do this in Lua;

furthermore we can avoid a temporary box because ε -TeX allows us to query glyph metrics directly.

```

121 \lltxmath_patch:NNnnn \resetMathstrut@ \cs_set_nopar:Npn { } {
122   \setbox \z@ \hbox {
123     \mathchardef \@tempa \mathcode `\< \relax % \)
124     \def \@tempb ##1 "##2 ##3 { \the \textfont "##3 \char" }
125     \expandafter \@tempb \meaning \@tempa \relax
126   }
127   \ht \Mathstrutbox@ \ht \z@
128   \dp \Mathstrutbox@ \dp \z@
129 } {
130   \box_set_ht:Nn \Mathstrutbox@ {
131     \lltxmath_char_dim:NN \fontcharht \{ % \)
132   }
133   \box_set_dp:Nn \Mathstrutbox@ {
134     \lltxmath_char_dim:NN \fontchardp \)
135   }
136 }

```

subarray The `subarray` environment uses legacy font dimensions. We simply patch it to use LuaTeX font parameters (and L^AT_EX3 expressions instead of T_EX arithmetic). Since subscript arrays are conceptually vertical stacks, we use the sum of top and bottom shift for the default vertical baseline distance (`\baselineskip`) and the minimum vertical gap for stack for the minimum baseline distance (`\lineskip`).

```

137 \lltxmath_patch:NNnnn \subarray \cs_set:Npn { #1 } {
138   \vcenter
139   \bgroup
140   \Let@
141   \restore@math@cr
142   \default@tag
143   \baselineskip \fontdimen 10~ \scriptfont \tw@
144   \advance \baselineskip \fontdimen 12~ \scriptfont \tw@
145   \lineskip \thr@@ \fontdimen 8~ \scriptfont \thr@@
146   \lineskiplimit \lineskip
147   \ialign
148   \bgroup
149   \ifx c #1 \hfil \fi
150   $ \m@th \scriptstyle ## $
151   \hfil
152   \crcr
153 } {
154   \vcenter
155   \c_group_begin_token
156   \Let@
157   \restore@math@cr
158   \default@tag
159   \skip_set:Nn \baselineskip {
160     \luatexUmathstacknumup \scriptstyle
161     + \luatexUmathstackdenomdown \scriptstyle
162   }
163   \lineskip \luatexUmathstackvgap \scriptstyle
164   \lineskiplimit \lineskip
165   \ialign
166   \c_group_begin_token
167   \token_if_eq_meaning:NNT c #1 { \hfil }
168   \luatexUstartmath
169   \m@th
170   \scriptstyle

```

```

171 \luatexalignmark \luatexalignmark
172 \luatexUstopmath
173 \hfil
174 \crrcr
175 }

```

\frac Since \frac is declared by \DeclareRobustCommand, we must patch the macro \frac.

```

176 \ltxmath_patch:cNnnn { frac~ } \cs_set:Npn { #1 #2 } {
177   {
178     \begingroup #1 \endgroup @@over #2
179   }
180 } {
181   {
182     \luatexUstack { \group_begin: #1 \group_end: @@over #2 }
183   }
184 }

```

\@genfrac Generalized fractions are typeset by the internal \@genfrac command.

```

185 \ltxmath_patch:NNnnn \@genfrac \cs_set_nopar:Npn {
186   #1 #2 #3 #4 #5
187 } {
188   {
189     #1 { \begingroup #4 \endgroup #2 #3 \relax #5 }
190   }
191 } {
192   {
193     #1 {
194       \luatexUstack {
195         \group_begin: #4 \group_end: #2 #3 \scan_stop: #5
196       }
197     }
198   }
199 }
200 }

```

3.7 mathtools

mathtools' \cramped command and others that make use of its internal version use a hack involving a null radical. LuaTeX has primitives for setting material in cramped mode, so we make use of them.

```

\luatexcrampeddisplaystyle First we make sure that the needed primitives for cramped styles are available.
\luatexcrampedtextstyle 201 \luatexbase@ensure@primitive { crampeddisplaystyle }
\luatexcrampedscriptstyle 202 \luatexbase@ensure@primitive { crampedtextstyle }
\luatexcrampedscriptscriptstyle 203 \luatexbase@ensure@primitive { crampedscriptstyle }
204 \luatexbase@ensure@primitive { crampedscriptscriptstyle }

```

\MT_cramped_internal:Nn The macro \MT_cramped_internal:Nn<style>{\<expression>} typesets the <expression> in the cramped style corresponding to the given <style> (\displaystyle etc.); all we have to do in LuaTeX is to select the correct primitive. Rewriting the user-level \cramped command and employing \mathstyle would be possible as well, but we avoid this way since we want to patch only a single command.

```

205 \AtEndOfPackageFile * { mathtools } {
206   \ltxmath_patch:NNnnn \MT_cramped_internal:Nn
207   \cs_set_nopar:Npn { #1 #2 } {
208     \sbox \z@ {

```



```

209      $
210      \m@th
211      #1
212      \nulldelimiterspace = \z@
213      \radical \z@ { #2 }
214      $
215  }
216  \ifx #1 \displaystyle
217      \dimen@ = \fontdimen 8 \textfont 3
218      \advance \dimen@ .25 \fontdimen 5 \textfont 2
219  \else
220      \dimen@ = 1.25 \fontdimen 8
221      \ifx #1 \textstyle
222          \textfont
223      \else
224          \ifx #1 \scriptstyle
225              \scriptfont
226          \else
227              \scriptscriptfont
228          \fi
229      \fi
230      3
231  \fi
232  \advance \dimen@ -\ht\z@
233  \ht\z@ = -\dimen@
234  \box\z@
235  } {

```

Here the additional set of braces is absolutely necessary, otherwise the changed mathematical style would be applied to the material after the `\mathchoice` construct.

```

236  {
237      \use:c { luatexcramped \cs_to_str:N #1 } #2
238  }
239  }
240 }
241 </package>

```

4 Implementation of the Lua_{La}T_EX module

For the Lua module, we use the standard `luatexbase-modutils` template and the `module` function.

```

242 <*lua>
243 require("luatexbase.modutils")
244 require("luatexbase.cctb")
245 local err, warn, info, log = luatexbase.provides_module({
246     name = "lualatex-math",
247     date = "2011/05/05",
248     version = 0.1,
249     description = "Patches for mathematics typesetting with LuaLaTeX",
250     author = "Philipp Stephani",
251     licence = "LPPL v1.3+"
252 })
253 local unpack = unpack
254 local string = string
255 local tex = tex
256 local cctb = luatexbase.catcodetables

```

```

257 module("lualatex.math")

print_fam_slot The function print_fam_slot takes one argument which must be a number.
                It interprets the argument as a Unicode code point whose mathematical code
                is printed in the form  $\langle family \rangle_{\langle slot \rangle}$ , suitable for the right-hand side of e.g.
                \fontcharht\textfont.
258 function print_fam_slot(char)
259   local code = tex.getmathcode(char)
260   local class, family, slot = unpack(code)
261   local result = string.format("%i %i ", family, slot)
262   tex.sprint(cctb.string, result)
263 end

print_class_fam_slot The function print_class_fam_slot takes one argument which must be a number.
                     It interprets the argument as a Unicode code point whose mathematical code
                     is printed in the form  $\langle class \rangle_{\langle family \rangle_{\langle slot \rangle}}$ , suitable for the right-hand side of
                     \Umathchardef.
264 function print_class_fam_slot(char)
265   local code = tex.getmathcode(char)
266   local class, family, slot = unpack(code)
267   local result = string.format("%i %i %i ", class, family, slot)
268   tex.sprint(cctb.string, result)
269 end
270 \lua

```

5 Test files

Finally two small test files—but not a real test suite.

5.1 Common definitions

```

271 *test
272 \documentclass[pagesize=auto]{scrartcl}

Only xparse starting with 2008/08/03 has \NewDocumentCommand.
273 \usepackage{xparse}[2008/08/03]
274 \ExplSyntaxOn

pass This message is issued when a test passed.
275 \msg_new:nnn { test } { pass } { #1 }

\test_pass:x The macro \test_pass:x{<text>} issues the pass message with description <text>.
276 \cs_new_protected_nopar:Npn \test_pass:x #1 {
277   \msg_info:nnx { test } { pass } { #1 }
278 }

fail This message is issued when a test failed.
279 \msg_new:nnn { test } { fail } { #1 }

\test_fail:x The macro \test_fail:x{<text>} issues the fail message with description <text>.
280 \cs_new_protected_nopar:Npn \test_fail:x #1 {
281   \msg_error:nnx { test } { fail } { #1 }
282 }

\tl_const:Nx We need expanding constants.
283 \cs_generate_variant:Nn \tl_const:Nn { Nx }

```

`\c_test_equal_tl` Two shorthands for pretty-printing test results.

`\c_test_not_equal_tl` 284 `\tl_const:Nx \c_test_equal_tl { \c_space_tl == \c_space_tl }`
285 `\tl_const:Nx \c_test_not_equal_tl { \c_space_tl != \c_space_tl }`

`\test_equal_pass:nxn` The macro `\test_equal_pass:nxn{⟨first expression⟩}{⟨first value⟩}{⟨second expression⟩}{⟨second value⟩}` is called when the two values arising from the two expressions are equal.

286 `\cs_new_protected_nopar:Npn \test_equal_pass:nxn #1 #2 #3 #4 {`
287 `\test_pass:x {`
288 `\exp_not:n { #1 }`
289 `\c_test_equal_tl`
290 `#2`
291 `\c_test_equal_tl`
292 `#4`
293 `\c_test_equal_tl`
294 `\exp_not:n { #3 }`
295 `}`
296 `}`

`\test_equal_fail:nxn` The macro `\test_equal_pass:nxn{⟨first expression⟩}{⟨first value⟩}{⟨second expression⟩}{⟨second value⟩}` is called when the two values arising from the two expressions are not equal.

297 `\cs_new_protected_nopar:Npn \test_equal_fail:nxn #1 #2 #3 #4 {`
298 `\test_fail:x {`
299 `\exp_not:n { #1 }`
300 `\c_test_equal_tl`
301 `#2`
302 `\c_test_not_equal_tl`
303 `#4`
304 `\c_test_equal_tl`
305 `\exp_not:n { #3 }`
306 `}`
307 `}`

`\test_assert_equal:NNNNNnn` The macro `\test_assert_equal:NNNNNnn⟨set command⟩⟨use command⟩⟨compare command⟩⟨first temporary command⟩⟨second temporary command⟩{⟨first expression⟩}{⟨second expression⟩}` asserts that the two expressions are equal. The `⟨set command⟩` must have the argument specification `Nn`, the `⟨use command⟩` `N`, and the `⟨compare command⟩` `nNnTF`.

`\test_assert_equal:ccccnn`

308 `\cs_new_protected_nopar:Npn`
309 `\test_assert_equal:NNNNNnn #1 #2 #3 #4 #5 #6 #7 {`
310 `#1 #4 { #6 }`
311 `#1 #5 { #7 }`
312 `#3 { #4 } = { #5 } {`
313 `\test_equal_pass:nxn { #6 } { #2 #4 } { #7 } { #2 #5 }`
314 `}`
315 `\test_equal_fail:nxn { #6 } { #2 #4 } { #7 } { #2 #5 }`
316 `}`
317 `}`
318 `\cs_generate_variant:Nn \test_assert_equal:NNNNNnn { ccccc }`

`\test_assert_equal:nnn` The macro `\test_assert_equal:nnn{⟨data type⟩}{⟨first expression⟩}{⟨second expression⟩}` is a simplified version of `\test_assert_equal:NNNNNnn` for data types following the L^AT_EX3 naming conventions; `⟨data type⟩` must be `int`, `dim`, etc.

319 `\cs_new_protected_nopar:Npn \test_assert_equal:nnn #1 #2 #3 {`
320 `\test_assert_equal:ccccnn`
321 `{ #1 _set:Nn } { #1 _use:N } { #1 _compare:nNnTF }`

```

322 { \l_test_tmpa_ #1 } { \l_test_tmppb_ #1 } { #2 } { #3 }
323 }

\l_test_tmpa_int Scratch registers for numbers.
\l_test_tmppb_int 324 \int_new:N \l_test_tmpa_int
325 \int_new:N \l_test_tmppb_int

\AssertIntEqual The command \AssertIntEqual{<first expression>}{<second expression>} asserts
that the two integral expressions are equal.
326 \NewDocumentCommand \AssertIntEqual { m m } {
327 \test_assert_equal:nnn { int } { #1 } { #2 }
328 }

\l_test_tmpa_int Scratch registers for dimensions.
\l_test_tmppb_int 329 \dim_new:N \l_test_tmpa_dim
330 \dim_new:N \l_test_tmppb_dim

\AssertIntEqual The command \AssertDimEqual{<first expression>}{<second expression>} asserts
that the two dimension expressions are equal.
331 \NewDocumentCommand \AssertDimEqual { m m } {
332 \test_assert_equal:nnn { dim } { #1 } { #2 }
333 }

\AssertMathStyle The command \AssertMathStyle{<expression>} asserts that the current mathe-
matical style is equal to the value of the integral <expression>.
334 \NewDocumentCommand \AssertMathStyle { m } {
335 \AssertIntEqual { \luatexmathstyle } { #1 }
336 }

\test_assert_cramped:Nx The macro \test_assert_cramped:Nn<predicate>{<name>} asserts that we are in
math mode and that the current style fulfills the <predicate> (identified by the
<name>) which must have the argument specification n.
337 \cs_new_protected_nopar:Npn \test_assert_cramped:Nx #1 #2 {
338 \int_set:Nn \l_test_tmpa_int { \luatexmathstyle }
339 \bool_if:nTF {
340 \int_compare_p:nNn { \l_test_tmpa_int } > { \c_minus_one }
341 &&
342 #1 { \l_test_tmpa_int }
343 } {
344 \test_pass:x {
345 \exp_not:N \luatexmathstyle
346 \c_test_equal_tl
347 \int_use:N \l_test_tmpa_int
348 \c_space_tl
349 is~ a~ #2~ style
350 }
351 } {
352 \test_fail:x {
353 \exp_not:N \luatexmathstyle
354 \c_test_equal_tl
355 \int_use:N \l_test_tmpa_int
356 \c_space_tl
357 is~ not~ a~ #2~ style
358 }
359 }
360 }

```

`\AssertNoncrampedStyle` The command `\AssertNoncrampedStyle` asserts that the current mathematical style is one of the non-cramped styles.

```
361 \NewDocumentCommand \AssertNoncrampedStyle { } {
362   \test_assert_cramped:Nx \int_if_even_p:n { non-cramped }
363 }
```

`\AssertCrampedStyle` The command `\AssertCrampedStyle` asserts that the current mathematical style is one of the cramped styles.

```
364 \NewDocumentCommand \AssertCrampedStyle { } {
365   \test_assert_cramped:Nx \int_if_odd_p:n { cramped }
366 }
367 \ExplSyntaxOff
368 \end{test}
```

5.2 L^AT_EX 2_ε kernel

Here we only check whether different fractions and other style-changing commands result in the correct mathematical style.

```
369 <*test-kernel>
370 \usepackage{lualatex-math}
371 \begin{document}
372 \begin{displaymath}
373   \AssertMathStyle{0} \sqrt{\AssertMathStyle{1}}
374   \frac{\AssertMathStyle{2}}{\AssertMathStyle{3}}
375   a^{\frac{\AssertMathStyle{6}}{\AssertMathStyle{7}}}
376   \sqrt{\frac{\AssertMathStyle{3}}{\AssertMathStyle{3}}}
377   \displaystyle
378   \frac{\AssertMathStyle{2}}{\AssertMathStyle{3}}
379   \lualatexcrampeddisplaystyle
380   \frac{\AssertMathStyle{3}}{\AssertMathStyle{3}}
381   \textstyle
382   \frac{\AssertMathStyle{4}}{\AssertMathStyle{5}}
383   \lualatexcrampedtextstyle
384   \frac{\AssertMathStyle{5}}{\AssertMathStyle{5}}
385   \scriptstyle
386   \frac{\AssertMathStyle{6}}{\AssertMathStyle{7}}
387   \lualatexcrampedscriptstyle
388   \frac{\AssertMathStyle{7}}{\AssertMathStyle{7}}
389 \end{displaymath}
390 \begin{math}
391   \AssertMathStyle{2} \sqrt{\AssertMathStyle{3}}
392   \frac{\AssertMathStyle{4}}{\AssertMathStyle{5}}
393   a^{\frac{\AssertMathStyle{6}}{\AssertMathStyle{7}}}
394   \sqrt{\frac{\AssertMathStyle{5}}{\AssertMathStyle{5}}}
395   \displaystyle
396   \frac{\AssertMathStyle{2}}{\AssertMathStyle{3}}
397   \lualatexcrampeddisplaystyle
398   \frac{\AssertMathStyle{3}}{\AssertMathStyle{3}}
399   \textstyle
400   \frac{\AssertMathStyle{4}}{\AssertMathStyle{5}}
401   \lualatexcrampedtextstyle
402   \frac{\AssertMathStyle{5}}{\AssertMathStyle{5}}
403   \scriptstyle
404   \frac{\AssertMathStyle{6}}{\AssertMathStyle{7}}
405   \lualatexcrampedscriptstyle
406   \frac{\AssertMathStyle{7}}{\AssertMathStyle{7}}
407 \end{math}
```

```

408 \end{document}
409 </test-kernel>

```

5.3 amsmath and mathtools

Since mathtools loads amsmath anyway, we test both in one file.

\testbox First a scratch box register.

```

410 <*test-amsmath>
411 \usepackage{luatex-math}
412 \newsavebox{\testbox}

```

We set the mathematical code for the minus sign to some arbitrary Unicode value to test whether the load-time patch works.

```

413 \luatexUmathcode`\-="2 "33 "44444 \relax
414 \usepackage{amsmath}
415 \AssertIntEqual{\luatexUmathcode`\-}{33444444}
416 \makeatletter
417 \AssertIntEqual{\std@minus}{33444444}
418 \makeatother
419 \usepackage{mathtools}

```

The same for the document begin hook.

```

420 \luatexUmathcode`\="5 "66 "77777 \relax
421 \begin{document}
422 \AssertIntEqual{\luatexUmathcode`\=}{66A77777}
423 \makeatletter
424 \AssertIntEqual{\std@equal}{66A77777}
425 \makeatother

```

Here we test whether the strut box has the correct height and depth.

```

426 \sbox{\testbox}{$(\$) \% }
427 \makeatletter
428 \AssertDimEqual{\ht\Mathstrutbox@}{\ht\testbox}
429 \AssertDimEqual{\dp\Mathstrutbox@}{\dp\testbox}
430 \makeatother

```

Here we test for the various amsmath features that have to be patched: sub-arrays and various kind of fraction-like objects. The \substack command and subarray environment aren't really tested since it is hard to check whether the outcome looks right in an automated way. All tests are done in both inline and display mode.

```

431 \begin{equation*}
432 \AssertMathStyle{0} \sqrt{\AssertMathStyle{1}}
433 \sum_{
434 \substack{\frac{1}{2} \ \ \frac{3}{4} \ \ \frac{5}{6}}
435 }
436 \sum_{
437 \begin{subarray}{l} \frac{1}{2} \ \ \frac{3}{4} \ \ \frac{5}{6} \end{subarray}}
438 }
439 \frac{\AssertMathStyle{2}}{\AssertMathStyle{3}}
440 a^{\frac{\AssertMathStyle{6}}{\AssertMathStyle{7}}}
441 \dfrac{\AssertMathStyle{2}}{\AssertMathStyle{3}}
442 \tfrac{\AssertMathStyle{4}}{\AssertMathStyle{5}}
443 \binom{\AssertMathStyle{2}}{\AssertMathStyle{3}}
444 a^{\binom{\AssertMathStyle{6}}{\AssertMathStyle{7}}}
445 \dbinom{\AssertMathStyle{2}}{\AssertMathStyle{3}}
446 \tbinom{\AssertMathStyle{4}}{\AssertMathStyle{5}}
447 \genfrac{}{}{}{\AssertMathStyle{2}}{\AssertMathStyle{3}}
448 \genfrac{<}{/}{0}{\AssertMathStyle{2}}{\AssertMathStyle{3}}

```

```

449 \genfrac{}{}{1}{\AssertMathStyle{4}}{\AssertMathStyle{5}}
450 \genfrac{}{}{4pt}{2}{\AssertMathStyle{6}}{\AssertMathStyle{7}}
451 \genfrac{}{}{3}{\AssertMathStyle{6}}{\AssertMathStyle{7}}
452 \end{equation*}
453 \begin{math}
454 \AssertMathStyle{2} \sqrt{\AssertMathStyle{3}}
455 \sum_{
456 \substack{\frac{1}{2} \ \ \frac{3}{4} \ \ \frac{5}{6}}
457 }
458 \sum_{
459 \begin{subarray}{l} \frac{1}{2} \ \ \frac{3}{4} \ \ \frac{5}{6} \end{subarray}
460 }
461 \frac{\AssertMathStyle{4}}{\AssertMathStyle{5}}
462 a^{\frac{\AssertMathStyle{6}}{\AssertMathStyle{7}}}
463 \dfrac{\AssertMathStyle{2}}{\AssertMathStyle{3}}
464 \tfrac{\AssertMathStyle{4}}{\AssertMathStyle{5}}
465 \binom{\AssertMathStyle{4}}{\AssertMathStyle{5}}
466 a^{\binom{\AssertMathStyle{6}}{\AssertMathStyle{7}}}
467 \dbinom{\AssertMathStyle{2}}{\AssertMathStyle{3}}
468 \tbinom{\AssertMathStyle{4}}{\AssertMathStyle{5}}
469 \genfrac{}{}{}{}{\AssertMathStyle{4}}{\AssertMathStyle{5}}
470 \genfrac{}{}{}{0pt}{0}{\AssertMathStyle{2}}{\AssertMathStyle{3}}
471 \genfrac{}{}{}{1}{\AssertMathStyle{4}}{\AssertMathStyle{5}}
472 \genfrac{}{}{}{4pt}{2}{\AssertMathStyle{6}}{\AssertMathStyle{7}}
473 \genfrac{}{}{}{3}{\AssertMathStyle{6}}{\AssertMathStyle{7}}
474 \end{math}

```

Since mathtools' `\cramped` command uses `\mathchoice`, we cannot test for a single mathematical style since all of them are executed; instead, we just verify that all styles encountered are cramped.

```

475 \begin{equation*}
476 \AssertMathStyle{0}
477 a^{\AssertMathStyle{4}} a
478 \cramped{\AssertCrampedStyle a^{\AssertCrampedStyle a}}
479 a^{
480 \AssertMathStyle{4}
481 a^a
482 \cramped{\AssertCrampedStyle a^{\AssertCrampedStyle a}}
483 a^a
484 \AssertMathStyle{4}
485 }
486 a^{
487 a^{
488 \AssertMathStyle{6}
489 a^a
490 \cramped{\AssertCrampedStyle a^{\AssertCrampedStyle a}}
491 a^a
492 \AssertMathStyle{6}
493 }
494 }
495 a^{\AssertMathStyle{4}} a
496 \AssertMathStyle{0}
497 \end{equation*}
498 \begin{math}
499 \AssertMathStyle{2}
500 a^{\AssertMathStyle{4}} a
501 \cramped{\AssertCrampedStyle a^{\AssertCrampedStyle a}}
502 a^{
503 \AssertMathStyle{4}

```

```

504   a^a
505   \cramped{\AssertCrampedStyle a^{\AssertCrampedStyle a}}
506   a^a
507   \AssertMathStyle{4}
508 }
509 a^{
510   a^{
511     \AssertMathStyle{6}
512     a^a
513     \cramped{\AssertCrampedStyle a^{\AssertCrampedStyle a}}
514     a^a
515     \AssertMathStyle{6}
516   }
517 }
518 a^{\AssertMathStyle{4} a}
519 \AssertMathStyle{2}
520 \end{math}
521 \end{document}
522 </test-amsmath>

```

5.4 unicode-math

This test file loads both `amsmath` and `unicode-math`. The latter package contains fixes that somewhat overlap with ours. We have to take care in all packages that no attempt is made to patch a single macro twice. Therefore we treat warnings (that occur when trying to patch a macro with an unknown meaning) as errors here.

```

523 <*test-unicode>
524 \ExplSyntaxOn
525 \msg_redirect_class:nn { warning } { error }
526 \ExplSyntaxOff
527 \usepackage{amsmath}
528 \usepackage{unicode-math}[2011/05/05]
529 \setmathfont{XITS Math}
530 \usepackage{lualatex-math}
531 \begin{document}
532 \begin{equation*}
533   \AssertMathStyle{0} \sqrt{\AssertMathStyle{1}}
534   \frac{\AssertMathStyle{2}}{\AssertMathStyle{3}}
535   a^{\frac{\AssertMathStyle{6}}{\AssertMathStyle{7}}}
536   \dfrac{\AssertMathStyle{2}}{\AssertMathStyle{3}}
537   \tfrac{\AssertMathStyle{4}}{\AssertMathStyle{5}}
538 \end{equation*}
539 \end{document}
540 </test-unicode>

```

Change History

v0.1

General: Initial version 1

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols	
$\backslash$$	15, 17
$\backslash($	123, 131
$\backslash)$	123, 131, 134
$\backslash-$	102, 104, 109, 115, 118, 413, 415
$\backslash=$	103, 105, 110, 116, 119, 420, 422
$\backslash@@over$	178, 182
$\backslash@begindocumenthook$	113
$\backslash@genfrac$	185
$\backslash@ifpackageloaded$	64
$\backslash@tempa$	123, 125
$\backslash@tempb$	124, 125
$\backslash\backslash$	19, 26, 27, 28, 434, 437, 456, 459
A	
\backslashadvance	144, 218, 232
amsmath (package)	1, 2, 4--6, 14, 16
$\backslashAssertCrampedStyle$	364, 478, 482, 490, 501, 505, 513
\backslashAssertDimEqual	331, 428, 429
\backslashAssertIntEqual	326, 331, 335, 415, 417, 422, 424
$\backslashAssertMathStyle$	334, 373, 374, 375, 376, 378, 380, 382, 384, 386, 388, 391, 392, 393, 394, 396, 398, 400, 402, 404, 406, 432, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 454, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 476, 477, 480, 484, 488, 492, 495, 496, 499, 500, 503, 507, 511, 515, 518, 519, 533, 534, 535, 536, 537
$\backslashAssertNoncrampedStyle$	361
$\backslashAtBeginOfPackageFile$	101
\backslashAtEndOfPackage	14
$\backslashAtEndOfPackageFile$	106, 113, 205
\backslashAtEndPreamble	63
B	
\backslashbaselineskip	143, 144, 159
\backslashbegin	371, 372, 390, 421, 431, 437, 453, 459, 475, 498, 531, 532
\backslashbegingroup	67, 178, 189
\backslashbgroup	139, 148
\backslashbinom	2, 443, 444, 465, 466
$\backslashbool_if:nTF$	339
\backslashbox	234
$\backslashbox_set_dp:Nn$	133
$\backslashbox_set_ht:Nn$	130
C	
$\backslashc_group_begin_token$	155, 166
$\backslashc_lltxmath_std_equal_mathcode_int$	84, 105
$\backslashc_lltxmath_std_minus_mathcode_int$	84, 104
\backslashc_minus_one	340
\backslashc_space_tl	284, 285, 348, 356
$\backslashc_test_equal_tl$	284, 289, 291, 293, 300, 304, 346, 354
$\backslashc_test_not_equal_tl$	284, 302
\backslashchar	124
$\backslashchar_make_math_shift:N$	17
$\backslashchar_set_catcode:n$	11
$\backslashchar_set_mathcode:n$	104, 105
$\backslashchar_value_catcode:n$	12
$\backslashchk_if_free_cs:N$	38, 99, 100
\backslashcramped	478, 482, 490, 501, 505, 513
\backslashcrrc	152, 174
$\backslashcs_generate_variant:Nn$	61, 283, 318
$\backslashcs_if_eq:NNTF$	44
$\backslashcs_if_exist:NT$	40
$\backslashcs_new_nopar:Npn$	10, 93
$\backslashcs_new_protected_nopar:Npn$	39, 86, 276, 280, 286, 297, 308, 319, 337
$\backslashcs_set:Npn$	137, 176
$\backslashcs_set_eq:NN$	107, 108
$\backslashcs_set_nopar:Npn$	65, 121, 185, 207
$\backslashcs_to_str:N$	237
D	
\backslashdbinom	445, 467
\backslashdef	124
$\backslashdefault@tag$	142, 158
\backslashdfrac	441, 463, 536
$\backslashdim_new:N$	329, 330
$\backslashdimen@$	217, 218, 220, 232, 233
\backslashdisplaystyle	216, 377, 395
\backslashdocumentclass	272
\backslashdp	128, 429
E	
\backslashelse	219, 223, 226
\backslashend	389, 407, 408, 437, 452, 459, 474, 497, 520, 521, 538, 539
\backslashendgroup	67, 178, 189
\backslashendinput	36
environments:	
\backslashsubarray	137
etoolbox (package)	2
$\backslashexp_args:Nx$	14
$\backslashexp_not:N$	345, 353
$\backslashexp_not:n$	288, 294, 299, 305
\backslashexpandafter	125
expl3 (package)	2
\backslashExplSyntaxOff	367, 526
\backslashExplSyntaxOn	274, 524

F		<code>\luatex_if_engine:F</code> 34	
<code>fail (message)</code> 10, 279		<code>\luatexalignmark</code> 78, 171	
<code>\fi</code> 149, 228, 229, 231		luatexbase (package) 2, 3	
filehook (package) 2		luatexbase-modutils (package) 9	
<code>\fontchardp</code> 134		<code>\luatexbase@ensure@primitive</code> 62, 76, 77, 78, 79, 80, 81, 82, 83, 201, 202, 203, 204	
<code>\fontcharht</code> 131		<code>\luatexcrampeddisplaystyle</code> 201, 379, 397	
<code>\fontdimen</code> 143, 144, 145, 217, 218, 220		<code>\luatexcrampedscriptscriptstyle</code> 201	
<code>\frac</code> 2, 63, 176, 374, 375, 376, 378, 380, 382, 384, 386, 388, 392, 393, 394, 396, 398, 400, 402, 404, 406, 434, 437, 439, 440, 456, 459, 461, 462, 534, 535		<code>\luatexcrampedscriptstyle</code> 201, 387, 405	
functions:		<code>\luatexcrampedtextstyle</code> 201, 383, 401	
module 9		<code>\luatexmathstyle</code> 2, 335, 338, 345, 353	
<code>print_class_fam_slot</code> 10, 264		<code>\luatexUmathchardef</code> 76, 87	
<code>print_fam_slot</code> 10, 258		<code>\luatexUmathcode</code> 413, 415, 420, 422	
G		<code>\luatexUmathcodenum</code> 76, 109, 110	
<code>\genfrac</code> 2, 447, 448, 449, 450, 451, 469, 470, 471, 472, 473		<code>\luatexUmathstackdenomdown</code> 81, 161	
<code>\group_begin:</code> 42, 71, 182, 195		<code>\luatexUmathstacknumup</code> 81, 160	
<code>\group_end:</code> 47, 53, 71, 182, 195		<code>\luatexUmathstackvgap</code> 81, 163	
H		<code>\luatexUstack</code> 62, 71, 182, 194	
<code>\hbox</code> 122		<code>\luatexUstartmath</code> 78, 168	
<code>\hfil</code> 149, 151, 167, 173		<code>\luatexUstopmath</code> 78, 172	
<code>\ht</code> 127, 232, 233, 428		M	
I		<code>\m@th</code> 150, 169, 210	
<code>\ialign</code> 147, 165		macro-expected (message) 22	
<code>\ifx</code> 149, 216, 221, 224		<code>\makeatletter</code> 416, 423, 427	
<code>\int_compare_p:nNn</code> 340		<code>\makeatother</code> 418, 425, 430	
<code>\int_const:Nn</code> 84, 85		<code>\mathchardef</code> 115, 116, 123	
<code>\int_eval:n</code> 11, 89, 96		<code>\mathcode</code> 115, 116, 123	
<code>\int_if_even_p:n</code> 362		<code>\Mathstrutbox@</code> 127, 128, 130, 133, 428, 429	
<code>\int_if_odd_p:n</code> 365		<code>\mathstyle</code> 2	
<code>\int_new:N</code> 324, 325		mathtools (package) 1, 2, 8, 14, 15	
<code>\int_set:Nn</code> 338		<code>\meaning</code> 125	
<code>\int_use:N</code> 347, 355		messages:	
L		<code>fail</code> 10, 279	
<code>\l_lltxmath_equal_mathchar</code> 99, 108, 110		<code>luatex-required</code> 18	
<code>\l_lltxmath_minus_mathchar</code> 99, 107, 109		macro-expected 22	
<code>\l_test_tmpa_dim</code> 329		<code>pass</code> 10, 275	
<code>\l_test_tmpa_int</code> 324, 329, 338, 340, 342, 347, 355		<code>patch-macro</code> 31	
<code>\l_test_tmpb_dim</code> 330		<code>wrong-meaning</code> 25	
<code>\l_test_tmpb_int</code> 324, 329		module (function) 9	
<code>\Let@</code> 140, 156		<code>\msg_error:nn</code> 35	
<code>\lineskip</code> 145, 146, 163, 164		<code>\msg_error:nnx</code> 281	
<code>\lineskiplimit</code> 146, 164		<code>\msg_info:nnx</code> 45, 277	
<code>\lltxmath_char_dim:NN</code> 93, 131, 134		<code>\msg_new:nnn</code> 18, 22, 25, 31, 275, 279	
<code>\lltxmath_patch:cNnnn</code> 39, 176		<code>\msg_redirect_class:nn</code> 525	
<code>\lltxmath_patch:NNnnn</code> 39, 65, 121, 137, 185, 206		<code>\msg_warning:nnx</code> 56	
<code>\lltxmath_restore_catcode:N</code> 10, 15		<code>\msg_warning:nnxxx</code> 50	
<code>\lltxmath_set_mathchar:NN</code> 86, 102, 103, 118, 119		<code>\MT_cramped_internal:Nn</code> 205	
<code>\lltxmath_temp:w</code> 38, 43, 44, 52		N	
<code>\lua_now:x</code> 88, 95		nath (package) 4	
<code>luatex-required</code> (message) 18		<code>\NeedsTeXFormat</code> 2	
		<code>\NewDocumentCommand</code> 326, 331, 334, 361, 364	
		<code>\newsavebox</code> 412	
		<code>\nulldelimiterspace</code> 212	
		O	
		<code>\over</code> 67, 71	

P		<code>\sum_</code> 433, 436, 455, 458
packages:		T
<code>amsmath</code> 1, 2, 4--6, 14, 16		<code>\tbinom</code> 446, 468
<code>etoolbox</code> 2		<code>\test_assert_cramped:Nx</code> 337, 362, 365
<code>expl3</code> 2		<code>\test_assert_equal:cccccnn</code> 308, 320
<code>filehook</code> 2		<code>\test_assert_equal:nnn</code> 319, 327, 332
<code>luatexbase</code> 2, 3		<code>\test_assert_equal:NNNNnn</code> 308
<code>luatexbase-modutils</code> 9		<code>\test_equal_fail:nxn</code> 297, 315
<code>mathtools</code> 1, 2, 8, 14, 15		<code>\test_equal_pass:nxn</code> 286, 313
<code>nath</code> 4		<code>\test_fail:x</code> 280, 298, 352
<code>unicode-math</code> 1, 2, 16		<code>\test_pass:x</code> 276, 287, 344
<code>xparse</code> 10		<code>\testbox</code> 410, 426, 428, 429
<code>pass (message)</code> 10, 275		<code>\textfont</code> 94, 124, 217, 218, 222
<code>patch-macro (message)</code> 31		<code>\textstyle</code> 221, 381, 399
<code>print_class_fam_slot (function)</code> 10, 264		<code>\tfrac</code> 442, 464, 537
<code>print_fam_slot (function)</code> 10, 258		<code>\the</code> 124
<code>\ProvidesExplPackage</code> 4		<code>\thr@@</code> 145
R		<code>\tl_const:Nn</code> 283
<code>\radical</code> 213		<code>\tl_const:Nx</code> 283, 284, 285
<code>\relax</code> . . . 115, 116, 123, 125, 189, 413, 420		<code>\tl_replace_in:Nnn</code> 114
<code>\RequireLuaModule</code> 9		<code>\token_if_eq_meaning:NNT</code> 167
<code>\RequirePackage</code> 3, 6, 7, 8		<code>\token_if_macro:NTF</code> 41
<code>\resetMathstrut@</code> 121		<code>\token_to_meaning:N</code> 51, 52
<code>\restore@math@cr</code> 141, 157		<code>\token_to_str:N</code> 46, 51, 57
Robertson, Will 1		<code>\tw@</code> 143, 144
S		U
<code>\sbox</code> 208, 426		<code>unicode-math (package)</code> 1, 2, 16
<code>\scan_stop:</code> 91, 195		<code>\use:c</code> 237
<code>\scriptfont</code> 143, 144, 145, 225		<code>\usepackage</code> 273,
<code>\scriptscriptfont</code> 227		370, 411, 414, 419, 527, 528, 530
<code>\scriptstyle</code> 150,		V
160, 161, 163, 170, 224, 385, 403		<code>\vcenter</code> 138, 154
<code>\setbox</code> 122		W
<code>\setmathfont</code> 529		wrong-meaning (message) 25
<code>\skip_set:Nn</code> 159		X
<code>\sqrt</code> . . . 373, 376, 391, 394, 432, 454, 533		<code>xparse (package)</code> 10
<code>\std@equal</code> 108, 116, 119, 424		Z
<code>\std@equals</code> 107		<code>\z@</code> 122,
<code>\std@minus</code> 107, 115, 118, 417		127, 128, 208, 212, 213, 232, 233, 234
<code>\subarray</code> 137		
<code>subarray (environment)</code> 137		
<code>\substack</code> 434, 456		