

# The `lualatex-math` package\*

Philipp Stephani  
st\_philipp@yahoo.de

2011/09/18

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Interface</b>	<b>2</b>
<b>3</b>	<b>Implementation of the <math>\LaTeX 2_\epsilon</math> package</b>	<b>2</b>
3.1	Requirements	2
3.2	Messages	3
3.3	Initialization	4
3.4	Patching	4
3.5	$\LaTeX 2_\epsilon$ kernel	5
3.6	<code>amsmath</code>	6
3.7	<code>mathtools</code>	9
3.8	<code>icomma</code>	10
<b>4</b>	<b>Implementation of the <code>Lua<math>\LaTeX</math></code> module</b>	<b>10</b>
<b>5</b>	<b>Test files</b>	<b>11</b>
5.1	Common definitions	11
5.2	$\LaTeX 2_\epsilon$ kernel, allocation of math families	16
5.3	$\LaTeX 2_\epsilon$ kernel, <code>\mathstyle</code> primitive	16
5.4	<code>amsmath</code> and <code>mathtools</code>	17
5.5	<code>unicode-math</code>	20
5.6	<code>icomma</code> without <code>unicode-math</code>	20
5.7	<code>icomma</code> with <code>unicode-math</code>	20

## 1 Introduction

`Lua $\TeX$`  brings major improvements to all areas of  $\TeX$  typesetting and programming. They are made available through new primitives or the embedded Lua interpreter, and combining them with existing  $\LaTeX 2_\epsilon$  packages is not a task the average  $\LaTeX$  user should have to care about. Therefore a multitude of  $\LaTeX 2_\epsilon$  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

---

\*This document corresponds to `lualatex-math` v0.3b, dated 2011/09/18.

in this package. While in principle most  $\TeX$  documents written for traditional engines should work just fine with Lua $\TeX$ , 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 Lua $\TeX$ .

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  $\TeX$ . To keep the negative impact as small as possible, the `lualatex-math` package patches only the  $\TeX_{2\epsilon}$  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 Lua $\TeX$ . By its very nature, the package is likely to cause problems; in such cases, please refer to the issue tracker<sup>1</sup>.

## 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 Lua $\TeX$  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  $\TeX_{2\epsilon}$  kernel and the `amsmath`, `mathtools` and `icomma` 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  $\TeX$ .

`\mathstyle, \luatexmathstyle`  
`\frac, \binom, \genfrac`

One user-visible change is that the new `\mathstyle` primitive (usually called `\luatexmathstyle` in Lua $\TeX$ ) 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.

## 3 Implementation of the $\TeX_{2\epsilon}$ package

### 3.1 Requirements

```

1 (*package)
2 \NeedsTeXFormat{LaTeX2e}[2009/09/24]
3 \RequirePackage{expl3}[2011/09/05]
4 \ProvidesExplPackage{lualatex-math}{2011/09/18}{0.3b}%
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 ]

```

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

```

9 \RequireLuaModule { luatex-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 } }
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   \lltxmath_restore_catcode:N \$
16 }
17 \char_set_catcode_math_toggle: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 }

```

different-meanings Issued when two control sequences have different meanings, but should not.

```

22 \msg_new:nnnn { luatex-math } { different-meanings } {
23   I've~ expected~ the~ control~ sequences \\
24   #1~ and~ #3 \\
25   to~ have~ the~ same~ meaning,~ but~ their~ meanings~ are~ different.
26 } {
27   The~ meaning~ of~ #1~ is: \\
28   #2 \\
29   The~ meaning~ of~ #3~ is: \\
30   #4
31 }

```

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

```

32 \msg_new:nnn { luatex-math } { macro-expected } {
33   I've~ expected~ that~ #1~ is~ a~ macro,~ but~ it~ isn't.
34 }

```

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.

```

35 \msg_new:nnn { luatex-math } { wrong-meaning } {
36   I've~ expected~ #1~ to~ have~ the~ meaning \\
37   #3, \\
38   but~ it~ has~ the~ meaning \\
39   #2.
40 }

```

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

```

41 \msg_new:nnn { lualatex-math } { patch-macro } {
42   I'm~ going~ to~ patch~ macro~ #1.
43 }

```

### 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.

```

44 \luatex_if_engine:F {
45   \msg_error:nn { lualatex-math } { luatex-required }
46   \endinput
47 }

```

### 3.4 Patching

```

\ltxmath_temp:w A scratch macro.
48 \cs_new_eq:NN \ltxmath_temp:w \prg_do_nothing:

\luatexUmathcode We need the extended versions of \mathcode and \mathchardef. The command
\luatexUmathcodenum \luatexbase@ensure@primitive{<name>} makes sure that the LuaTeX primitive
\luatexUmathchardef \<name> is available under the qualified name \luatex<name>.
49 \luatexbase@ensure@primitive { Umathcode }
50 \luatexbase@ensure@primitive { Umathcodenum }
51 \luatexbase@ensure@primitive { Umathchardef }

\ltxmath_assert_eq:NN The macro \ltxmath_assert_eq:NN<first command><second command> tests
whether the control sequences <first command> and <second command> have the
same meaning, and prints an error message if they do not.
52 \cs_new_protected_nopar:Npn \ltxmath_assert_eq:NN #1 #2 {
53   \cs_if_eq:NNF #1 #2 {
54     \msg_error:nnxxx { lualatex-math } { different-meanings }
55     { \token_to_str:N #1 } { \token_to_meaning:N #1 }
56     { \token_to_str:N #2 } { \token_to_meaning:N #2 }
57   }
58 }

\ltxmath_patch:NNnnn The auxiliary macro \ltxmath_patch:NNnnn<command><factory command>{<pa-
\ltxmath_patch:cNnnn rameter 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.
59 \cs_new_protected_nopar:Npn \ltxmath_patch:NNnnn #1 #2 #3 #4 #5 {
60   \cs_if_exist:NT #1 {
61     \token_if_macro:NTF #1 {
62       \group_begin:
63       #2 \ltxmath_temp:w #3 { #4 }
64       \cs_if_eq:NNTF #1 \ltxmath_temp:w {
65         \msg_info:nxx { lualatex-math } { patch-macro }
66         { \token_to_str:N #1 }
67       \group_end:
68       #2 #1 #3 { #5 }
69     } {
70       \msg_warning:nnxxx { lualatex-math } { wrong-meaning }

```

```

71         { \token_to_str:N #1 } { \token_to_meaning:N #1 }
72         { \token_to_meaning:N \lltxmath_temp:w }
73     \group_end:
74 }
75 } {
76     \msg_warning:nxx { lualatex-math } { macro-expected }
77     { \token_to_str:N #1 }
78 }
79 }
80 }
81 \cs_generate_variant:Nn \lltxmath_patch:NNnnn { c }

```

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

```

82 \cs_new_protected_nopar:Npn \lltxmath_set_mathchar:NN #1 #2 {
83     \luatexUmathchardef #1
84     \lua_now:x {
85         lualatex.math.print_class_fam_slot( \int_eval:n { `#2 } )
86     }
87     \scan_stop:
88 }

```

### 3.5 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> kernel

In LuaT<sub>E</sub>X, we have 256 math families at our disposal. Therefore we modify the L<sup>A</sup>T<sub>E</sub>X allocation macros `\newfam` and `\new@mathgroup` accordingly.

First we test whether `\newfam` and `\new@mathgroup` are equal.

```

89 \lltxmath_assert_eq:NN \newfam \new@mathgroup

```

`\new@mathgroup` It is enough to modify the maximum number of families known to the allocation system; the macro `\alloc@` takes care of the rest. This would work even if the `etex` package weren't loaded.

```

90 \lltxmath_patch:NNnnn \new@mathgroup \cs_set_nopar:Npn { } {
91     \alloc@ 8 \mathgroup \chardef \sixt@@n
92 } {
93     \alloc@ 8 \mathgroup \chardef \c_two_hundred_fifty_six
94 }

```

`\newfam` We have to reset `\newfam` to equal `\new@mathgroup`.

```

95 \cs_set_eq:NN \newfam \new@mathgroup

```

LuaT<sub>E</sub>X enables access to the current mathematical style via the `\mathstyle` primitive. For this to work, fraction-like constructs (e.g.,  $\langle numerator\rangle\over\langle denominator\rangle$ ) have to be enclosed in a `\Ustack` group. `\frac` can be patched to do this, but the plain T<sub>E</sub>X remnants `\choose`, `\brack` and `\brace` should be discouraged.

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

```

96 \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  $\LaTeX 2_\epsilon$  kernel definition if the `amsmath` package is not loaded; the corresponding patch for `amsmath` follows below.

```

97 \AtEndPreamble {
98   \ifpackageloaded { amsmath } { } {
99     \lltxmath_patch:NNnnn \frac \cs_set_nopar:Npn { #1 #2 } {
100       {
101         \begingroup #1 \endgroup \over #2
102       }
103     } {

```

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

```

104     {
105       \luatexUstack { \group_begin: #1 \group_end: \over #2 }
106     }
107   }
108 }
109 }

```

### 3.6 amsmath

The popular `amsmath` package is subject to three  $\text{Lua}\TeX$ -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.

```

\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 110 \luatexbase@ensure@primitive { alignmark }
                  111 \luatexbase@ensure@primitive { Ustartmath }
                  112 \luatexbase@ensure@primitive { Ustopmath }

```

```

\luatexUmathstacknumup Now we require the font parameters we will use.
\luatexUmathstackdenomdown 113 \luatexbase@ensure@primitive { Umathstacknumup }
\luatexUmathstackvgap 114 \luatexbase@ensure@primitive { Umathstackdenomdown }
115 \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.
116 \int_const:Nn \c_lltxmath_std_minus_mathcode_int { "2200 }
117 \int_const:Nn \c_lltxmath_std_equal_mathcode_int { "303D }

```

```

\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.
118 \cs_new_nopar:Npn \lltxmath_char_dim:NN #1 #2 {
119   #1 \textfont
120   \lua_now:x {
121     lua $\text{lua}$ l $\text{lua}$ tex.math.print_fam_slot( \int_eval:n { `#2 } )
122   }
123 }

```

`\l_1ltxmath_minus_mathchar` `\l_1ltxmath_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`.

```
124 \chk_if_free_cs:N \l_1ltxmath_minus_mathchar
125 \chk_if_free_cs:N \l_1ltxmath_equal_mathchar
126 \AtBeginOfPackageFile { amsmath } {
127   \ltxmath_set_mathchar:NN \l_1ltxmath_minus_mathchar \-
128   \ltxmath_set_mathchar:NN \l_1ltxmath_equal_mathchar \=
```

Now we temporarily reset the mathematical codes.

```
129 \char_set_mathcode:nn { \- } { \c_1ltxmath_std_minus_mathcode_int }
130 \char_set_mathcode:nn { \= } { \c_1ltxmath_std_equal_mathcode_int }
131 \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.

```
132   \cs_set_eq:NN \std@minus \l_1ltxmath_minus_mathchar
133   \cs_set_eq:NN \std@equal \l_1ltxmath_equal_mathchar
```

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

```
134   \luatexUmathcodenum \- \l_1ltxmath_minus_mathchar
135   \luatexUmathcodenum \= \l_1ltxmath_equal_mathchar
136 }
137 }
```

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.

```
138 \AtEndOfPackageFile * { amsmath } {
139   \tl_replace_once:Nnn \@begindocumenthook {
140     \mathchardef \std@minus \mathcode \- \relax
141     \mathchardef \std@equal \mathcode \= \relax
142   } {
143     \ltxmath_set_mathchar:NN \std@minus \-
144     \ltxmath_set_mathchar:NN \std@equal \=
145   }
```

`\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  $\epsilon$ - $\TeX$  allows us to query glyph metrics directly.

```
146   \ltxmath_patch:NNnnn \resetMathstrut@ \cs_set_nopar:Npn { } {
147     \setbox \z@ \hbox {
148       \mathchardef \@tempa \mathcode \(\ \relax % \)
149       \def \@tempb ##1 "##2 ##3 { \the \textfont "##3 \char" }
150       \expandafter \@tempb \meaning \@tempa \relax
151     }
```

```

152   \ht \Mathstrutbox@ \ht \z@
153   \dp \Mathstrutbox@ \dp \z@
154 } {
155   \box_set_ht:Nn \Mathstrutbox@ {
156     \lltxmath_char_dim:NN \fontcharht \(\ % \)
157   }
158   \box_set_dp:Nn \Mathstrutbox@ {
159     \lltxmath_char_dim:NN \fontchardp \)
160   }
161 }

```

subarray The `subarray` environment uses legacy font dimensions. We simply patch it to use LuaTeX font parameters (and L<sup>A</sup>T<sub>E</sub>X<sub>3</sub> expressions instead of T<sub>E</sub>X 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`).

```

162 \lltxmath_patch:NNnnn \subarray \cs_set:Npn { #1 } {
163   \vcenter
164   \bgroup
165   \Let@
166   \restore@math@cr
167   \default@tag
168   \baselineskip \fontdimen 10~ \scriptfont \tw@
169   \advance \baselineskip \fontdimen 12~ \scriptfont \tw@
170   \lineskip \thr@@ \fontdimen 8~ \scriptfont \thr@@
171   \lineskiplimit \lineskip
172   \ialign
173   \bgroup
174   \ifx c #1 \hfil \fi
175   $ \m@th \scriptstyle ## $
176   \hfil
177   \crrc
178 } {
179   \vcenter
180   \c_group_begin_token
181   \Let@
182   \restore@math@cr
183   \default@tag
184   \skip_set:Nn \baselineskip {
185     \luatexUmathstacknumup \scriptstyle
186     + \luatexUmathstackdenomdown \scriptstyle
187   }
188   \lineskip \luatexUmathstackvgap \scriptstyle
189   \lineskiplimit \lineskip
190   \ialign
191   \c_group_begin_token
192   \token_if_eq_meaning:NNT c #1 { \hfil }
193   \luatexUstartmath
194   \m@th
195   \scriptstyle
196   \luatexalignmark \luatexalignmark
197   \luatexUstopmath
198   \hfil
199   \crrc
200 }

```

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



```

201 \ltxmath_patch:cNnnn { frac~ } \cs_set:Npn { #1 #2 } {
202   {
203     \begingroup #1 \endgroup \@@over #2
204   }
205 } {
206   {
207     \luatexUstack { \group_begin: #1 \group_end: \@@over #2 }
208   }
209 }

```

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

```

210 \ltxmath_patch:NNnnn \@genfrac \cs_set_nopar:Npn {
211   #1 #2 #3 #4 #5
212 } {
213   {
214     #1 { \begingroup #4 \endgroup #2 #3 \relax #5 }
215   }
216 } {
217   {
218     #1 {
219       \luatexUstack {
220         \group_begin: #4 \group_end: #2 #3 \scan_stop: #5
221       }
222     }
223   }
224 }
225 }

```

### 3.7 mathtools

`mathtools`' `\cramped` command and others that make use of its internal version use a hack involving a null radical. Lua<sub>T</sub><sub>E</sub>X 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 226 \luatexbase@ensure@primitive { crampeddisplaystyle }
\luatexcrampedscriptstyle 227 \luatexbase@ensure@primitive { crampedtextstyle }
\luatexcrampedscriptscriptstyle 228 \luatexbase@ensure@primitive { crampedscriptstyle }
229 \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 Lua<sub>T</sub><sub>E</sub>X 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.

```

230 \AtEndOfPackageFile * { mathtools } {
231   \ltxmath_patch:NNnnn \MT_cramped_internal:Nn
232   \cs_set_nopar:Npn { #1 #2 } {
233     \sbox \z@ {
234       $
235       \m@th
236       #1
237       \nulldelimiterspace = \z@
238       \radical \z@ { #2 }
239       $
240     }
241     \ifx #1 \displaystyle

```

```

242     \dimen@ = \fontdimen 8 \textfont 3
243     \advance \dimen@ .25 \fontdimen 5 \textfont 2
244   \else
245     \dimen@ = 1.25 \fontdimen 8
246     \ifx #1 \textstyle
247       \textfont
248     \else
249       \ifx #1 \scriptstyle
250         \scriptfont
251       \else
252         \scriptscriptfont
253       \fi
254     \fi
255     3
256   \fi
257   \advance \dimen@ -\ht\z@
258   \ht\z@ = -\dimen@
259   \box\z@
260 } {

```

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

```

261   {
262     \use:c { luatexcramped \cs_to_str:N #1 } #2
263   }
264 }
265 }

```

### 3.8 icomma

The `icomma` package uses `\mathchardef` to save the mathematical code of the comma character. This breaks for Unicode fonts. The incompatibility was noticed by Peter Breitfeld.<sup>2</sup>

`\mathcomma` `icomma` defines the mathematical character shorthand `\icomma` at the beginning of the document, therefore we again patch `\@begindocumenthook`.

```

266 \AtEndOfPackageFile * { icomma } {
267   \tl_replace_once:Nnn \@begindocumenthook {
268     \mathchardef \mathcomma \mathcode `\",
269   } {
270     \lltxmath_set_mathchar:NN \mathcomma \,
271   }
272 }
273 </package>

```

## 4 Implementation of the Lua<sup>L</sup>A<sup>T</sup>E<sup>X</sup> module

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

```

274 <lua>
275 require("luatexbase.modutils")
276 require("luatexbase.cctb")
277 local err, warn, info, log = luatexbase.provides_module({
278   name = "lualatex-math",

```

<sup>2</sup><https://groups.google.com/d/topic/de.comp.text.tex/Cputk-AJS5I/discussion>

```

279 date = "2011/05/05",
280 version = 0.1,
281 description = "Patches for mathematics typesetting with LuaLaTeX",
282 author = "Philipp Stephani",
283 licence = "LPPL v1.3+"
284 })
285 local unpack = unpack
286 local string = string
287 local tex = tex
288 local cctb = luatexbase.catcodetables
289 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`.

```

290 function print_fam_slot(char)
291   local code = tex.getmathcode(char)
292   local class, family, slot = unpack(code)
293   local result = string.format("%i %i ", family, slot)
294   tex.sprint(cctb.string, result)
295 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`.

```

296 function print_class_fam_slot(char)
297   local code = tex.getmathcode(char)
298   local class, family, slot = unpack(code)
299   local result = string.format("%i %i %i ", class, family, slot)
300   tex.sprint(cctb.string, result)
301 end
302 </lua>

```

## 5 Test files

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

### 5.1 Common definitions

```

303 <*test>
304 \documentclass[pagesize=auto]{scrartcl}

Only xparse starting with 2008/08/03 has \NewDocumentCommand.
305 \usepackage{xparse}[2008/08/03]
306 \usepackage{luacode}
307 \ExplSyntaxOn
308 \AtBeginDocument { \errorcontextlines = \c_fifteen }

```

`pass` This message is issued when a test passed.

```

309 \msg_new:nnn { test } { pass } { #1 }

```

`\test_pass:x` The macro `\test_pass:x{<text>}` issues the `pass` message with description  $\langle text \rangle$ .

```

310 \cs_new_protected_nopar:Npn \test_pass:x #1 {
311   \msg_info:nnx { test } { pass } { #1 }
312 }

```

```

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

\test_fail:x The macro \test_fail:x{<text>} issues the fail message with description <text>.
314 \cs_new_protected_nopar:Npn \test_fail:x #1 {
315   \msg_error:nxx { test } { fail } { #1 }
316 }

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

\c_test_equal_tl Two shorthands for pretty-printing test results.
\c_test_not_equal_tl 318 \tl_const:Nx \c_test_equal_tl { \c_space_tl == \c_space_tl }
319 \tl_const:Nx \c_test_not_equal_tl { \c_space_tl != \c_space_tl }

\test_equal_pass:nxx The macro \test_equal_pass:nxx{<first expression>}{<first value>}{<second ex-
pression>}{<second value>} is called when the two values arising from the two
expressions are equal.
320 \cs_new_protected_nopar:Npn \test_equal_pass:nxx #1 #2 #3 #4 {
321   \test_pass:x {
322     \exp_not:n { #1 }
323     \c_test_equal_tl
324     #2
325     \c_test_equal_tl
326     #4
327     \c_test_equal_tl
328     \exp_not:n { #3 }
329   }
330 }

\test_equal_fail:nxx The macro \test_equal_pass:nxx{<first expression>}{<first value>}{<second ex-
pression>}{<second value>} is called when the two values arising from the two
expressions are not equal.
331 \cs_new_protected_nopar:Npn \test_equal_fail:nxx #1 #2 #3 #4 {
332   \test_fail:x {
333     \exp_not:n { #1 }
334     \c_test_equal_tl
335     #2
336     \c_test_not_equal_tl
337     #4
338     \c_test_equal_tl
339     \exp_not:n { #3 }
340   }
341 }

\test_assert_equal:NNNNNnn The macro \test_assert_equal:NNNNNnn<set command><use command><compare
command><first temporary command><second temporary command>{<first expres-
sion>}{<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.
342 \cs_new_protected_nopar:Npn
343 \test_assert_equal:NNNNNnn #1 #2 #3 #4 #5 #6 #7 {
344   #1 #4 { #6 }
345   #1 #5 { #7 }
346   #3 { #4 } = { #5 } {
347     \test_equal_pass:nxx { #6 } { #2 #4 } { #7 } { #2 #5 }

```

```

348 } {
349   \test_equal_fail:nxn { #6 } { #2 #4 } { #7 } { #2 #5 }
350 }
351 }
352 \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<sup>A</sup>T<sub>E</sub>X3 naming conventions; `⟨data type⟩` must be `int`, `dim`, etc.

```

353 \cs_new_protected_nopar:Npn \test_assert_equal:nnn #1 #2 #3 {
354   \test_assert_equal:ccccnn
355   { #1 _set:Nn } { #1 _use:N } { #1 _compare:nNnTF }
356   { l_test_tmpa_ #1 } { l_test_tmpb_ #1 } { #2 } { #3 }
357 }

```

`\l_test_tmpa_int` Scratch registers for numbers.

`\l_test_tmpb_int`

```

358 \int_new:N \l_test_tmpa_int
359 \int_new:N \l_test_tmpb_int

```

`\AssertIntEqual` The command `\AssertIntEqual{⟨first expression⟩}{⟨second expression⟩}` asserts that the two integral expressions are equal.

```

360 \NewDocumentCommand \AssertIntEqual { m m } {
361   \test_assert_equal:nnn { int } { #1 } { #2 }
362 }

```

`\l_test_tmpa_int` Scratch registers for dimensions.

`\l_test_tmpb_int`

```

363 \dim_new:N \l_test_tmpa_dim
364 \dim_new:N \l_test_tmpb_dim

```

`\AssertDimEqual` The command `\AssertDimEqual{⟨first expression⟩}{⟨second expression⟩}` asserts that the two dimension expressions are equal.

```

365 \NewDocumentCommand \AssertDimEqual { m m } {
366   \test_assert_equal:nnn { dim } { #1 } { #2 }
367 }

```

`\AssertMathStyle` The command `\AssertMathStyle{⟨expression⟩}` asserts that the current mathematical style is equal to the value of the integral `⟨expression⟩`.

```

368 \NewDocumentCommand \AssertMathStyle { m } {
369   \AssertIntEqual { \luatexmathstyle } { #1 }
370 }

```

`\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`.

```

371 \cs_new_protected_nopar:Npn \test_assert_cramped:Nx #1 #2 {
372   \int_set:Nn \l_test_tmpa_int { \luatexmathstyle }
373   \bool_if:nTF {
374     \int_compare_p:nNn { \l_test_tmpa_int } > { \c_minus_one }
375     &&
376     #1 { \l_test_tmpa_int }
377   } {
378     \test_pass:x {
379       \exp_not:N \luatexmathstyle
380       \c_test_equal_tl
381       \int_use:N \l_test_tmpa_int
382       \c_space_tl
383       is~ a~ #2~ style

```

```

384   }
385 } {
386   \test_fail:x {
387     \exp_not:N \luatexmathstyle
388     \c_test_equal_tl
389     \int_use:N \l_test_tmpa_int
390     \c_space_tl
391     is~ not~ a~ #2~ style
392   }
393 }
394 }

```

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

```

395 \NewDocumentCommand \AssertNoncrampedStyle { } {
396   \test_assert_cramped:Nx \int_if_even_p:n { non-cramped }
397 }

```

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

```

398 \NewDocumentCommand \AssertCrampedStyle { } {
399   \test_assert_cramped:Nx \int_if_odd_p:n { cramped }
400 }

```

`\l_test_tmpa_box` Scratch registers for box constructions.

`\l_test_tmpb_box`

```

401 \box_new:N \l_test_tmpa_box
402 \box_new:N \l_test_tmpb_box

```

`contains_space` The function `contains_space(head, width)` returns `true` if the node list starting at `head` or any of its sublists contain a glue or kern node of width `width` (or any glue or kern node if `width` is `nil`).

```

403 \begin{luacode*}
404 function contains_space(head, width)
405   for n in node.traverse(head) do
406     local id = n.id
407     if id == 10 or id == 11 then
408       if width then
409         if (id == 10 and n.spec.width == width)
410            or (id == 11 and n.kern == width) then
411           return true
412         end
413       else
414         return true
415       end
416     elseif id == 0 or id == 1 then
417       if contains_space(n.head, width) then
418         return true
419       end
420     end
421   end
422   return false
423 end
424 \end{luacode*}

```

`\AssertNoSpace` The command `\AssertNoSpace{text}` asserts that the node list that is the result of typesetting `text` contains no glue or kern nodes.

```

425 \NewDocumentCommand \AssertNoSpace { m } {

```

```

426 \hbox_set:Nn \l_test_tmpa_box { #1 }
427 \int_if_odd:nTF {
428   \lua_now:x {
429     local~ b = tex.getbox(\int_use:N \l_test_tmpa_box)
430     if~ contains_space(b.head) then~
431       tex.sprint("0")
432     else~
433       tex.sprint("1")
434     end
435   }
436 } {
437   \test_pass:x {
438     \tl_to_str:n { #1 } ~
439     contains~ no~ skip~ or~ kern~ node
440   }
441 } {
442   \test_fail:x {
443     \tl_to_str:n { #1 } ~
444     contains~ a~ skip~ or~ kern~ node
445   }
446 }
447 }

```

`\AssertMuSpace` The command `\AssertMuSpace{<text>}{<muskip>}` asserts that the node list that is the result of typesetting `<text>` contains at least one glue or kern node of with `<muskip>`.

```

448 \makeatletter
449 \NewDocumentCommand \AssertMuSpace { m m } {
450   \hbox_set:Nn \l_test_tmpa_box { #1 }
451   \hbox_set:Nn \l_test_tmpb_box { $ \mskip #2 \m@th $ }
452   \int_if_odd:nTF {
453     \lua_now:x {
454       local~ b = tex.getbox(\int_use:N \l_test_tmpa_box)
455       local~ s = tex.getbox(\int_use:N \l_test_tmpb_box)
456       if~ contains_space(b.head, s.width) then~
457         tex.sprint("1")
458       else~
459         tex.sprint("0")
460       end
461     }
462   } {
463     \test_pass:x {
464       \tl_to_str:n { #1 } ~
465       contains~ a~ skip~ or~ kern~ node~ of~ width~
466       \tl_to_str:n { #2 }
467     }
468   } {
469     \test_fail:x {
470       \tl_to_str:n { #1 } ~
471       contains~ no~ skip~ or~ kern~ node~ of~ width~
472       \tl_to_str:n { #2 }
473     }
474   }
475 }
476 \makeatother
477 \ExplSyntaxOff
478 </test>

```

## 5.2 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> kernel, allocation of math families

The L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> kernel itself allocates four families (also known as “math groups” in L<sup>A</sup>T<sub>E</sub>X parlance). Therefore we should still be able to allocate 252 families. We do this alternately with `\newfam`, `\new@mathgroup` and `\DeclareSymbolFont`.

```
479 (*test-kernel-alloc)
480 \usepackage{lualatex-math}
481 \makeatletter
482 \ExplSyntaxOn
483 \prg_stepwise_inline:nnnn { \c_four } { \c_one } {
484   \c_two_hundred_fifty_five - \c_one
485 } {
486   \prg_case_int:nnn { \int_mod:nn { #1 } { \c_three } } {
487     { \c_zero } {
488       \chk_if_free_cs:N \g_test_family_int
489       \newfam \g_test_family_int
490       \AssertIntEqual { \g_test_family_int } { #1 }
491       \cs_undefine:N \g_test_family_int
492     }
493     { \c_one } {
494       \chk_if_free_cs:N \g_test_mathgroup_int
495       \new@mathgroup \g_test_mathgroup_int
496       \AssertIntEqual { \g_test_mathgroup_int } { #1 }
497       \cs_undefine:N \g_test_mathgroup_int
498     }
499     { \c_two } {
500       \DeclareSymbolFont { Test #1 } { OT1 } { cmr } { m } { n }
501       \exp_args:Nc \AssertIntEqual { sym Test #1 } { #1 }
502     }
503   } {
504     \test_fail:x { This~ cannot~ happen }
505   }
506 }
507 \DeclareSymbolFont { Test 255 } { OT1 } { cmr } { bx } { it }
508 \DeclareSymbolFontAlphabet { \TestAlphabet } { Test 255 }
509 \exp_args:Nc \AssertIntEqual { sym Test 255 }
510 { \c_two_hundred_fifty_five }
511 \ExplSyntaxOff
512 \makeatother
513 \begin{document}
514 \[
515 \TestAlphabet{
516   abc
517   \AssertIntEqual{\fam}{255}
518   \AssertIntEqual{\mathgroup}{255}
519 }
520 \]
521 \end{document}
522 </test-kernel-alloc)
```

## 5.3 L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> kernel, `\mathstyle` primitive

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

```
523 (*test-kernel-style)
524 \usepackage{lualatex-math}
525 \begin{document}
526 \begin{displaymath}
```



```

527 \AssertMathStyle{0} \sqrt{\AssertMathStyle{1}}
528 \frac{\AssertMathStyle{2}}{\AssertMathStyle{3}}
529 a^{\frac{\AssertMathStyle{6}}{\AssertMathStyle{7}}}
530 \sqrt{\frac{\AssertMathStyle{3}}{\AssertMathStyle{3}}}
531 \displaystyle
532 \frac{\AssertMathStyle{2}}{\AssertMathStyle{3}}
533 \luatexcrampeddisplaystyle
534 \frac{\AssertMathStyle{3}}{\AssertMathStyle{3}}
535 \textstyle
536 \frac{\AssertMathStyle{4}}{\AssertMathStyle{5}}
537 \luatexcrampedtextstyle
538 \frac{\AssertMathStyle{5}}{\AssertMathStyle{5}}
539 \scriptstyle
540 \frac{\AssertMathStyle{6}}{\AssertMathStyle{7}}
541 \luatexcrampedscriptstyle
542 \frac{\AssertMathStyle{7}}{\AssertMathStyle{7}}
543 \end{displaymath}
544 \begin{math}
545 \AssertMathStyle{2} \sqrt{\AssertMathStyle{3}}
546 \frac{\AssertMathStyle{4}}{\AssertMathStyle{5}}
547 a^{\frac{\AssertMathStyle{6}}{\AssertMathStyle{7}}}
548 \sqrt{\frac{\AssertMathStyle{5}}{\AssertMathStyle{5}}}
549 \displaystyle
550 \frac{\AssertMathStyle{2}}{\AssertMathStyle{3}}
551 \luatexcrampeddisplaystyle
552 \frac{\AssertMathStyle{3}}{\AssertMathStyle{3}}
553 \textstyle
554 \frac{\AssertMathStyle{4}}{\AssertMathStyle{5}}
555 \luatexcrampedtextstyle
556 \frac{\AssertMathStyle{5}}{\AssertMathStyle{5}}
557 \scriptstyle
558 \frac{\AssertMathStyle{6}}{\AssertMathStyle{7}}
559 \luatexcrampedscriptstyle
560 \frac{\AssertMathStyle{7}}{\AssertMathStyle{7}}
561 \end{math}
562 \end{document}
563  $\end{test-kernel-style}$ 

```

## 5.4 amsmath and mathtools

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

$\text{\testbox}$  First a scratch box register.

```

564  $\text{*test-amsmath}$ 
565 \usepackage{lualatex-math}
566 \newsavebox{\testbox}

```

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

```

567 \luatexUmathcode`\-="2 "33 "44444 \relax
568 \usepackage{amsmath}
569 \AssertIntEqual{\luatexUmathcode`\-}{"33444444}
570 \makeatletter
571 \AssertIntEqual{\std@minus}{"33444444}
572 \makeatother
573 \usepackage{mathtools}

```

The same for the document begin hook.

```

574 \luatexUmathcode`\="5 "66 "77777 \relax
575 \begin{document}
576 \AssertIntEqual{\luatexUmathcode`\=}{"66A77777}
577 \makeatletter
578 \AssertIntEqual{\std@equal}{"66A77777}
579 \makeatother

```

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

```

580 \sbox{\testbox}{$(\$) \% )}
581 \makeatletter
582 \AssertDimEqual{\ht\Mathstrutbox@}{\ht\testbox}
583 \AssertDimEqual{\dp\Mathstrutbox@}{\dp\testbox}
584 \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.

```

585 \begin{equation*}
586 \AssertMathStyle{0} \sqrt{\AssertMathStyle{1}}
587 \sum_{
588   \substack{\frac{1}{2} \ \ \ \frac{3}{4} \ \ \ \frac{5}{6}}
589 }
590 \sum_{
591   \begin{subarray}{l} \frac{1}{2} \ \ \ \frac{3}{4} \ \ \ \frac{5}{6} \end{subarray}}
592 }
593 \frac{\AssertMathStyle{2}}{\AssertMathStyle{3}}
594 a^{\frac{\AssertMathStyle{6}}{\AssertMathStyle{7}}}
595 \dfrac{\AssertMathStyle{2}}{\AssertMathStyle{3}}
596 \tfrac{\AssertMathStyle{4}}{\AssertMathStyle{5}}
597 \binom{\AssertMathStyle{2}}{\AssertMathStyle{3}}
598 a^{\binom{\AssertMathStyle{6}}{\AssertMathStyle{7}}}
599 \dbinom{\AssertMathStyle{2}}{\AssertMathStyle{3}}
600 \tbinom{\AssertMathStyle{4}}{\AssertMathStyle{5}}
601 \genfrac{}{}{}{\AssertMathStyle{2}}{\AssertMathStyle{3}}
602 \genfrac{<}{/}{Opt}{0}{\AssertMathStyle{2}}{\AssertMathStyle{3}}
603 \genfrac{}{}{}{1}{\AssertMathStyle{4}}{\AssertMathStyle{5}}
604 \genfrac{|}{|}{4pt}{2}{\AssertMathStyle{6}}{\AssertMathStyle{7}}
605 \genfrac{}{}{}{3}{\AssertMathStyle{6}}{\AssertMathStyle{7}}
606 \end{equation*}
607 \begin{math}
608 \AssertMathStyle{2} \sqrt{\AssertMathStyle{3}}
609 \sum_{
610   \substack{\frac{1}{2} \ \ \ \frac{3}{4} \ \ \ \frac{5}{6}}
611 }
612 \sum_{
613   \begin{subarray}{l} \frac{1}{2} \ \ \ \frac{3}{4} \ \ \ \frac{5}{6} \end{subarray}}
614 }
615 \frac{\AssertMathStyle{4}}{\AssertMathStyle{5}}
616 a^{\frac{\AssertMathStyle{6}}{\AssertMathStyle{7}}}
617 \dfrac{\AssertMathStyle{2}}{\AssertMathStyle{3}}
618 \tfrac{\AssertMathStyle{4}}{\AssertMathStyle{5}}
619 \binom{\AssertMathStyle{4}}{\AssertMathStyle{5}}
620 a^{\binom{\AssertMathStyle{6}}{\AssertMathStyle{7}}}
621 \dbinom{\AssertMathStyle{2}}{\AssertMathStyle{3}}
622 \tbinom{\AssertMathStyle{4}}{\AssertMathStyle{5}}
623 \genfrac{}{}{}{\AssertMathStyle{4}}{\AssertMathStyle{5}}
624 \genfrac{<}{/}{Opt}{0}{\AssertMathStyle{2}}{\AssertMathStyle{3}}
625 \genfrac{}{}{}{1}{\AssertMathStyle{4}}{\AssertMathStyle{5}}

```

```

626 \genfrac{}{}{4pt}{2}{\AssertMathStyle{6}}{\AssertMathStyle{7}}
627 \genfrac{}{}{3}{\AssertMathStyle{6}}{\AssertMathStyle{7}}
628 \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.

```

629 \begin{equation*}
630 \AssertMathStyle{0}
631 a^{\AssertMathStyle{4} a}
632 \cramped{\AssertCrampedStyle a^{\AssertCrampedStyle a}}
633 a^{
634 \AssertMathStyle{4}
635 a^a
636 \cramped{\AssertCrampedStyle a^{\AssertCrampedStyle a}}
637 a^a
638 \AssertMathStyle{4}
639 }
640 a^{
641 a^{
642 \AssertMathStyle{6}
643 a^a
644 \cramped{\AssertCrampedStyle a^{\AssertCrampedStyle a}}
645 a^a
646 \AssertMathStyle{6}
647 }
648 }
649 a^{\AssertMathStyle{4} a}
650 \AssertMathStyle{0}
651 \end{equation*}
652 \begin{math}
653 \AssertMathStyle{2}
654 a^{\AssertMathStyle{4} a}
655 \cramped{\AssertCrampedStyle a^{\AssertCrampedStyle a}}
656 a^{
657 \AssertMathStyle{4}
658 a^a
659 \cramped{\AssertCrampedStyle a^{\AssertCrampedStyle a}}
660 a^a
661 \AssertMathStyle{4}
662 }
663 a^{
664 a^{
665 \AssertMathStyle{6}
666 a^a
667 \cramped{\AssertCrampedStyle a^{\AssertCrampedStyle a}}
668 a^a
669 \AssertMathStyle{6}
670 }
671 }
672 a^{\AssertMathStyle{4} a}
673 \AssertMathStyle{2}
674 \end{math}
675 \end{document}
676 </test-amsmath>

```

## 5.5 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.

```
677 (*test-unicode)
678 \ExplSyntaxOn
679 \msg_redirect_class:nn { warning } { error }
680 \ExplSyntaxOff
681 \usepackage{amsmath}
682 \usepackage{unicode-math}[2011/05/05]
683 \setmathfont{XITS Math}
684 \usepackage{lualatex-math}
685 \begin{document}
686 \begin{equation*}
687   \AssertMathStyle{0} \sqrt{\AssertMathStyle{1}}
688   \frac{\AssertMathStyle{2}}{\AssertMathStyle{3}}
689   a^{\frac{\AssertMathStyle{6}}{\AssertMathStyle{7}}}
690   \dfrac{\AssertMathStyle{2}}{\AssertMathStyle{3}}
691   \tfrac{\AssertMathStyle{4}}{\AssertMathStyle{5}}
692 \end{equation*}
693 \end{document}
694 </test-unicode>
```

## 5.6 icomma without unicode-math

This test file loads only icomma to test whether our patch works for Computer Modern.

```
695 (*test-icomma)
696 \usepackage{lualatex-math}
697 \usepackage{icomma}
698 \begin{document}
699 $1,234 \; (x, y)$
700 \AssertNoSpace{$1,234$}
701 \AssertMuSpace{$(x, y)$}{\thinmuskip}
702 \AssertIntEqual{\mathcomma}{"1C003B}
703 \end{document}
704 </test-icomma>
```

## 5.7 icomma with unicode-math

This test file loads both icomma and unicode-math to test whether they interact well.

```
705 (*test-icomma-unicode)
706 \usepackage{unicode-math}[2011/05/05]
707 \setmathfont{XITS Math}
708 \usepackage{lualatex-math}
709 \usepackage{icomma}
710 \begin{document}
711 $1,234 \; (x, y)$
712 \AssertNoSpace{$1,234$}
713 \AssertMuSpace{$(x, y)$}{\thinmuskip}
714 \AssertIntEqual{\mathcomma}{"0C002C}
715 \end{document}
716 </test-icomma-unicode>
```

# Change History

v0.1	
General: Initial version	1
v0.2	
General: Added patch for the icomma package	10
Added test file for icomma with unicode-math	20
Added test file for icomma without unicode-math	20
v0.3	
General: Added test file for modified family allocation scheme	16
Patched math group allocation to gain access to all families	5
v0.3a	
General: Updated for changes in l3kernel	1
v0.3b	
\@begindocumenthook: Another update for a change in l3kernel	7

# 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

\\$	15, 17
\(	148, 156
\)	148, 156, 159
\,	268, 270
\-	127, 129, 134, 140, 143, 567, 569
\;	699, 711
\=	128, 130, 135, 141, 144, 574, 576
\@@over	203, 207
\@begindocumenthook	<u>138</u> , 267
\@genfrac	<u>210</u>
\@ifpackageloaded	98
\@tempa	148, 150
\@tempb	149, 150
\[	514
\\	19, 23, 24, 27, 28, 29, 36, 37, 38, 588, 591, 610, 613
\]	520

## A

\advance	169, 243, 257
\alloc@	91, 93
<b>amsmath</b> (package)	1, 2, 5--7, 17, 18, 20
\AssertCrampedStyle	<u>398</u> , 632, 636, 644, 655, 659, 667
\AssertDimEqual	<u>365</u> , 582, 583
\AssertIntEqual	<u>360</u> , 369, 490, 496, 501, 509, 517, 518, 569, 571, 576, 578, 702, 714
\AssertMathStyle	<u>368</u> , 527, 528, 529, 530, 532, 534, 536, 538, 540, 542, 545, 546, 547, 548, 550, 552, 554, 556, 558, 560, 586, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602, 603, 604, 605, 608, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624, 625, 626, 627, 630, 631, 634, 638, 642, 646, 649, 650, 653, 654, 657, 661, 665, 669, 672, 673, 687, 688, 689, 690, 691
\AssertMuSpace	<u>448</u> , 701, 713
\AssertNoncrampedStyle	395
\AssertNoSpace	<u>425</u> , 700, 712
\AtBeginDocument	308

<code>\AtBeginOfPackageFile</code>	126
<code>\AtEndOfPackage</code>	14
<code>\AtEndOfPackageFile</code>	131, 138, 230, 266
<code>\AtEndPreamble</code>	97

## B

<code>\baselineskip</code>	168, 169, 184
<code>\begin</code>	403, 513, 525, 526, 544, 575, 585, 591, 607, 613, 629, 652, 685, 686, 698, 710
<code>\begingroup</code>	101, 203, 214
<code>\bgroup</code>	164, 173
<code>\binom</code>	2, 597, 598, 619, 620
<code>\bool_if:nTF</code>	373
<code>\box</code>	259
<code>\box_new:N</code>	401, 402
<code>\box_set_dp:Nn</code>	158
<code>\box_set_ht:Nn</code>	155
Breitfeld, Peter	10

## C

<code>\c_fifteen</code>	308
<code>\c_four</code>	483
<code>\c_group_begin_token</code>	180, 191
<code>\c_lltxmath_std_equal_mathcode_int</code>	116, 130
<code>\c_lltxmath_std_minus_mathcode_int</code>	116, 129
<code>\c_minus_one</code>	374
<code>\c_one</code>	483, 484, 493
<code>\c_space_tl</code>	318, 319, 382, 390
<code>\c_test_equal_tl</code>	318, 323, 325, 327, 334, 338, 380, 388
<code>\c_test_not_equal_tl</code>	318, 336
<code>\c_three</code>	486
<code>\c_two</code>	499
<code>\c_two_hundred_fifty_five</code>	484, 510
<code>\c_two_hundred_fifty_six</code>	93
<code>\c_zero</code>	487
<code>\char</code>	149
<code>\char_set_catcode:mn</code>	11
<code>\char_set_catcode_math_toggle:N</code>	17
<code>\char_set_mathcode:mn</code>	129, 130
<code>\char_value_catcode:n</code>	12
<code>\chardef</code>	91, 93
<code>\chk_if_free_cs:N</code>	124, 125, 488, 494
<code>contains_space (function)</code>	14, 403
<code>\cramped</code>	632, 636, 644, 655, 659, 667
<code>\crrc</code>	177, 199
<code>\cs_generate_variant:Nn</code>	81, 317, 352
<code>\cs_if_eq:NMF</code>	53
<code>\cs_if_eq:NWTF</code>	64
<code>\cs_if_exist:NT</code>	60
<code>\cs_new_eq:MN</code>	48
<code>\cs_new_nopar:Npn</code>	10, 118
<code>\cs_new_protected_nopar:Npn</code>	52, 59, 82, 310, 314, 320, 331, 342, 353, 371
<code>\cs_set:Npn</code>	162, 201
<code>\cs_set_eq:NN</code>	95, 132, 133
<code>\cs_set_nopar:Npn</code>	90, 99, 146, 210, 232
<code>\cs_to_str:N</code>	262
<code>\cs_undefine:N</code>	491, 497

## D

<code>\dbinom</code>	599, 621
----------------------	----------

<code>\DeclareSymbolFont</code>	500, 507
<code>\DeclareSymbolFontAlphabet</code>	508
<code>\def</code>	149
<code>\default@tag</code>	167, 183
<code>\dfrac</code>	595, 617, 690
different-meanings (message)	<a href="#">22</a>
<code>\dim_new:N</code>	363, 364
<code>\dimen@</code>	242, 243, 245, 257, 258
<code>\displaystyle</code>	241, 531, 549
<code>\documentclass</code>	304
<code>\dp</code>	153, 583

## E

<code>\else</code>	244, 248, 251
<code>\end</code>	424, 521, 543, 561, 562, 591, 606, 613, 628, 651, 674, 675, 692, 693, 703, 715
<code>\endgroup</code>	101, 203, 214
<code>\endinput</code>	46
environments:	
<code>subarray</code>	<a href="#">162</a>
<code>\errorcontextlines</code>	308
<code>etex</code> (package)	5
<code>etoolbox</code> (package)	<a href="#">2</a>
<code>\exp_args:Nc</code>	501, 509
<code>\exp_args:Nx</code>	14
<code>\exp_not:N</code>	379, 387
<code>\exp_not:n</code>	322, 328, 333, 339
<code>\expandafter</code>	150
<code>expl3</code> (package)	<a href="#">2</a>
<code>\ExplSyntaxOff</code>	477, 511, 680
<code>\ExplSyntaxOn</code>	307, 482, 678

## F

fail (message)	<a href="#">12</a> , <a href="#">313</a>
<code>\fam</code>	517
<code>\fi</code>	174, 253, 254, 256
<code>filehook</code> (package)	<a href="#">2</a>
<code>\fontchardp</code>	159
<code>\fontcharht</code>	156
<code>\fontdimen</code>	168, 169, 170, 242, 243, 245
<code>\frac</code>	<a href="#">2</a> , <a href="#">97</a> , <a href="#">201</a> , 528, 529, 530, 532, 534, 536, 538, 540, 542, 546, 547, 548, 550, 552, 554, 556, 558, 560, 588, 591, 593, 594, 610, 613, 615, 616, 688, 689
functions:	
<code>contains_space</code>	<a href="#">14</a> , <a href="#">403</a>
<code>module</code>	<a href="#">10</a>
<code>print_class_fam_slot</code>	<a href="#">11</a> , <a href="#">296</a>
<code>print_fam_slot</code>	<a href="#">11</a> , <a href="#">290</a>

## G

<code>\g_test_family_int</code>	488, 489, 490, 491
<code>\g_test_mathgroup_int</code>	494, 495, 496, 497
<code>\genfrac</code>	<a href="#">2</a> , 601, 602, 603, 604, 605, 623, 624, 625, 626, 627
<code>\group_begin:</code>	62, 105, 207, 220
<code>\group_end:</code>	67, 73, 105, 207, 220

## H

<code>\hbox</code>	147
<code>\hbox_set:Nn</code>	426, 450, 451
<code>\hfil</code>	174, 176, 192, 198
<code>\ht</code>	152, 257, 258, 582

## I

<code>\ialign</code> .....	172, 190
<b>icomma</b> (package) .....	<a href="#">1</a> , <a href="#">2</a> , <a href="#">10</a> , <a href="#">20</a> , <a href="#">21</a>
<code>\ifx</code> .....	174, 241, 246, 249
<code>\int_compare_p:nNn</code> .....	374
<code>\int_const:Nn</code> .....	116, 117
<code>\int_eval:n</code> .....	11, 85, 121
<code>\int_if_even_p:n</code> .....	396
<code>\int_if_odd:nTF</code> .....	427, 452
<code>\int_if_odd_p:n</code> .....	399
<code>\int_mod:nn</code> .....	486
<code>\int_new:N</code> .....	358, 359
<code>\int_set:Nn</code> .....	372
<code>\int_use:N</code> .....	381, 389, 429, 454, 455

## L

<b>l3kernel</b> (package) .....	<a href="#">21</a>
<code>\l_lltxmath_equal_mathchar</code> .....	<a href="#">124</a> , 133, 135
<code>\l_lltxmath_minus_mathchar</code> .....	<a href="#">124</a> , 132, 134
<code>\l_test_tmpa_box</code> .....	<a href="#">401</a> , 426, 429, 450, 454
<code>\l_test_tmpa_dim</code> .....	363
<code>\l_test_tmpa_int</code> .....	<a href="#">358</a> , <a href="#">363</a> , 372, 374, 376, 381, 389
<code>\l_test_tmpb_box</code> .....	<a href="#">401</a> , 451, 455
<code>\l_test_tmpb_dim</code> .....	364
<code>\l_test_tmpb_int</code> .....	<a href="#">358</a> , <a href="#">363</a>
<code>\Let@</code> .....	165, 181
<code>\lineskip</code> .....	170, 171, 188, 189
<code>\lineskiplimit</code> .....	171, 189
<code>\lltxmath_assert_eq:NN</code> .....	<a href="#">52</a> , 89
<code>\lltxmath_char_dim:NN</code> .....	<a href="#">118</a> , 156, 159
<code>\lltxmath_patch:cNnnn</code> .....	<a href="#">59</a> , 201
<code>\lltxmath_patch:NNwnn</code> .....	<a href="#">59</a> , 90, 99, 146, 162, 210, 231
<code>\lltxmath_restore_catcode:N</code> .....	<a href="#">10</a> , 15
<code>\lltxmath_set_mathchar:NN</code> .....	<a href="#">82</a> , 127, 128, 143, 144, 270
<code>\lltxmath_temp:w</code> .....	<a href="#">48</a> , 63, 64, 72
<code>\lua_now:x</code> .....	84, 120, 428, 453
luatex-required (message) .....	<a href="#">18</a>
<code>\luatex_if_engine:F</code> .....	44
<code>\luatexalignmark</code> .....	<a href="#">110</a> , 196
<b>luatexbase</b> (package) .....	<a href="#">2</a> , <a href="#">4</a>
<b>luatexbase-modutils</b> (package) .....	<a href="#">10</a>
<code>\luatexbase@ensure@primitive</code> ..	<a href="#">49</a> , <a href="#">50</a> , <a href="#">51</a> , <a href="#">96</a> , <a href="#">110</a> , <a href="#">111</a> , <a href="#">112</a> , <a href="#">113</a> , <a href="#">114</a> , <a href="#">115</a> , <a href="#">226</a> , <a href="#">227</a> , <a href="#">228</a> , <a href="#">229</a>
<code>\luatexcrampeddisplaystyle</code> .....	<a href="#">226</a> , 533, 551
<code>\luatexcrampedscriptscriptstyle</code> .....	<a href="#">226</a>
<code>\luatexcrampedscriptstyle</code> .....	<a href="#">226</a> , 541, 559
<code>\luatexcrampedtextstyle</code> .....	<a href="#">226</a> , 537, 555
<code>\luatexmathstyle</code> .....	<a href="#">2</a> , 369, 372, 379, 387
<code>\luatexUmathchardef</code> .....	<a href="#">49</a> , 83
<code>\luatexUmathcode</code> .....	<a href="#">49</a> , 567, 569, 574, 576
<code>\luatexUmathcodenum</code> .....	<a href="#">49</a> , 134, 135
<code>\luatexUmathstackdenomdown</code> .....	<a href="#">113</a> , 186
<code>\luatexUmathstacknumup</code> .....	<a href="#">113</a> , 185
<code>\luatexUmathstackvgap</code> .....	<a href="#">113</a> , 188
<code>\luatexUstack</code> .....	<a href="#">96</a> , 105, 207, 219
<code>\luatexUstartmath</code> .....	<a href="#">110</a> , 193
<code>\luatexUstopmath</code> .....	<a href="#">110</a> , 197

## M

<code>\m@th</code> .....	175, 194, 235, 451
--------------------------	--------------------



macro-expected (message)	32
<code>\makeatletter</code>	448, 481, 570, 577, 581
<code>\makeatother</code>	476, 512, 572, 579, 584
<code>\mathchardef</code>	140, 141, 148, 268
<code>\mathcode</code>	140, 141, 148, 268
<code>\mathcomma</code>	266, 702, 714
<code>\mathgroup</code>	91, 93, 518
<code>\Mathstrutbox@</code>	152, 153, 155, 158, 582, 583
<code>\mathstyle</code>	2
<b>mathtools (package)</b>	1, 2, 9, 17, 19
<code>\meaning</code>	150
messages:	
different-meanings	22
fail	12, 313
luatex-required	18
macro-expected	32
pass	11, 309
patch-macro	41
wrong-meaning	35
module (function)	10
<code>\msg_error:nn</code>	45
<code>\msg_error:nmx</code>	315
<code>\msg_error:nnxxxx</code>	54
<code>\msg_info:nmx</code>	65, 311
<code>\msg_new:nmn</code>	18, 32, 35, 41, 309, 313
<code>\msg_new:nnnn</code>	22
<code>\msg_redirect_class:nn</code>	679
<code>\msg_warning:nmx</code>	76
<code>\msg_warning:nnxxx</code>	70
<code>\mskip</code>	451
<code>\MT_cramped_internal:Nn</code>	230

## N

<b>nath (package)</b>	5
<code>\NeedsTeXFormat</code>	2
<code>\new@mathgroup</code>	89, 90, 95, 495
<code>\NewDocumentCommand</code>	360, 365, 368, 395, 398, 425, 449
<code>\newfam</code>	89, 95, 489
<code>\newsavebox</code>	566
<code>\nulldelimiterspace</code>	237

## O

<code>\over</code>	101, 105
--------------------	----------

## P

packages:	
<code>amsmath</code>	1, 2, 5--7, 17, 18, 20
<code>etex</code>	5
<code>etoolbox</code>	2
<code>expl3</code>	2
<code>filehook</code>	2
<code>icomma</code>	1, 2, 10, 20, 21
<code>l3kernel</code>	21
<code>luatexbase</code>	2, 4
<code>luatexbase-modutils</code>	10
<code>mathtools</code>	1, 2, 9, 17, 19
<code>nath</code>	5
<code>unicode-math</code>	1, 2, 20, 21
<code>xparse</code>	11

pass (message)	11, 309
patch-macro (message)	41
\prg_case_int:nmn	486
\prg_do_nothing:	48
\prg_stepwise_inline:nmnn	483
print_class_fam_slot (function)	11, 296
print_fam_slot (function)	11, 290
\ProvidesExplPackage	4

## R

\radical	238
\relax	140, 141, 148, 150, 214, 567, 574
\RequireLuaModule	9
\RequirePackage	3, 6, 7, 8
\resetMathstrut@	146
\restore@math@cr	166, 182
Robertson, Will	1

## S

\sbox	233, 580
\scan_stop:	87, 220
\scriptfont	168, 169, 170, 250
\scriptscriptfont	252
\scriptstyle	175, 185, 186, 188, 195, 249, 539, 557
\setbox	147
\setmathfont	683, 707
\sixt@n	91
\skip_set:Nn	184
\sqrt	527, 530, 545, 548, 586, 608, 687
\std@equal	133, 141, 144, 578
\std@equals	132
\std@minus	132, 140, 143, 571
\subarray	162
subarray (environment)	162
\substack	588, 610
\sum_	587, 590, 609, 612

## T

\tbinom	600, 622
\test_assert_cramped:Nx	371, 396, 399
\test_assert_equal:ccccmn	342, 354
\test_assert_equal:nmn	353, 361, 366
\test_assert_equal:NNNNNm	342
\test_equal_fail:nxx	331, 349
\test_equal_pass:nxx	320, 347
\test_fail:x	314, 332, 386, 442, 469, 504
\test_pass:x	310, 321, 378, 437, 463
\TestAlphabet	508, 515
\testbox	564, 580, 582, 583
\textfont	119, 149, 242, 243, 247
\textstyle	246, 535, 553
\tfrac	596, 618, 691
\the	149
\thinmuskip	701, 713
\thr@@	170
\tl_const:Nn	317
\tl_const:Nx	317, 318, 319
\tl_replace_once:Nmn	139, 267
\tl_to_str:n	438, 443, 464, 466, 470, 472

<code>\token_if_eq_meaning:NNT</code> .....	192
<code>\token_if_macro:NTF</code> .....	61
<code>\token_to_meaning:N</code> .....	55, 56, 71, 72
<code>\token_to_str:N</code> .....	55, 56, 66, 71, 77
<code>\tw@</code> .....	168, 169

## U

<code>unicode-math</code> (package) .....	1, 2, 20, 21
<code>\use:c</code> .....	262
<code>\usepackage</code> .....	305, 306, 480, 524, 565, 568, 573, 681, 682, 684, 696, 697, 706, 708, 709

## V

<code>\vcenter</code> .....	163, 179
-----------------------------	----------

## W

wrong-meaning (message) .....	<a href="#">35</a>
-------------------------------	--------------------

## X

<code>xparse</code> (package) .....	11
-------------------------------------	----

## Z

<code>\z@</code> .....	147, 152, 153, 233, 237, 238, 257, 258, 259
------------------------	---