

The luamml package ^{*}

Marcel Krüger

October 30, 2026

1 Use case

When generating output for the web or tagged output, mathematical content should often be represented as MathML. This uses LuaTeX callbacks to automatically attempt to convert LuaTeX math mode output into MathML.

2 Usage

The `luamml` package is designed to be used in automated ways by other packages and usually should not be invoked directly by the end user.

Typically you will use it through `latex-lab-math`. See the `latex-lab-math` documentation for user instructions.

If you want to use `luamml` in another context, the `luamml` package can be included directly, followed by enclosing blocks which should generate files with `\luamml_begin_single_file:` and `\luamml_end_single_file:`. The filename can be set with `\luamml_set_filename:n`.

3 Improving MathML conversion

When using constructs which do not automatically get converted in acceptable form, conversion hints can be provided with `\luamml_annotate:en`. This allows to provide a replacement MathML structure in Lua table form, for example

```
\luamml_annotate:en {  
  nucleus = true,  
  core = {[0] = 'mi', 'TeX'},  
}{  
  \hbox{\TeX}  
}
```

produces a `<mi>TeX</mi>` element in the output instead of trying to import `TeX` as a mathematical expression.

It is possible to add a structure around the construct, stash that structure and then to tell `\luamml_annotate:en` to insert it later inside the math. For this the keys `struct` (which takes a label as argument) or `structnum` (which takes a structure number) can be used. For example

^{*}This document corresponds to `luamml` v0.9.3, dated 2026-10-30.

```

$a = b \quad
\tagstructbegin{tag=mtext,stash}\tagmcbegin{}
\luamml_annotate:en{nucleus=true,structnum=\tag_get:n{struct_num}}
{\mbox{some~text~with~\emph{structure}}}}
\tagmcbend\tagstructend
$

```

Such a construction should check that the flag for structure elements has actually been set to avoid orphaned structures if the stashed structure is ignored.

More about the table structure is explained in an appendix.

4 Features & Limitations

Currently all mathematical expressions which purely contain Unicode encoded math mode material without embedded non-math should get converted successfully. Usage with non-Unicode math (T_EX's 8-bit math fonts) is highly experimental and undocumented. Any attempt to build complicated structures by embedding arbitrary T_EX code in the middle of math mode needs to have a MathML replacement specified. We try to automate more cases in the future.

A Luamml's representation of XML and MathML

In the following I assume basic familiarity with both LuaT_EX's representation of math nodes and MathML.

A.1 Representation of XML elements

In many places, luamml passes around XML elements. Every element is represented by a Lua table. Element 0 must always be present and is a string representing the tag name. The positive integer elements of the table represent child elements (either strings for direct text content or nested tables for nested elements). All string members which do not start with a colon are attributes, whose value is the result of applying `tostring` to the field value. This implies that these values should almost always be strings, except that the value 0 (since it never needs a unit) can often be set as a number. For example the XML document

```

<math block="display">
  <mn>0</mn>
  <mo> &lt; </mo>
  <mi mathvariant="normal">x</mi>
</math>

```

would be represented by the Lua table

```

{[0] = "math", block="display",
 {[0] = "mn", "0"},
 {[0] = "mo", "<"},
 {[0] = "mi", mathvariant="normal", "x"}
}

```

A.2 Expression cores

MathML knows the concept of “embellished operators”:

The precise definition of an “embellished operator” is:

- an `<mo>` element;
- or one of the elements `<msub>`, `<msup>`, `<msubsup>`, `<munder>`, `<mover>`, `<munderover>`, `<mmultiscripts>`, `<mfrac>`, or `<semantics>` (§ 5.1 Annotation Framework), whose first argument exists and is an embellished operator;
- or one of the elements `<mstyle>`, `<mphantom>`, or `<mpadded>`, such that an `mrow` containing the same arguments would be an embellished operator;
- or an `<maction>` element whose selected sub-expression exists and is an embellished operator;
- or an `<mrow>` whose arguments consist (in any order) of one embellished operator and zero or more space-like elements.

For every embellished operator, MathML calls the `<mo>` element defining the embellished operator the “core” of the embellished operator.

`Luamml` makes this slightly more general: Every expression is represented by a pair of two elements: The expression and its core. The core is always a `<mo>`, `<mi>`, or `<mn>`, `nil` or `s` special marker for space like elements.

If and only if the element is a embellished operator the core is a `<mo>` element representing the core of the embellished operator. The core is a `<mi>` or a `<mn>` element if and only if the element would be an embellished operator with this core if this element where a `<mo>` element. The core is the special space like marker for space like elements. Otherwise the core is `nil`.

A.3 Translation of math noads

A math lists can contain the following node types: noad, fence, fraction, radical, accent, style, choice, ins, mark, adjust, boundary, whatsit, penalty, disc, glue, and kern. The “noads”

A.3.1 Translation of kernel noads

The math noads of this list contain nested kernel noads. So in the first step, we look into how kernel nodes are translated to math nodes.

math_char kernel noads First the family and character value in the `math_char` are used to lookup the Unicode character value of this `math_char`. (For `unicode-math`, this is usually just the character value. Legacy maths has to be remapped based on the family.) Then there are two cases: The digits 0 to 9 are mapped to `<mn>` elements, everything else becomes a `<mi>` element with `mathvariant` set to `normal`. (The `mathvariant` value might get suppressed if the character defaults to mathvariant `normal`.) In either case, the `tex:family` attribute is set to the family number if it's not 0.

The core is always set to the expression itself. E.g. the `math_char` kernel noad `\fam3 a` would become (assuming no remapping for this family)

```
{[0] = 'mi',
  mathvariant = 'normal',
  ["tex:family"] = 3,
  "a"
}
```

A.3.2 sub_box kernel noads

I am open to suggestions how to convert them properly.

A.3.3 sub_mlist kernel noads

The inner list is converted as a `<mrow>` element, with the core being the core of the `<mrow>` element. See the rules for this later.

A.3.4 delim kernel noads

If the `small_char` is zero, these get converted as space like elements of the form

```
{[0] = 'mspace',
  width = '1.196pt',
}
```

where 1.196 is replaced by the current value of `\nulldelimiterspace` converted to bp.

Otherwise the same rules as for `math_char` apply, except that instead of `mi` or `<mn>` elements, `mo` elements are generated, `mathvariant` is never set, `stretchy` is set to `true` if the operator is not on the list of default stretchy operators in the MathML specification and `lspace` and `rspace` attributes are set to zero.

A.3.5 acc kernel noads

Depending on the surrounding element containing the `acc` kernel noad, it is either stretchy or not. If it's stretchy, the same rules as for `delim` apply, except that `lspace` and `rspace` are not set. Otherwise the `stretchy` attribute is set to false if the operator is on the list of default stretchy operators.

B Package Implementation

B.1 Initialization

```
1 <@@=luamml>
2 <*luatex>
3 \ProvidesExplPackage {luamml} {2026-10-30} {0.9.3}
4   {Automatically generate presentational MathML from LuaTeX math expressions}
5 </luatex>
6 <*pdfTeX>
7 \ProvidesExplPackage {luamml-pdf} {2026-10-30} {0.9.3}
8   {MathML generation for Lua pdfLaTeX}
9 </pdfTeX>
```

B.2 Initialization

These variable have to appear before the Lua module is loaded and will be used to communicate information to the callback.

Here `\tracingmathml` does not use a `expl3` name since it is not intended for programming use but only as a debugging helper for the user. The other variables are internal, but we provide public interfaces for setting them later.

```
10 \int_new:N \l__luamml_flag_int
```

This integer decides if the xml uses pretty indentation if output to a file or to the terminal. It is used in `luamml-tex.lua` with a bitwise operation. Indentation in files is activated if bit 0 is set. Indentation in the log is activated if bit 1 is set. Bit 2 sets indentation for code added with the `\luamml_register_output_hook:N`. Bit 3 sets indentation for MathML written as stream to the PDF with `\luamml_get_last_mathml_stream:e`.

```
11 \int_new:N \l__luamml_pretty_int
12 \tl_new:N \l__luamml_filename_tl
13 \tl_new:N \l__luamml_root_tl
14 \tl_set:Nn \l__luamml_root_tl { mrow }
15 \tl_new:N \l__luamml_label_tl
16 \pdfTeX\int_new:N \g__luamml_formula_id_int
17 \tl_new:N \l__luamml_label_tl
18
19 \int_set:Nn \l__luamml_pretty_int { 1 }
```

Currently we disable flattening in `LuaTeX` to avoid annotations getting lost. Instead we load `luamml-mathflatten` which reimplements math flattening at a later stage.

```
20 \tl_new:N \l__luamml_mathflattenmode_tl
21 \tl_set:Nn \l__luamml_mathflattenmode_tl { 0 }
```

Now we can load the Lua module which defines the callback. Of course until `pdfTeX` starts implementing `\directlua` this is only done in `LuaTeX`.

```
22 \directlua{lua_load_module("luamml-tex")}
```

B.3 Hook

We also call a hook with arguments at the end of every MathML conversion with the result. Currently only implemented in `LuaTeX` since it immediately provides the output. TODO: the `latex-lab-math` code registers its own function and so effectively kills the hook here. This should be sorted

```
23 \directlua{lua_load_module("luamml-tex")}
24 \hook_new_with_args:nn { luamml / converted } { 1 }
25
26 \cs_new_protected:Npn \__luamml_output_hook:n {
27   \hook_use:nnw { luamml / converted } { 1 }
28 }
29 \__luamml_register_output_hook:N \__luamml_output_hook:n
30 \directlua{lua_load_module("luamml-tex")}
```

B.4 Flags

The most important interface is for setting the flag which controls how the formulas should be converted.

`\luamml_process:` Consider the current formula to be a complete, free-standing mathematical expression which should be converted to MathML. Additionally, the formula is also saved in the `start_math` node as with `\luamml_save:`.

```
31 \cs_new_protected:Npn \luamml_process: {
32   \tl_set:Nn \l__luamml_label_tl {}
33   \int_set:Nn \l__luamml_flag_int { 3 }
34 }
```

Temporarily for compatibility

```
35 \cs_set_eq:NN \luamml_flag_process: \luamml_process:
```

(End of definition for \luamml_process:.)

`\__luamml_maybe_structelem:` A internal helper which can be added to a tag to preserve the external state of the `structelem` flag.

```
36 \cs_new:Npn \__luamml_maybe_structelem: {
37   (
38     8 * \int_mod:nn {
39       \int_div_truncate:nn { \l__luamml_flag_int } {8}
40     } {2}
41   ) +
42 }
```

(End of definition for __luamml_maybe_structelem:.)

`\__luamml_style_to_num:N`

```
43 \cs_new:Npn \__luamml_style_to_num:N #1 {
44   \langle luatex \rangle 32 * #1
45   \langle *pdfTeX \rangle
46   \token_case_meaning:NnF #1 {
47     \displaystyle {0}
48     \textstyle {32}
49     \scriptstyle {64}
50     \scriptscriptstyle {96}
51   } {
52     \Invalid_mathstyle
53   }
54   \langle /pdfTeX \rangle
55 }
```

(End of definition for __luamml_style_to_num:N.)

`\luamml_save:n` Convert the current formula but only save its representation in the `math` node without emitting it as a complete formula. This is useful when the expression forms part of a bigger formula and will be integrated into its MathML tables later by special code. It optionally accepts three parameters: A label, one math style command (`\displaystyle`, `\textstyle`, etc.) which is the implicit math style (so the style which the surrounding code expects this style to have) and a name for the root element (defaults to `mrow`). If the root element name is `mrow`, it will get suppressed in some cases.

```
56 \cs_new_protected:Npn \luamml_save:n #1 {
```

```

57 \tl_set:Nn \l__luamml_label_tl {#1}
58 \int_set:Nn \l__luamml_flag_int { \__luamml_maybe_structelem: 1 }
59 }
60 \cs_new_protected:Npn \luamml_save:nN #1#2 {
61 \tl_set:Nn \l__luamml_label_tl {#1}
62 \int_set:Nn \l__luamml_flag_int { \__luamml_maybe_structelem: 17 + \__luamml_style_to_num:N
63 }
64 \cs_new_protected:Npn \luamml_save:nn #1 {
65 \tl_set:Nn \l__luamml_label_tl {#1}
66 \int_set:Nn \l__luamml_flag_int { \__luamml_maybe_structelem: 5 }
67 \tl_set:Nn \l__luamml_root_tl
68 }
69 \cs_new_protected:Npn \luamml_save:nNn #1#2 {
70 \tl_set:Nn \l__luamml_label_tl {#1}
71 \int_set:Nn \l__luamml_flag_int { \__luamml_maybe_structelem: 21 + \__luamml_style_to_num:N
72 \tl_set:Nn \l__luamml_root_tl
73 }

```

Temporarily for compatibility

```

74 \cs_set_eq:NN \luamml_flag_save:n \luamml_save:n
75 \cs_set_eq:NN \luamml_flag_save:nN \luamml_save:nN
76 \cs_set_eq:NN \luamml_flag_save:nn \luamml_save:nn
77 \cs_set_eq:NN \luamml_flag_save:nNn \luamml_save:nNn

```

(End of definition for \luamml_save:n and others.)

\luamml_ignore: Completely ignore the math mode material.

```

78 \cs_new_protected:Npn \luamml_ignore: {
79 \int_set:Nn \l__luamml_flag_int { 0 }
80 }

```

Temporarily for compatibility

```

81 \cs_set_eq:NN \luamml_flag_ignore: \luamml_ignore:

```

(End of definition for \luamml_ignore:.)

\luamml_structelem: Like \luamml_process:, but additionally adds PDF structure elements. This only works in Lua_{TeX} and requires that the tagpdf package has been loaded *before* luamml.

```

82 <*/uax>
83 \cs_new_protected:Npn \luamml_structelem: {
84 \tl_set:Nn \l__luamml_label_tl {}
85 \int_set:Nn \l__luamml_flag_int { 11 }
86 }

```

Temporarily for compatibility

```

87 \cs_set_eq:NN \luamml_flag_structelem: \luamml_structelem:
88 </uax>

```

(End of definition for \luamml_structelem:.)

\luamml_set_filename:n Allows to set a filename to which the generated MathML gets written. Previous content from the file will get overwritten. This includes results written by a previous formula. Therefore this has to be called separately for every formula or it must expand to different values to be useful. The value is fully expanded when the file is written.

Only complete formulas get written into files (so formulas where `\luamml_process:` or `\luamml_structelem:` are in effect).

Only implemented in Lua_T_E_X, in pdf_T_E_X the arguments for `pdfmml` determine the output location.

```

89 <*luatex>
90 \cs_new_protected:Npn \luamml_set_filename:n {
91   \tl_set:Nn \l__luamml_filename_tl
92 }
93 </luatex>

```

(End of definition for \luamml_set_filename:n.)

`\luamml_begin_single_file:` Everything between these two commands gets written into the same XML file. The filename is expanded when `\luamml_begin_single_file:` gets executed.

(Implemented in Lua)

(End of definition for \luamml_begin_single_file: and \luamml_end_single_file:.)

By default, the flag is set to assume complete formulas.

```

94 \luamml_process:

```

B.5 Annotations

These are implemented very differently depending on the engine, but the interface should be the same.

B.5.1 Lua_T_E_X

```

95 <*luatex>

```

`\luamml_annotate:nen` A simple annotation scheme: The first argument is the number of top level noads to be annotated, the second parameter the annotation and the third parameter the actual list of math tokens. The first argument can be omitted to let Lua_T_E_X determine the number itself.

Passing the first parameter explicitly is useful for any annotations which should be compatible with future pdf_T_E_X versions of this functionality.

```

96 \cs_new_protected:Npn \luamml_annotate:nen #1#2#3 {
97   \__luamml_flattened_mathgroup_begin:
98     #3
99   \__luamml_annotate_end:we \tex_numexpr:D #1 \scan_stop: {#2}
100 }

```

101

```

102 \cs_new_protected:Npn \luamml_annotate:en #1#2 {
103   \__luamml_flattened_mathgroup_begin:
104     #2
105   \__luamml_annotate_end:e {#1}
106 }

```

(End of definition for \luamml_annotate:nen and \luamml_annotate:en.)

`\luamml_attribute:nnn` Set a MathML attribute on a MathML node.

```

\luamml_attribute_core:nnn
107 \cs_new_protected:Npn \luamml_attribute:een #1#2#3 {
108   \__luamml_flattened_mathgroup_begin:
109     #3

```

```

110   \_luamml_annotate_attribute_end:wee {#1} {#2}
111 }
112
113 \cs_new_protected:Npn \luamml_attribute_core:een #1#2#3 {
114   \_luamml_flattened_mathgroup_begin:
115     #3
116   \_luamml_annotate_attribute_end:wee core {#1} {#2}
117 }

```

(End of definition for \luamml_attribute:nnn and \luamml_attribute_core:nnn.)

```

118 </luatex>

```

B.5.2 pdfTeX

```

119 <*pdfTeX>

```

_luamml_pdf_showlists: Here and in many other locations the pdfTeX implementation is based on \showlists, so we define a internal wrapper which sets all relevant parameters.

```

120 \cs_if_exist:NTF \showstream {
121   \iow_new:N \l__luamml_pdf_stream
122   \iow_open:Nn \l__luamml_pdf_stream { \jobname .tml }
123   \cs_new_protected:Npn \_luamml_pdf_showlists: {
124     \group_begin:
125       \int_set:Nn \tex_showboxdepth:D { \c_max_int }
126       \int_set:Nn \tex_showboxbreadth:D { \c_max_int }
127       \showstream = \l__luamml_pdf_stream
128       \tex_showlists:D
129     \group_end:
130   }
131 } {
132   \cs_set_eq:NN \l__luamml_pdf_stream \c_log_iow
133   \cs_set_eq:NN \_luamml_pdf_set_showstream: \scan_stop:
134   \cs_new_protected:Npn \_luamml_pdf_showlists: {
135     \group_begin:
136       \int_set:Nn \l_tmpa_int { \tex_interactionmode:D }
137       \int_set:Nn \tex_interactionmode:D { 0 }
138       \int_set:Nn \tex_showboxdepth:D { \c_max_int }
139       \int_set:Nn \tex_showboxbreadth:D { \c_max_int }
140       \tex_showlists:D
141       \int_set:Nn \tex_interactionmode:D { \l_tmpa_int }
142     \group_end:
143   }
144 }

```

(End of definition for _luamml_pdf_showlists:.)

\luamml_annotate:nen Now we can define the annotation commands for pdfTeX.

```

\luamml_annotate:en
145 \cs_generate_variant:Nn \tl_to_str:n { e }
146 \int_new:N \g__luamml_annotation_id_int
147 \cs_new_protected:Npn \luamml_annotate:nen #1#2#3 {
148   \int_gincr:N \g__luamml_annotation_id_int
149   \iow_shipout_x:Nx \l__luamml_pdf_stream {
150     LUAMML_MARK_REF:
151     \int_use:N \g__luamml_annotation_id_int

```

```

152   :
153 }
154 \iow_now:Nx \l__luamml_pdf_stream {
155   LUAMML_MARK:
156   \int_use:N \g__luamml_annotation_id_int
157   :
158   count = \int_eval:n {#1},
159   #2
160   \iow_newline:
161   LUAMML_MARK_END
162 }
163 #3
164 }
165 \cs_new_protected:Npn \luamml_annotate:en #1#2 {
166   \int_gincr:N \g__luamml_annotation_id_int
167   \iow_shipout_x:Nx \l__luamml_pdf_stream {
168     LUAMML_MARK_REF:
169     \int_use:N \g__luamml_annotation_id_int
170     :
171   }
172   \iow_now:Nx \l__luamml_pdf_stream {
173     LUAMML_MARK:
174     \int_use:N \g__luamml_annotation_id_int
175     :
176     count = data.count[\int_use:N \g__luamml_annotation_id_int],
177     #1
178     \iow_newline:
179     LUAMML_MARK_END
180   }
181   \use:x {
182     \iow_now:Nn \l__luamml_pdf_stream {
183       LUAMML_COUNT:
184       \int_use:N \g__luamml_annotation_id_int
185     }
186     \__luamml_pdf_showlists:
187     \exp_not:n {#2}
188     \iow_now:Nn \l__luamml_pdf_stream {
189       LUAMML_COUNT_END:
190       \int_use:N \g__luamml_annotation_id_int
191     }
192     \__luamml_pdf_showlists:
193   }
194 }

```

(End of definition for \luamml_annotate:nen and \luamml_annotate:en.)

```

195 \</pdfTeX>

```

B.6 Trigger for specific formula

This only applies for pdfTeX since in LuaTeX everything is controlled by the callback, but for compatibility the function is defined anyway.

`\luamml_pdf_write:` We could accept parameters for the flag and tag here, but for compatibility with LuaTeX

they are passed in macros instead.

```

196 <*pdfTeX>
197 \cs_new_protected:Npn \luamml_pdf_write: {
198   \int_gincr:N \g__luamml_formula_id_int
199   \iow_now:Nx \l__luamml_pdf_stream {
200     LUAMML_FORMULA_BEGIN:
201     \int_use:N \g__luamml_formula_id_int
202     :
203     \int_use:N \l__luamml_flag_int
204     :
205     \l__luamml_root_tl
206     :
207     \l__luamml_label_tl
208   }
209   \__luamml_pdf_showlists:
210   \iow_now:Nx \l__luamml_pdf_stream {
211     LUAMML_FORMULA_END
212   }
213 }
214 </pdfTeX>
215 \luatex\cs_new_eq:NN \luamml_pdf_write: \scan_stop:

```

(End of definition for \luamml_pdf_write:.)

B.7 Further helpers

\RegisterFamilyMapping The Lua version of this is defined in the Lua module.

```

216 <*pdfTeX>
217 \NewDocumentCommand \RegisterFamilyMapping {m m} {
218   \iow_now:Nx \l__luamml_pdf_stream {
219     LUAMML_INSTRUCTION:REGISTER_MAPPING: \int_use:N #1 : #2
220   }
221 }
222 </pdfTeX>

```

(End of definition for \RegisterFamilyMapping.)

B.8 Sockets

In various places luamml has to add code to kernel commands. This is done through tagging sockets which are predeclared in ltagging.

name	number of argument
math/luamml/save/nNn	1
math/luamml/save/nn	1
math/luamml/annotate/false	2
math/luamml/array/save	0
math/luamml/array/finalize	0
math/luamml/array/initcol	0
math/luamml/array/finalizecol	1
math/luamml/mtable/finalizecol	1
math/luamml/mtable/innertable/save	0
math/luamml/mtable/smallmatrix/save	0
math/luamml/mtable/innertable/finalize	0
math/luamml/mtable/finalize	1
math/luamml/mtable/aligncol	1
math/luamml/mtable/tag/save	0
math/luamml/mtable/tag/set	0
math/display/tag/begin	0
math/display/tag/end	0
math/luamml/hbox	2
math/luamml/artifact	0
math/luamml/finph0nt	2
math/luamml/finsm0sh	2

B.8.1 Save sockets

These sockets are wrappers around the `\luamml_save:...` commands

```

223 \NewTaggingSocketPlug{math/luamml/save/nNn}{luamml}
224 {
225   \luamml_save:nNn #1
226 }
227 \AssignTaggingSocketPlug{math/luamml/save/nNn}{luamml}
228 \NewTaggingSocketPlug{math/luamml/save/nn}{luamml}
229 {
230   \luamml_save:nn #1
231 }
232 \AssignTaggingSocketPlug{math/luamml/save/nn}{luamml}

```

B.8.2 sockets to annotate content

```

233 <*\luatex>
234 \NewTaggingSocketPlug{math/luamml/annotate/false}{luamml}
235 {
236   \luamml_annotate:en { core = false }
237   {
238     #2
239   }
240 }
241 \AssignTaggingSocketPlug{math/luamml/annotate/false}{luamml}
242 </\luatex>

```

B.8.3 socket plugs for the array package

The luamml support makes only sense with luatex.

```
243 <*\luatex>
244 \AddToHook{package/array/after}{\lua_now:n { require'luamml-array' }}
```

math/luamml/array/save (*plug*) The socket of this plug is used in `\endarray`.

```
245 \NewTaggingSocketPlug{math/luamml/array/save}{luamml}
246 {
247   \__luamml_array_save_array:
248 }
```

math/luamml/array/finalize (*plug*) This socket of this plug is used in `\endarray`.

```
249 \NewTaggingSocketPlug{math/luamml/array/finalize}{luamml}
250 {
251   \mode_if_math:T { \__luamml_array_finalize_array: }
252 }
```

math/luamml/array/initcol (*plug*) The socket of this plug is used in `\@classz`.

```
253 \NewTaggingSocketPlug{math/luamml/array/initcol}{luamml}
254 {
255   \__luamml_array_init_col:
256 }
```

luamml/array/finalizecol (*plug*) The socket of this plug is used in `\@classz`.

```
257 \NewTaggingSocketPlug{math/luamml/array/finalizecol}{luamml}
258 {
259   \__luamml_array_finalize_col:w #1~
260 }

261 \AssignTaggingSocketPlug{math/luamml/array/save}{luamml}
262 \AssignTaggingSocketPlug{math/luamml/array/finalize}{luamml}
263 \AssignTaggingSocketPlug{math/luamml/array/initcol}{luamml}
264 \AssignTaggingSocketPlug{math/luamml/array/finalizecol}{luamml}
265 </\luatex>
```

B.8.4 amsmath alignments

This socket is used at the end of alignment cells and adds the content to the current row.

```
266 <*\luatex>
267 \NewTaggingSocketPlug{math/luamml/mtable/finalizecol}{luamml}
268 {
269   \use:c{__luamml_amsmath_add_#1_to_row:}
270 }
271 \AssignTaggingSocketPlug{math/luamml/mtable/finalizecol}{luamml}
272
273 </\luatex>
```

These sockets save an inner table

```
274 <*\luatex>
275 \NewTaggingSocketPlug{math/luamml/mtable/innertable/save}{luamml}
276 {
277   \__luamml_amsmath_save_inner_table:n \@currentenv
```

```

278 }
279 \AssignTaggingSocketPlug{math/luamml/mtable/innertable/save}{luamml}
280 \NewTaggingSocketPlug{math/luamml/mtable/smallmatrix/save}{luamml}
281 {
282   \_luamml_amsmath_save_smallmatrix:
283 }
284 \AssignTaggingSocketPlug{math/luamml/mtable/smallmatrix/save}{luamml}
285 \NewTaggingSocketPlug{math/luamml/mtable/innertable/finalize}{luamml}
286 {
287   \_luamml_amsmath_finalize_inner_table:
288 }
289 \AssignTaggingSocketPlug{math/luamml/mtable/innertable/finalize}{luamml}
290 \end{luatex}

```

This socket finalize the `mtable` in alignments like `align` or `gather`. It takes an argument, the environment. It should be used normally with `\UseExpandableTaggingSocket`.

```

291 \begin{luatex}
292 \NewTaggingSocketPlug{math/luamml/mtable/finalize}{luamml}
293 {
294   \_luamml_amsmath_finalize_table:n {#1}
295 }
296 \AssignTaggingSocketPlug{math/luamml/mtable/finalize}{luamml}
297 \end{luatex}

```

This socket adds attributes for the alignment in `multline`. It takes an argument, the alignment.

```

298 \begin{luatex}
299 \NewTaggingSocketPlug{math/luamml/mtable/aligncol}{luamml}
300 {
301   \_luamml_amsmath_set_row_columnalign:n {#1}
302 }
303 \AssignTaggingSocketPlug{math/luamml/mtable/aligncol}{luamml}
304 \end{luatex}

```

B.8.5 Tags and labels

These sockets save and set tags and labels in alignments.

```

305 \begin{luatex}
306 \int_new:N \l__luamml_amsmath_tag_struct_num_int
307 \NewTaggingSocketPlug{math/luamml/mtable/tag/save}{luamml}
308 {
309   \_luamml_amsmath_save_tag_with_struct_elem:N \l__luamml_amsmath_tag_struct_num_int
310 }
311 \AssignTaggingSocketPlug{math/luamml/mtable/tag/save}{luamml}
312 \NewTaggingSocketPlug{math/luamml/mtable/tag/set}{luamml}
313 {
314   \_luamml_amsmath_set_tag:
315 }
316 \AssignTaggingSocketPlug{math/luamml/mtable/tag/set}{luamml}
317
318 \end{luatex}

```

If math structure elements are created the Lbl-structure of a tag must be moved inside the math structure, typically as an additional column in an `mtable` with an intent `:equation-label` or `:no-equation-label`.

The `luamml`-code handles this by stashing the Lbl-structure, storing the structure number in an array and reusing it once it creates the math structure elements.

This should only be done for specific environments, we define a constant to test:

```

319 <*\luatex>
320 \clist_map_inline:nn
321 {
322   eqnarray,
323   eqnarray*,
324   align,
325   align*,
326   alignat,
327   alignat*,
328   xalignat,
329   xalignat*,

```

there is never a tag/label in `xxalignat`, so does it make sense to add a label column? Left out for now.

```

330   %xxalignat,
331   flalign,
332   flalign*,
333   gather,
334   gather*,

```

`multline` have at most one tag, so we do not use a label column but rely on the external Lbl for now.

```

335   %multline, % NO
336   %multline*, % NO
337   equation,
338   equation*,

```

`split` has never a numbering so is ignored

```

339   %split, % NO
340 }
341 {\tl_const:cn { c__luamml_label_#1_tl}{}}

342 \NewTaggingSocketPlug{math/display/tag/begin}{luamml}
343 {
344   \tag_mc_end:
345   \bool_lazy_and:nnT
346     { \tl_if_exist_p:c { c__luamml_label_ \@currenvir _tl } }
347     { \int_if_odd_p:n { \int_div_truncate:nn { \l__luamml_flag_int } { 8 } } }
348     {
349       \tag_struct_begin:n {tag=mtext,stash}
350       \int_set:Nn \l__luamml_amsmath_tag_struct_num_int { \tag_get:n {struct_num} }
351     }
352   \tag_mc_begin:n {}
353 }
354 \AssignTaggingSocketPlug{math/display/tag/begin}{luamml}
355 \NewTaggingSocketPlug{math/display/tag/end}{luamml}
356 {

```

```

357 \tag_mc_end:
358 \bool_lazy_and:nnT
359 { \tl_if_exist_p:c { c__luamml_label_ \@currenvir _tl } }
360 { \int_if_odd_p:n { \int_div_truncate:nn { \l__luamml_flag_int } { 8 } } }
361 {
362   \tag_struct_end:
363 }
364 \tag_mc_begin:n {}
365 }
366 \AssignTaggingSocketPlug{math/display/tag/end}{luamml}
367 </luatex>

```

B.8.6 Horizontal boxes

This socket annotates an `\hbox` inside box commands used in math. We test for the socket until the release 2025-06-01.

```

368 <*luatex>
369 \NewTaggingSocketPlug{math/luamml/hbox}{luamml}
370 {
371   \bool_lazy_and:nnTF
372   { \mode_if_math_p: }
373   { \int_if_odd_p:n { \int_div_truncate:nn { \l__luamml_flag_int } { 8 } } }
374   {
375     \tag_struct_begin:n
376     {
377       tag=mtext,
378       stash,
379     }
380     \tag_mc_begin:n {}
381     \luamml_annotate:en
382     {
383       nucleus = true,
384       structnum=\tag_get:n{struct_num}
385     }
386     { #2 }
387     \tag_mc_end:
388     \tag_struct_end:
389   }
390   { #2 }
391 }
392 \AssignTaggingSocketPlug{math/luamml/hbox}{luamml}
393 </luatex>

```

B.8.7 Artifact characters

Unicode characters like a root sign should be marked as artifacts to avoid duplication e.g. in derivation if mathml structure elements are used that imply the meaning.

```

394 <*luatex>
395 \NewTaggingSocketPlug{math/luamml/artifact}{luamml}
396 {
397   \int_if_odd:nT { \int_div_truncate:nn { \l__luamml_flag_int } { 8 } }
398   {
399     \tag_mc_begin:n{artifact}

```

```

400     }
401   }
402   \AssignTaggingSocketPlug{math/luamml/artifact}{luamml}
403 \end{luatex}

```

B.8.8 Math phantom socket

This socket is used around `\finph@nt`.

```

404 \begin{luatex}
405 \NewTaggingSocketPlug{math/luamml/finph@nt}{luamml}
406 {
407   \luamml_annotate:nen {1}
408   {
409     nucleus = true,
410     mathml =
411     {
412       [0] = 'mpadded',
413       \ifh@else
414         width = 0,
415       \fi
416       \ifv@else
417         height = 0, depth = 0,
418       \fi
419       consume_label'mathphant',
420     },
421   }
422   { #2 }
423 }
424 \AssignTaggingSocketPlug{math/luamml/finph@nt}{luamml}
425 \end{luatex}

```

B.8.9 Math smash socket

This socket is used around `\finsm@sh`.

```

426 \begin{luatex}
427 \NewTaggingSocketPlug{math/luamml/finsm@sh}{luamml}
428 {
429   \luamml_annotate:nen {2}
430   {
431     nucleus = true,
432     mathml = consume_label('mathsmash',
433       function(padded, core)
434         padded.height, padded.depth = 0, 0~
435         return~padded, core~
436       end),
437   }
438   { #2 }
439 }
440 \AssignTaggingSocketPlug{math/luamml/finsm@sh}{luamml}
441 \end{luatex}

```

B.9 Patching

For some packages, we ship with patches to make them more compatible and to demonstrate how other code can be patched to work with `luamml`.

These are either loaded directly if the packages are loaded or delayed using L^AT_EX's hook system otherwise.

`__luamml_patch_package:nn`
`__luamml_patch_package:n`

For this, we use two helpers: First a wrapper which runs arbitrary code either now (if the package is already loaded) or as soon as the package loads, second an application of the first one to load packages following `luamml`'s naming scheme for these patch packages.

```

442 \cs_new_protected:Npn __luamml_patch_package:nn #1 #2 {
443   \@ifpackageloaded {#1} {#2} {
444     \hook_gput_code:nnn {package/#1/after} {luamml} {#2}
445   }
446 }
447 \cs_new_protected:Npn __luamml_patch_package:n #1 {
448   __luamml_patch_package:nn {#1} {
449     \RequirePackage { luamml-patches-#1 }
450   }
451 }
```

(End of definition for `__luamml_patch_package:nn` and `__luamml_patch_package:n`.)

We currently provide minimal patching for the kernel, `amsmath`. Currently only the kernel code supports pdf_TE_X, but it's planned to extend this.

```

452 \RequirePackage { luamml-patches-kernel }
453 <*luatex>
454 __luamml_patch_package:n {amsmath}
455 </luatex>
```

Index

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

A		D	
<code>\AddToHook</code>	244	<code>\cs_new:Npn</code>	36, 43
<code>\AssignTaggingSocketPlug</code>		<code>\cs_new_eq:NN</code>	215
.	227, 232, 241, 261, 262, 263,	<code>\cs_new_protected:Npn</code> .	26, 31, 56,
264, 271, 279, 284, 289, 296, 303,		60, 64, 69, 78, 83, 90, 96, 102, 107,	
311, 316, 354, 366, 392, 402, 424, 440		113, 123, 134, 147, 165, 197, 442, 447	
		<code>\cs_set_eq:NN</code>	
		. . .	35, 74, 75, 76, 77, 81, 87, 132, 133
B		E	
bool commands:		<code>\directlua</code>	5
<code>\bool_lazy_and:nnTF</code>	345, 358, 371	<code>\displaystyle</code>	6, 47
C		E	
clist commands:		<code>\else</code>	413, 416
<code>\clist_map_inline:nn</code>	320	<code>\endarray</code>	13
cs commands:		exp commands:	
<code>\cs_generate_variant:Nn</code>	145	<code>\exp_not:n</code>	187
<code>\cs_if_exist:NTF</code>	120		

F		
<code>\fi</code>	415, 418	
G		
group commands:		
<code>\group_begin:</code>	124, 135	
<code>\group_end:</code>	129, 142	
H		
<code>\hbox</code>	16	
hook commands:		
<code>\hook_gput_code:nnn</code>	444	
<code>\hook_new_with_args:nn</code>	24	
<code>\hook_use:nnw</code>	27	
I		
int commands:		
<code>\int_div_truncate:nn</code>	39, 347, 360, 373, 397	
<code>\int_eval:n</code>	158	
<code>\int_gincr:N</code>	148, 166, 198	
<code>\int_if_odd:nTF</code>	397	
<code>\int_if_odd_p:n</code>	347, 360, 373	
<code>\int_mod:nn</code>	38	
<code>\int_new:N</code>	10, 11, 16, 17, 146, 306	
<code>\int_set:Nn</code>	19, 33, 58, 62, 66, 71, 79, 85, 125, 126, 136, 137, 138, 139, 141, 350	
<code>\int_use:N</code>	151, 156, 169, 174, 176, 184, 190, 201, 203, 219	
<code>\c_max_int</code>	125, 126, 138, 139	
<code>\l_tmpa_int</code>	136, 141	
Invalid commands:		
<code>\Invalid_mathstyle</code>	52	
iow commands:		
<code>\iow_new:N</code>	121	
<code>\iow_newline:</code>	160, 178	
<code>\iow_now:Nn</code>	154, 172, 182, 188, 199, 210, 218	
<code>\iow_open:Nn</code>	122	
<code>\iow_shipout_x:Nn</code>	149, 167	
<code>\c_log_iow</code>	132	
J		
<code>\jobname</code>	122	
L		
lua commands:		
<code>\lua_load_module:n</code>	21, 22	
<code>\lua_now:n</code>	244	
luamml commands:		
<code>\luamml_annotate:nn</code>	1, 96, 102, 145, 165, 236, 381	
<code>\luamml_annotate:nnn</code>	96, 96, 145, 147, 407, 429	
<code>\luamml_attribute:nnn</code>	107, 107	
<code>\luamml_attribute_core:nnn</code>	107, 113	
<code>\luamml_begin_single_file:</code>	1, 8, 94	
<code>\luamml_end_single_file:</code>	1, 94	
<code>\luamml_flag_ignore:</code>	81	
<code>\luamml_flag_process:</code>	35	
<code>\luamml_flag_save:n</code>	74	
<code>\luamml_flag_save:nN</code>	75	
<code>\luamml_flag_save:nn</code>	76	
<code>\luamml_flag_save:nNn</code>	77	
<code>\luamml_flag_structelem:</code>	87	
<code>\luamml_get_last_mathml_stream:n</code>	5	
<code>\luamml_ignore:</code>	78, 78, 81	
<code>\luamml_pdf_write:</code>	196, 197, 215	
<code>\luamml_process:</code>	7, 8, 31, 31, 35, 94	
<code>\luamml_register_output_hook:N</code>	5	
<code>\luamml_save:</code>	6	
<code>\luamml_save:...</code>	12	
<code>\luamml_save:n</code>	56, 56, 74	
<code>\luamml_save:nN</code>	56, 60, 75	
<code>\luamml_save:nn</code>	56, 64, 76, 230	
<code>\luamml_save:nNn</code>	56, 69, 77, 225	
<code>\luamml_set_filename:n</code>	1, 89, 90	
<code>\luamml_structelem:</code>	8, 82, 83, 87	
luamml internal commands:		
<code>__luamml_amsmath_finalize_inner_table:</code>	287	
<code>__luamml_amsmath_finalize_table:n</code>	294	
<code>__luamml_amsmath_save_inner_table:n</code>	277	
<code>__luamml_amsmath_save_smallmatrix:</code>	282	
<code>__luamml_amsmath_save_tag_with_struct_elem:N</code>	309	
<code>__luamml_amsmath_set_row_columnalign:n</code>	301	
<code>__luamml_amsmath_set_tag:</code>	314	
<code>\l__luamml_amsmath_tag_struct_num_int</code>	306, 309, 350	
<code>__luamml_annotate_attribute_end:wee</code>	110, 116	
<code>__luamml_annotate_end:n</code>	105	
<code>__luamml_annotate_end:we</code>	99	
<code>\g__luamml_annotation_id_int</code>	146, 148, 151, 156, 166, 169, 174, 176, 184, 190	
<code>__luamml_array_finalize_array:</code>	251	
<code>__luamml_array_finalize_col:w</code>	259	
<code>__luamml_array_init_col:</code>	255	
<code>__luamml_array_save_array:</code>	247	
<code>\l__luamml_filename_tl</code>	12, 91	

<code>\l__luamml_flag_int</code>	10, 33, 39, 58, 62, 66, 71, 79, 85, 203, 347, 360, 373, 397
<code>__luamml_flattened_mathgroup_</code> <code>begin:</code>	97, 103, 108, 114
<code>\g__luamml_formula_id_int</code>	16, 198, 201
<code>\l__luamml_label_tl</code>	15, 32, 57, 61, 65, 70, 84, 207
<code>__luamml_maybe_structelem:</code>	36, 36, 58, 62, 66, 71
<code>__luamml_output_hook:n</code>	26, 29
<code>__luamml_patch_package:n</code>	442, 447, 454
<code>__luamml_patch_package:nn</code>	442, 442, 448
<code>__luamml_pdf_set_showstream:</code> ..	133
<code>__luamml_pdf_showlists:</code>	120, 123, 134, 186, 192, 209
<code>\l__luamml_pdf_stream</code>	121, 122, 127, 132, 149, 154, 167, 172, 182, 188, 199, 210, 218
<code>\l__luamml_pretty_int</code>	11, 19
<code>__luamml_register_output_hook:N</code>	29
<code>\l__luamml_root_tl</code>	13, 14, 67, 72, 205
<code>__luamml_style_to_num:N</code>	43, 43, 62, 71
M	
mode commands:	
<code>\mode_if_math:TF</code>	251
<code>\mode_if_math_p:</code>	372
N	
<code>\NewDocumentCommand</code>	217
<code>\NewTaggingSocketPlug</code>	223, 228, 234, 245, 249, 253, 257, 267, 275, 280, 285, 292, 299, 307, 312, 342, 355, 369, 395, 405, 427
P	
Plugs:	
<code>tagssupport/math/luamml/array/finalize</code>	249
<code>tagssupport/math/luamml/array/finalizecol</code>	257
<code>tagssupport/math/luamml/array/initcol</code>	253
<code>tagssupport/math/luamml/array/save</code>	245
<code>\ProvidesExplPackage</code>	3, 7
R	
<code>\RegisterFamilyMapping</code>	216
<code>\RequirePackage</code>	449, 452
S	
scan commands:	
<code>\scan_stop:</code>	20, 99, 133, 215
<code>\scriptscriptstyle</code>	50
<code>\scriptstyle</code>	49
<code>\showlists</code>	9
<code>\showstream</code>	120, 127
T	
tag commands:	
<code>\tag_get:n</code>	350, 384
<code>\tag_mc_begin:n</code> ...	352, 364, 380, 399
<code>\tag_mc_end:</code>	344, 357, 387
<code>\tag_struct_begin:n</code>	349, 375
<code>\tag_struct_end:</code>	362, 388
<code>tagssupport/math/luamml/array/finalize</code> (plug)	249
<code>tagssupport/math/luamml/array/finalizecol</code> (plug)	257
<code>tagssupport/math/luamml/array/initcol</code> (plug)	253
<code>tagssupport/math/luamml/array/save</code> (plug)	245
TeX and L ^A T _E X 2 _ε commands:	
<code>\@classz</code>	13
<code>\@currenvir</code>	277, 346, 359
<code>\@ifpackageloaded</code>	443
<code>\finph@nt</code>	17
<code>\finsm@sh</code>	17
<code>\ifh@</code>	413
<code>\ifv@</code>	416
tex commands:	
<code>\tex_interactionmode:D</code> .	136, 137, 141
<code>\tex_mathflattenmode:D</code>	20
<code>\tex_numexpr:D</code>	99
<code>\tex_showboxbreadth:D</code>	126, 139
<code>\tex_showboxdepth:D</code>	125, 138
<code>\tex_showlists:D</code>	128, 140
<code>\textstyle</code>	6, 48
tl commands:	
<code>\tl_const:Nn</code>	341
<code>\tl_if_exist_p:N</code>	346, 359
<code>\tl_new:N</code>	12, 13, 15
<code>\tl_set:Nn</code>	14, 32, 57, 61, 65, 67, 70, 72, 84, 91
<code>\tl_to_str:n</code>	145
token commands:	
<code>\token_case_meaning:NnTF</code>	46
<code>\tracingmathml</code>	5, 17
U	
use commands:	
<code>\use:N</code>	269
<code>\use:n</code>	181
<code>\UseExpandableTaggingSocket</code>	14