The lualatex-math package[*]

Philipp Stephani
p.stephani2@gmail.com

2015/09/22

Contents

1 Introduction

LuaTeX 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 2ε packages is not a task the average LaTeX user should have to care about. Therefore a multitude of LaTeX 2ε 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 TeX documents written for traditional engines should work just fine with LuaTeX, 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 LuaLaTeX.

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

[*] This document corresponds to lualatex-math v1.5, dated 2015/09/22.

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 LaTeX 2$_\varepsilon$ 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 tracker[1].

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 LuaLaTeX 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 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 LaTeX 2$_\varepsilon$ kernel and the amsmath, amsopn, 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 LaTeX.

`\mathstyle`

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

One user-visible change is that the new `\mathstyle` primitive 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 LaTeX 2$_\varepsilon$ package

3.1 Requirements

```
⟨*package⟩
⟨@@=lltxmath⟩
\NeedsTeXFormat{LaTeX2e}[2009/09/24]
\RequirePackage{expl3}[2015/09/07]
\ProvidesExplPackage{lualatex-math}{2015/09/22}{1.5}%
  {Patches for mathematics typesetting with LuaLaTeX}
\RequirePackage { etoolbox } [ 2007/10/08 ]
\cs_if_exist:NF \newluabytecode
  { \RequirePackage { luatexbase } [ 2010/05/27 ] }
\RequirePackage { filehook } [ 2011/03/09 ]
\directlua{require("lualatex-math")}
```

`\@@_restore_catcode:N`

Executing the exhaustive expansion of `\@@_restore_catcode:N`⟨*character token*⟩ restores the category code of the ⟨*character token*⟩ to its current value.

```
\cs_new_nopar:Npn \@@_restore_catcode:N #1 {
  \char_set_catcode:nn { \int_eval:n { `#1 } }
    { \char_value_catcode:n { `#1 } }
}
```

[1] https://github.com/phst/lualatex-math/issues

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.

```
\exp_args:Nx \AtEndOfPackage {
  \@@_restore_catcode:N \$
}
\char_set_catcode_math_toggle:N \$
```

3.2 Messages

luatex-required Issued when not running under LuaTEX.

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

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

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

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.

```
\msg_new:nnn { lualatex-math } { wrong-meaning } {
  I've~ expected~ #1~ to~ have~ the~ meaning \\
  #3, \\
  but~ it~ has~ the~ meaning \\
  #2.
}
```

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

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

3.3 Initialization

Unless we are running under LuaTEX, we issue an error and quit immediately.

```
\sys_if_engine_luatex:F {
  \msg_error:nn { lualatex-math } { luatex-required }
  \endinput
}
```

3.4 Patching

\@@_temp:w A scratch macro.

```
\cs_new_eq:NN \@@_temp:w \prg_do_nothing:
```

\@@_patch:NNnnn
\@@_patch:cNnnn The auxiliary macro `\@@_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.

```
\cs_new_protected_nopar:Npn \@@_patch:NNnnn #1 #2 #3 #4 #5 {
  \cs_if_exist:NT #1 {
    \token_if_macro:NTF #1 {
      \group_begin:
      #2 \@@_temp:w #3 { #4 }
      \cs_if_eq:NNTF #1 \@@_temp:w {
        \msg_info:nnx { lualatex-math } { patch-macro }
          { \token_to_str:N #1 }
        \group_end:
        #2 #1 #3 { #5 }
      } {
        \msg_warning:nnxxx { lualatex-math } { wrong-meaning }
          { \token_to_str:N #1 } { \token_to_meaning:N #1 }
          { \token_to_meaning:N \@@_temp:w }
        \group_end:
      }
    } {
      \msg_warning:nnx { lualatex-math } { macro-expected }
        { \token_to_str:N #1 }
    }
  }
}
\cs_generate_variant:Nn \@@_patch:NNnnn { c }
```

\@@_set_mathchar:NN

The macro `\@@_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*⟩. We cannot use the `\Umathcharnumdef` primitive here since we would then rely on the `\Umathcodenum` primitive which is currently broken.[2]

```
\cs_new_protected_nopar:Npn \@@_set_mathchar:NN #1 #2 {
  \utex_mathchardef:D #1
  \lua_now_x:n {
    lualatex.math.print_class_fam_slot( \int_eval:n { `#2 } )
  }
  \scan_stop:
}
```

3.5 LaTeX 2ε kernel

LuaTeX 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 TeX remnants `\choose`, `\brack` and `\brace` should be discouraged.

\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 2ε kernel definition if the amsmath package is not loaded; the corresponding patch for amsmath follows below.

```
\AtEndPreamble {
  \@ifpackageloaded { amsmath } { } {
    \@@_patch:NNnnn \frac \cs_set_nopar:Npn { #1 #2 } {
```

[2] http://tug.org/pipermail/luatex/2012-October/003794.html

```
   {
    \begingroup #1 \endgroup \over #2
   }
  } {
```

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

```
   {
    \utex_stack:D { \group_begin: #1 \group_end: \over #2 }
   }
  }
 }
}
```

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.

\c_@@_std_minus_mathcode_int
\c_@@_std_equal_mathcode_int

These constants contain the standard TEX mathematical codes for the minus and 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.

```
\int_const:Nn \c_@@_std_minus_mathcode_int { "2200 }
\int_const:Nn \c_@@_std_equal_mathcode_int { "303D }
```

\l_@@_minus_mathchar
\l_@@_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`.

```
\tl_new:N \l_@@_minus_mathchar
\tl_new:N \l_@@_equal_mathchar
\AtBeginOfPackageFile { amsmath } {
 \@@_set_mathchar:NN \l_@@_minus_mathchar \-
 \@@_set_mathchar:NN \l_@@_equal_mathchar \=
```

Now we temporarily reset the mathematical codes.

```
 \char_set_mathcode:nn { `\- } { \c_@@_std_minus_mathcode_int }
 \char_set_mathcode:nn { `\= } { \c_@@_std_equal_mathcode_int }
 \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.

```
  \cs_set_eq:NN \std@minus \l_@@_minus_mathchar
  \cs_set_eq:NN \std@equal \l_@@_equal_mathchar
```

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

```
    \utex_mathcodenum:D `\- \l_@@_minus_mathchar
    \utex_mathcodenum:D `\= \l_@@_equal_mathchar
  }
}
```

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

\@begindocumenthook

amsmath repeats the definiton 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.

```
\AtEndOfPackageFile * { amsmath } {
  \tl_replace_once:Nnn \@begindocumenthook {
    \mathchardef \std@minus \mathcode `\- \relax
    \mathchardef \std@equal \mathcode `\= \relax
  } {
    \@@_set_mathchar:NN \std@minus \-
    \@@_set_mathchar:NN \std@equal \=
  }
```

subarray

The subarray environment uses legacy font dimensions. We simply patch it to use LuaTEX font parameters (and LATEX3 expressions instead of TEX 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).

```
  \@@_patch:NNnnn \subarray \cs_set:Npn { #1 } {
    \vcenter
    \bgroup
    \Let@
    \restore@math@cr
    \default@tag
    \baselineskip \fontdimen 10~ \scriptfont \tw@
    \advance \baselineskip \fontdimen 12~ \scriptfont \tw@
⟨@@=⟩
    \lineskip \thr@@ \fontdimen 8~ \scriptfont \thr@@
⟨@@=lltxmath⟩
    \lineskiplimit \lineskip
    \ialign
    \bgroup
    \ifx c #1 \hfil \fi
    $ \m@th \scriptstyle ## $
    \hfil
    \crcr
  } {
    \vcenter
    \c_group_begin_token
    \Let@
    \restore@math@cr
    \default@tag
    \skip_set:Nn \baselineskip {
      \utex_stacknumup:D \scriptstyle
      + \utex_stackdenomdown:D \scriptstyle
    }
    \lineskip \utex_stackvgap:D \scriptstyle
    \lineskiplimit \lineskip
    \ialign
    \c_group_begin_token
    \token_if_eq_meaning:NNT c #1 { \hfil }
    \utex_startmath:D
    \m@th
    \scriptstyle
    \luatex_alignmark:D \luatex_alignmark:D
    \utex_stopmath:D
    \hfil
    \crcr
  }
```

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

```
  \@@_patch:cNnnn { frac~ } \cs_set:Npn { #1 #2 } {
    {
⟨@@=⟩
      \begingroup #1 \endgroup \@@over #2
    }
  } {
    {
      \utex_stack:D { \group_begin: #1 \group_end: \@@over #2 }
⟨@@=lltxmath⟩
    }
  }
```

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

```
  \@@_patch:NNnnn \@genfrac \cs_set_nopar:Npn {
    #1 #2 #3 #4 #5
  } {
    {
      #1 { \begingroup #4 \endgroup #2 #3 \relax #5 }
    }
  } {
    {
      #1 {
        \utex_stack:D {
          \group_begin: #4 \group_end: #2 #3 \scan_stop: #5
        }
      }
    }
  }
}
```

3.7 amsopn

The amsopn package can be used standalone, but is also loaded by amsmath. It provides the \DeclareMathOperator command which breaks when the minus character is a Unicode math character; this issue was brought to my attention by Jean-François Burnol.

\newmcodes@ We only need to patch one usage of \mathcode in the internal macro \newmcodes@, which is called by all user-defined operators.

```
\group_begin:
\char_set_catcode_other:N \"
\AtEndOfPackageFile * { amsopn } {
  \@@_patch:NNnnn \newmcodes@ \cs_gset_nopar:Npn { } {
    \mathcode `\' 39
    \mathcode `\* 42
    \mathcode `\. "613A
    \ifnum \mathcode `\- = 45 ~ \else
      \mathchardef \std@minus \mathcode `\- \relax
    \fi
    \mathcode `\- 45
    \mathcode `\/ 47
    \mathcode `\: "603A \relax
  } {
    \char_set_mathcode:nn { `\' } { 39 }
    \char_set_mathcode:nn { `\* } { 42 }
    \char_set_mathcode:nn { `\. } { "613A }
    \int_compare:nNnF { \utex_mathcodenum:D `\- } = { 45 } {
      \@@_set_mathchar:NN \std@minus \-
    }
    \char_set_mathcode:nn { `\- } { 45 }
    \char_set_mathcode:nn { `\/ } { 47 }
    \char_set_mathcode:nn { `\: } { "603A }
  }
}
\group_end:
```

3.8 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.

\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.

```
\AtEndOfPackageFile * { mathtools } {
  \@@_patch:NNnnn \MT_cramped_internal:Nn
    \cs_set_nopar:Npn { #1 #2 } {
    \sbox \z@ {
      $
      \m@th
      #1
      \nulldelimiterspace = \z@
      \radical \z@ { #2 }
      $
    }
    \ifx #1 \displaystyle
      \dimen@ = \fontdimen 8 \textfont 3
      \advance \dimen@ .25 \fontdimen 5 \textfont 2
    \else
      \dimen@ = 1.25 \fontdimen 8
      \ifx #1 \textstyle
        \textfont
      \else
        \ifx #1 \scriptstyle
          \scriptfont
        \else
          \scriptscriptfont
        \fi
      \fi
      3
    \fi
    \advance \dimen@ -\ht\z@
    \ht\z@ = -\dimen@
    \box\z@
  } {
```

Here the additional set of braces is absolutely necessary, otherwise the changed mathematical style would be applied to the material after the \mathchoice construct. As the original command works in both text and math mode, we use \ensuremath here.

```
    {
      \ensuremath {
        \use:c { luatex_cramped \cs_to_str:N #1 :D } #2
      }
    }
  }
}
```

3.9 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.[3]

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

```
\AtEndOfPackageFile * { icomma } {
  \tl_replace_once:Nnn \@begindocumenthook {
    \mathchardef \mathcomma \mathcode `\,
  } {
    \@@_set_mathchar:NN \mathcomma \,
  }
}
⟨/package⟩
```

4 Implementation of the LuaLaTeX module

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

```
⟨*lua⟩
lualatex = lualatex or {}
lualatex.math = lualatex.math or {}
luatexbase.provides_module({
  name = "lualatex-math",
  date = "2013/08/03",
  version = 1.3,
  description = "Patches for mathematics typesetting with LuaLaTeX",
  author = "Philipp Stephani",
  licence = "LPPL v1.3+"
})
```

unpack The function unpack needs to be treated specially as it got moved around in Lua 5.2.

```
local unpack = unpack or table.unpack
```

```
local cctb = luatexbase.catcodetables or
  {string = luatexbase.registernumber("catcodetable@string")}
```

[3] https://groups.google.com/forum/#!topic/de.comp.text.tex/Cputk-AJS5I/discussion

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 ⟨*class*⟩␣⟨*family*⟩␣⟨*slot*⟩, suitable for the right-hand side of `\Umathchardef`.

```
function lualatex.math.print_class_fam_slot(char)
  local code = tex.getmathcode(char)
  local class, family, slot = unpack(code)
  local result = string.format("%i %i %i ", class, family, slot)
  tex.sprint(cctb.string, result)
end

return lualatex.math
⟨/lua⟩
```

Change History

v0.1
- General: Initial version 1

v0.2
- General: Added patch for the icomma package 9

v0.3
- General: Patched math group allocation to gain access to all families 4

v0.3a
- General: Updated for changes in l3kernel 1

v0.3b
- `\@begindocumenthook`: Another update for a change in l3kernel 5

v0.3c
- `\@@_set_mathchar:NN`: l3kernel renamed `\lua_now:x` to `\lua_now_x:n` 4

v1.0
- General: Switched to l3docstrip 1

v1.1
- `\@@_set_mathchar:NN`: Update reasoning why `\Umathcharnumdef` is not used here 4
- General: Add fix and unit test for amsopn 7

v1.2
- `\l_@@_equal_mathchar`: Replace removed macro `\chk_if_free_cs:N` 5

v1.3
- General: Stop using the deprecated `module` function 9
- `unpack`: Integrate Philipp Gesang's patch to make the `unpack` function compatible with Lua 5.2 9

v1.3a
- `\@@_set_mathchar:NN`: l3kernel has (currently) dropped `\lua_now_x:n` 4

v1.4
- General: Removed patch for math group allocation; the kernel itself now supports all available math families 4
- `\MT_cramped_internal:Nn`: Added `\ensuremath` to work around issue 11 8

v1.4a
- `\@@_set_mathchar:NN`: `\lua_now_x:n` is back 4
- General: Avoid `\RequireLuaModule` 2
  - Load luatexbase only if required 2
  - Load all of luatexbase 9
  - Pick up new name for string catcode table where available 9
  - Use expl3 versions of LuaTEX math primitives 2

v1.5
- General: Removed patch for `\Mathstrutbox@`; amsmath now has a definition usable in LuaLATEX 6
  - Removed unused helper macro `\@@_char_dim:NN` 5
  - Removed unused Lua function `print_fam_slot` 9

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.