

The `latex-lab-math` code^{*}

Frank Mittelbach, Joseph Wright, L^AT_EX Project

v0.7h 2026-09-23

Abstract

This module handles the tagging of math. It adapts the `math` command and switches to capture math material and then tag this material.

Contents

1	Introduction	2
2	Math capture	3
3	Avoiding math capture	3
3.1	Options to suppressing math capture and tagging	3
3.1.1	Using the trigger token <code>\m@th</code> <i>inside</i> the math	4
3.1.2	Using <code>\m@th</code> <i>before</i> the opening <code>\$</code>	4
3.1.3	Disabling math tagging with <code>\MathCollectFalse</code>	5
4	Math capture interfaces	5
4.1	Code level interfaces	5
4.2	Document level interfaces	5
5	Math tagging	6
5.1	Code requirements	6
5.2	Inline math	6
5.3	Display math	7
5.4	Associated Files	7
5.5	MathML in an attribute	8
5.6	Automatic mathml creation with <code>luamml</code>	8
5.7	Summary of math options	9
6	Debugging	11
7	Known current bugs, etc.	11
7.1	Capture/grabbing problems	11
7.2	Fake math	12
7.2.1	Open problems	12
7.3	Processor	12
7.4	Other problems	13
7.5	Other Todos	13

*

8	The Implementation	13
8.1	File declaration	13
8.2	Setup	13
8.3	Debugging	14
8.4	Data structures	14
8.5	Math attributes	15
8.6	Code related to AF	15
8.7	Mathstyle detection	25
8.8	Tagging options	26
8.8.1	Meta keys	26
8.9	Sockets	28
8.9.1	Main inline math sockets	28
8.9.2	Main display math sockets	28
8.9.3	Sockets plugs for tags (labels)	29
8.9.4	Internal sockets	30
8.10	Interface commands	34
8.11	Content grabbing	35
8.12	Token-by-token inline grabbing	37
8.13	Marking math environments	39
8.14	Regaining control after a display has finished with \$\$	41
8.15	Document commands	44
8.16	<code>\everymath</code> and <code>\everydisplay</code>	45
8.17	Modifying kernel environments	46
8.18	Modifying kernel commands	46
8.19	Disable math grabbing in the <code>begindocument</code> hook	49

Index	49
--------------	-----------

1 Introduction

Tagging math involves a variety of tasks that require that math is captured before the typesetting:

- When typesetting the math MC-tags and structure commands must be inserted at the begin and the end, and perhaps also around lines or other subparts of the equation.
- The source and/or a MathML-representation of the source must be available so that it can be (perhaps after some preprocessing) be used in an associated file or in an alternate text.
- It should be possible to measure the math for, e.g., a `bbox` setting.

This file implements capture of all math mode material at the outer level, i.e., a formula is captured in its entirety with inner text blocks (possibly containing further math) absorbed as part of the formula. For example,

```
\[ a \in A \text{ for all } a<5\$ \]
```

would only result in a single capture of the tokens “`a_\in A_\text{\_for\_all\_}a<5\$`”.

2 Math capture

In the current setup

- `$`, `\(...\)`, `\[...\]` and `$$` grab (eventually through a command in `\everymath`/`cseverydisplay`) if the boolean `\l_@@_collected_bool` is false. If the boolean is true they behave normally and can for example contain verbatim.
- All (registered) environments grab their body regardless of the state of the boolean. For `equation`, `equation*` and `math` this is a change as they no longer can contain verbatim.

3 Avoiding math capture

In most cases when an environment or command switches into math, the semantic meaning of the content *is* math and grabbing and then tagging that as a Formula is adequate.

But there are exceptions, most prominently with the math shift token `$`.

- `$` can be used to center a box with `\vcenter`, for example, in the tabular code¹. The opening `$` is then often in a different command than the closing `$` and the content of the box should be tagged normally, including all math it contains. This means one wants to avoid that the opening `$` triggers the math grabbing and creates a Formula structure, and after the `$` one wants to switch back to normal text and math capture/tagging.
- `$` is used to place superscripts and subscripts, see the definition of `\textsuperscript` which uses `\ensuremath` internally². Inside the superscript in special cases you may want more tagging (including nested math tagging) but typically it is simple text.
- `$` is used to access a char in a math font.

For example the `\meta` command uses internally the math commands `\langle` and `\rangle` around its argument:

`$_\langle\textit{argument}\rangle_$_` which gives $\langle argument \rangle$.

The symbols are then clearly not math. (Beside disabling math tagging one could also use text commands like `\textlangle` and `\textrangle`: $\langle argument \rangle$. Depending on the font that could even look better, but it requires that the font supports the chars, which is not the case for various OpenType fonts.) Additional tagging inside the math should be not needed. If the symbols need an actual text then a Span can be added around the whole material.

3.1 Options to suppressing math capture and tagging

We are therefore providing a number of options to suppress the collection/grabbing of the math and with it the tagging. These commands can be used to control locally the tagging of math. The first two are new commands. `\math` is a standard L^AT_EX command used in a number of places to set `\mathsurround` to zero; with active tagging it is also used to identify math that should not be tagged, because it has traditionally be used

¹This will change in L^AT_EX 2026-06-01. The tabular code will then no longer use `\vcenter` and math

²This will change in L^AT_EX 2026-06-01. `\textsuperscript` and `\textsubscript` will no longer use math mode.

in exactly the places where math mode is used for its layout characteristics and not for representing a formula.³ Details are described below. (`\SuspendTagging` is documented in source2e.pdf.)

To set `\mathsurround` without disabling math tagging, use `\mathsurround\z@` directly.

3.1.1 Using the trigger token `\m@th` *inside* the math

If the grabbed math contains the token `\m@th` the tagging/processing of the math is suppressed. As the math is nevertheless grabbed, this requires that the end math shift token is not hidden in some other command. `\m@th` sets also `\mathsurround` to zero, which is often wanted in untagged math anyway.

Text tagging is *not* suppressed inside the math, e.g., `\mbox{\emph{text}}` will still create an Em-structure. In contrast nested math is not tagged.

To suppress the text tagging `\SuspendTagging{}` can be used:

- no math tagging, but `\emph` is tagged:

$$\text{\m@th\langle\mbox{\emph{if and only if}}\ $x=y$}\rangle$$$
- no internal tagging:

$$\text{\m@th\SuspendTagging{}\langle\mbox{\emph{if and only if}}\ $x=y$}\rangle$$$

The method is quite suitable for symbols and also for sub- and superscripts (as long as they don't contain nested math that should be tagged).

3.1.2 Using `\m@th` *before* the opening `$`

`\m@th` (as defined in the latex-lab-math code) sets also the internal boolean which controls the grabbing to true and so when it is used *before* some math it disables math grabbing and tagging for all following math in the current group (including nested math). Text tagging inside the math is still active, it can be disabled as above with `\SuspendTagging{}`.

When `\m@th` is used like this it is possible to reenale the math tagging of nested math, by adding `\MathCollectTrue` after the opening `$`.

- no math tagging, but `\emph` is tagged:

$$\text{\m@th$\langle\mbox{\emph{if and only if}}\ $x=y$}\rangle$}$$
- both nested math and `\emph` is tagged:

$$\text{\m@th$\MathCollectTrue\langle\mbox{\emph{if and only if}}\ $x=y$}\rangle$}$$

The method is suitable for symbols and sub- and superscripts too (it is actually how superscripts avoid math tagging before L^AT_EX 2026-06-01). It can also be used in cases where the end math shift token is hidden in some other command. But it is not so useful for larger boxes as it sets `\mathsurround` to zero. Grouping should be used to avoid side-effects on following math.

³This way even code that is not adjusted up for tagging is handled correctly if it uses `\m@th`.

3.1.3 Disabling math tagging with `\MathCollectFalse`

Without a side-effect on `\mathsurround` the math grabbing can be suppressed and re-enabled by setting the boolean that controls the collecting. So in the following example the math in the `\mbox` is properly tagged, but the angled brackets are simple text.

```
%stop math grabbing
\MathCollectFalse
$
%(optional) restart math grabbing for nested math
\MathCollectTrue
\langle\mbox{\emph{if and only if}} $x=y$\rangle
$
%(optional) restart math grabbing for following math
% if there is no grouping
\MathCollectTrue
\quad $x=1$
```

The method is suitable if a box should be centered with `\vcenter` (and before L^AT_EX 2026-06-01 was used in `array.sty` for the tabular code).

4 Math capture interfaces

4.1 Code level interfaces

<code>\math_register_env:n</code>	<code>\math_register_env:n {<env>}</code>
<code>\math_register_env:nn</code>	<code>\math_register_env:nn {<env>} {<options>}</code>

Registers the `<env>` as a math environment which should be captured and made available. This is necessary for all top-level math mode environments: low-level errors may result if these are not correct set up. One or more key-value `<options>` may also be given:

arg-spec The argument specification taken by the beginning of the environment; this is used to remove non-mathematical material.

<code>\math_processor:n</code>	<code>\math_processor:n {<tokens>}</code>
--------------------------------	---

Declares that the captured math content should be passed to the `<tokens>`, which will receive the environment type as `#1` and the content as `#2`. The processing is done before the typesetting. It is not applied if `\ifmeasuring@` is true.

4.2 Document level interfaces

<code>\RegisterMathEnvironment</code>	<code>\RegisterMathEnvironment [<options>] {<env>}</code>
---------------------------------------	---

Registers the `<env>` as a math environment which should be captured and made available. This is necessary for all top-level math mode environments: low-level errors may result if these are not correct set up. One or more key-value `<options>` may also be given:

arg-spec The argument specification taken by the beginning of the environment; this is used to remove non-mathematical material.

<code>\MathCollectTrue</code>	<code>\MathCollectTrue</code>
<code>\MathCollectFalse</code>	<code>\MathCollectFalse</code>

This activates/deactivates the math collection of the math shift token. See above for cases when this can be useful.

5 Math tagging

5.1 Code requirements

The tagging code has to handle

- the embedding into the surrounding. This means
 - closing and reopening MC-chunks
 - closing and reopening text/P-structures
 - handling interferences of the tagging code with penalties and spacing.
- the actual tagging which means to do some or all of the following tasks:
 - setup content for an associated source file
 - setup content for an associated MathML file
 - setup content for the /Alt key
 - setup content for the /ActualText key
 - setup attributes
 - add associated files
 - add a Formula structure
 - surround elements of the equation with MathML structure elements (currently only luatex with luamml)

5.2 Inline math

The embedding code is added through the tagging sockets

- `math/inline/begin`
- `math/inline/end`

The sockets simply push and pop the MC currently. Without tagging they use the noop-plug.

The actual tagging is in done through the tagging sockets

- `math/inline/formula/begin` This socket takes the math as second argument and its code should output it for typesetting. The `default` plug of the socket calls these three internal sockets for the tagging support:
 - `math/content` This should set up the various content variables (empty variables are ignored by the structure code and so can be used to suppress a setting).
 - `math/struct/begin` This calls `\tag_struct_begin:n`. It should also write the associated files if needed.

- `math/inline/formula/end` This socket ends the formula structure(s). The default plug calls this internal socket:

– `tagssupport/math/struct/end`

5.3 Display math

to be written

5.4 Associated Files

The current code allows the attachment of two types of associated file to the Formula structure: the L^AT_EX source and a MathML representation. Technically both can be attached—AF is an array of file references—in practice there can be problems with PDF consumers: e.g., ngpdf used both and so showed the equation twice (this has been corrected in the newest version) and Foxit seems to see only the first AF in the array (so we attach the MathML as the first file).

The L^AT_EX source can be (and is) attached automatically. It can be suppressed by an option with `math/tex/AF=false`, see below.

The MathML is attached if the files `\jobname-mathml.html` and/or `\jobname-luamml-mathml.html` are found and if they contains a suitable MathML snippet for the current formula. If the files contain more than one suitable snippet (as identified by the hash) the first one is used. `\jobname-luamml-mathml.html` is automatically generated (see below section 5.6) and read after `\jobname-mathml.html`. This means that `\jobname-mathml.html` can contain improved versions of a formula.

The MathML processing can be suppressed globally by emptying the list of mathml files with `math/mathml/sources=`. Locally for a formula `math/mathml/AF=false` can be used.

For a MathML representation a file with such representations must be provided. If the equation is numbered the numbering should be part of the MathML as the Lb1 substructure is ignored if an MathML is used (see https://github.com/foxitsoftware/PDF_UA-2).

The MathML representation is given in a special format. It is meant to be a valid html file that can be viewed in a browser. For this it can start with `<!DOCTYPE html><html>` and end with `</html>` It should have the extension `.html`. The `<mathml>` content is read with special catcodes, so can contain ambersands, hashes, comment chars and unmatched braces such as `<mo>{</mo>`

The file should contain a number of representations in this format:

```
<div>
  <h2>\mml <key></h2>
  <p><source></p>
  <p><hash></p>
  <math <attributes> >
    <mathml>
      </math>
</div>
```

The keywords `<div>`, `<h2>\mml`, `<p>`, `<math>`, `</math>` `</div>` are required as they are used to delimit the arguments by the L^AT_EX code.

`<key>` and `<source>` are only used for debugging, they help to identify the equation referred by this representation. The source should be used correctly escaped `&` and `<` so that it gives valid html!

`<attributes>` is not required either, but can, e.g., contain attributes to improve the display in a browser:

```
<math alttext="\mathbf{G}" class="ltx_Math" display="inline">
```

It can also contain the name space declaration:

```
xmlns="http://www.w3.org/1998/Math/MathML"4
```

By default the code tries at the begin of the document to read a file `\jobname-mathml.html` in the html-format. The file name can be changed with

```
mathml/setfiles={filename1,filename2}
```

(without extension, `html` is added automatically). If there is a list, all files are loaded. If a file doesn't exist it is ignored, only an info is written to the log.

Currently every MathML-snippet from a file is embedded into the PDF, it is not checked first if it is actually used (simply writing everything to the PDF is a bit easier than keeping everything in memory and also means that the snippets are one after the other in the PDF).

As mentioned above the MathML-AF can be suppressed for the equations in a group with `math/mathml/AF=false`, or completely by setting `math/mathml/sources=` in the preamble.

Files embedded in a PDF can be listed in the attachments panel of a PDF viewer. This is probably not so useful for lots of small files (but one could create collections), but as long as PDF editors or viewers don't offer proper support to access the AF it can help so have them there. The MathML are added by default, but the $\LaTeX$ source not. This can be changed with `viewer/pane/mathsourcetrue` (anywhere in the document) and `viewer/pane/mathml=false` (in the preamble, before the external file is read).

5.5 MathML in an attribute

Microsoft Word puts the MathML into a proprietary attribute. For testing purpose this is implemented here too. The writing of such an attribute can be activated globally with `tagging-setup={math/setup=mathml-MS}`. Locally it can be disabled with `math/MS/use=false`. The code makes use of the exported MathML file, so it needs two compilation and will not work if embedding of MathML is disabled.

5.6 Automatic mathml creation with luamml

If `lualatex` and one of the packages `unicode-math` or `lua-unicode-math` is used, the package `luamml` is loaded and this package will then automatically generate the file `\jobname-luamml-mathml.html` with MathML representations of all math formulas. This file is then used in subsequent compilations and works also with `pdflatex`.

The generation of the file can be suppressed (in the preamble) with `math/mathml/luamml/write=false`.

If neither `unicode-math` or `lua-unicode-math` are used, the loading of `luamml` and with it the generation of the file can be forced with `math/mathml/luamml/load=true`

⁴But it is probably not needed and only blows up the PDF.

or `math/mathml/luamml/write=true` but be aware that it is then possible that various symbols are mapped to the wrong Unicode code points.

The package `luamml` is still quite experimental and the output should be checked. The `\jobname-luamml-mathml.html` file may be previewed in a browser although you may need to add additional css or javascript declarations to enable browser support for all MathML constructs.

5.7 Summary of math options

The following options exist to make math more accessible:

ActualText An **ActualText** can be placed on structure elements, but can also be added in the stream on a BDC marker with a **Span** tag (normally an independent marker without an MCID number, it is not clear yet if it can be used on an MC-chunk). The content is a text string, typically one or a few Unicode characters. **ActualText** is meant to replace the content and should only be used on small entities, e.g., to define the semantic or the Unicode code point of a symbol. **ActualText** is not supported by all PDF reader. It is also unknown where it should be used at best (in a structure element, or on an independent Span-BDC) and what happens if it is used in more than one place.

enabled by default? False

how to enable/disable No interface yet. **ActualText** can only be added on the Formula structure element by changing the `tagssupport/math/content` or some other socket. For a BDC marker one can, e.g., use

```
\pdf_string_from_unicode:nn{utf16/hex}{€}\l_tmpa_tl
\pdf_bdc:ee{Span}{/ActualText\l_tmpa_tl}content\pdf_emc:
```

There should be no page break in the `<content>` and the BDC should be correctly nested into tagging, so, e.g., a `\leavevmode` should be issued before the bdc command.

Consumer support in part and in part buggy, needs tests ...

Alt Like **ActualText** the **Alt** key can be used on structure elements and on **Span** in the stream. It should contain a description of the content and is mainly meant for images. PDF/UA-1, which views math formulas as illustrations, mandates the key also for Formula structure elements.

enabled by default? false unless PDF/UA-1 is detected, then it is enabled in the `begindocument/end` hook (this will be reconsidered when it is clear, that the use of **Alt** does not shadow MathML). It can be enabled for all engines and PDF versions.

enable/disable `\tagpdfsetup{math/alt/use}` (local boolean, so can be used on individual equations)

default value A template text (stored in `\l_@@_content_template_tl`) starting with LaTeX formula starts.

user value No interface currently provided. This needs optional arguments or an external setup command. See <https://github.com/latex3/tagging-project/discussions/717>.

source-AF The L^AT_EX-source of the equation can be attached as an associated file with mime-type application/Fx-tex. The AFRelationship is Source. The source is embedded without expansion. This means that targets of references and macros are not resolved. The files are by default not shown in the EmbeddedFiles pane, this can be enabled with `viewer/pane/mathsource=true`. If an A-standard is used, it must be one that allows embedded files, e.g., A-4f.

enabled by default? true for all engines and PDF versions

enable/disable `\tagpdfsetup{math/tex/AF}` (local boolean, so can be used on individual equations)

default value source code including dollars or environment name.

consumer support Currently only ngpdf makes use of it: if there is no MathML it passes the source to mathjax.

luamml The following options make (with lualatex) use of the luamml package. luamml is currently automatically loaded (at the end of the preamble) if unicode-math or lua-unicode-math has been detected. The loading can be forced or suppressed with `\tagpdfsetup{math/mathml/luamml/load=true/false}`.

luamml affects all math, locally it can be stopped with `math/mathml/ignore`, or by using the commands described in the package.

mathml-AF A MathML representation of the equation can be attached to the structure. The configuration possibilities are rather complex as the keys have to control three different aspects: The *generation* of the file with the MathML fragments, the *reading* and *embedding* of the MathML fragments, and the *association* of a MathML fragment to a specific equation.

generation With pdfL^AT_EX MathML fragments can not be generated automatically, but a file with dummy fragments for every equation will be written if `\tagpdfsetup{math/mathml/write-dummy}` is issued in the preamble.

With luaL^AT_EX a file with MathML fragments will be created automatically if the package luamml has been loaded (see above).

reading and embedding By default the code will read and embed MathML from `\jobname-mathml.html` and `\jobname-luamml-mathml.html` in this order and the first fragment with a new hash value will be inserted. The list of sources and their order can be changed with the key `math/mathml/sources`, setting that to an empty value suppresses the loading MathML associated files completely. For efficiency reasons it embeds math fragments directly, there is no check yet if the fragment is actually used.

The files are by default shown in the EmbeddedFiles pane, this can be disabled with `viewer/pane/mathml=false`.

attaching A MathML fragment is currently attached as an associated file to a Formula if the hash of the source matches the hash of the fragment. This is not a perfect test: equations with the same source and so the same hash can have different MathML representation, e.g., if there are references or commands or counters in the equation. This will change in a future version. The attachment can be suppressed locally with `math/mathml/AF=false`. The MathML fragment will still be embedded in the PDF!

TODO: adapt test

mathml structure elements MathML structure elements can be used in PDF 2.0 directly. In PDF 1.7, one could theoretically use them if one declares a role mapping first, (this can be done with `\tagpdfsetup{role/mathml-tags}`) which maps all to `Span`. But such a role mapping breaks the reading and with it the accessibility of the math, as only `Span` is passed to the screen readers and so it is not recommended. Automatic generation of structure elements is only possible with `lualatex`. It requires that the package `luamml` has been loaded.

enabled by default? false

enable/disable `\tagpdfsetup{math/mathml/structelem}` (local setting, so can be used with grouping on individual equations).

consumer support Needs more tests.

6 Debugging

```
\DebugMathOn
\DebugMathOff
\math_debug_on:
\math_debug_off:
```

These commands enable/disable debugging messages.

7 Known current bugs, etc.

7.1 Capture/grabbing problems

1. Incorrect grabbing of $\$$ -math when there is also explicit $\$$ -math within a *text environment* that is itself within the math that should all be grabbed. For example,

$$\$a\begin{minipage}{1cm}\$b\end{minipage}\$$$

would only result in the capture of the tokens “`a\begin_{minipage}{1cm}`”. This can be avoided by an additional brace group:

$$\${a\begin{minipage}{1cm}\$b\end{minipage}}\$$$

2. Similar incorrect grabbing with $\$$ $\$$ also.
3. The grabbing, for all the display environments (and `\` `\]`), needs to deal with nesting: `amsmath` contains code for this.
4. The math can’t contain verbatim and verbatim-like commands. This is nothing new for the `amsmath` environments but changes $\$$ and `\[ \]` and `equation*` (see, e.g., tagging-project issue #30).
5. Begin and end of the math or math environment can not be hidden in commands. For example `>{\$}1<{\$}` in a tabular would lead to errors. Therefore in a tabular a slower token-by-token grabbing is used.

7.2 Fake math

For the current state see 3 above. The text here is mainly kept for history.

In a number of places in L^AT_EX math commands (mainly `$`) is used only for technical reason, e.g., to access a math font, to setup a symbol or to use `\vcenter`.

The code identifies such fake math mostly by making use of the `\m@th` command where two methods are used for the automatic detection:

- After grabbing math content the code checks if the content contains the token `\m@th` and if yes it doesn't call the processor before reinserting the content and perhaps adding tagging code. This method requires that the math can be grabbed (e.g. that the end dollar is visible) and that the `\m@th` is visible. It applies for example in `\@iiiparbox` where the code from `$\vcenter` to `\m@th$` is grabbed and put back. It does not work for example for `tabular` where the dollars and the `\m@th` token are spread around over three commands. `tabular` needs therefore manual intervention. A look in the list of usages (in `usage-of-m@th.md`) justifies this approach. All usages are either not math at all, or related to small elements that probably shouldn't be grabbed and processed on their own.
- `\m@th` is redefined so that it sets the boolean `\l_@@_collected_bool` to true. If `\m@th` is used inside math that has been grabbed this doesn't change much as the boolean is set by the grabbing anyway. For usages outside math the benefit is not so clear: The setting avoids that in L^AT_EX_ε the epsilon is processed as math, but it also prevents that the content of the amsmath command `\boxed` is processed as math. It means that if one wants to reenale math processing inside some (fake) math one has to do it after `\m@th` calls.

7.2.1 Open problems

1. The grabbing code doesn't pass the info that it detected a `\m@th` token. This means that the tagging code has to do the same check (and doesn't do this in all cases yet).
2. Commands are missing to locally disable the grabbing and processing, e.g., to handle `tabular`.
3. It must be checked if setting the boolean in `\m@th` really makes sense or if commands like L^AT_EX_ε should be handled manually.

7.3 Processor

The grabbed math is at first passed to the processor. The processor is not called in a measuring phase (from the amsmath `\ifmeasuring@`) and if the `\m@th` token is detected. It is not quite clear what purpose the processor has. As it is a public interface it can't be used for internal code. And typesetting happens later and the processor can't really change this. Currently it is mostly used for debugging and messages. If the `\m@th` is found the `\l_@@_fakemath_bool` is set, so if the code is changed this must be preserved.

7.4 Other problems

1. The presence of `\m@th` in association with `\ensuremath` does not necessarily indicate fake math. This is because wanting `mathsurround` to be zero is very reasonable and common, *even when the math is genuine* (and hence needs to be collected).
TODO: this claim needs some examples.
2. User-defined environments can create problems; but this area, of new, copied and changed environments, has not yet been developed.

Joseph wrote, inter alia:

My thinking [regarding] `\RegisterMathEnvironment`

- (New) Math environments should not be created-then-patched, but only generated by a [(future)] dedicated command (`\DeclareMathEnvironment`, presumably)
- Math environments created with `ltxcmd` [commands] should not be copied, . . .
- Package authors should be able to manually set up math environments with a public boolean.

7.5 Other ToDos

1. Add (some of) the math display commands that were “lifted from plain”, e.g., `\displaylines` `\eqalign`(??).
2. The `breqn` packages changes catcodes and that isn’t yet covered by our mechanism.
3. `\intertext` is not correctly taken into account by the code splitting multiline math into subformulas.

`\MaybeStop` (temporarily) not executed, as it is unknown on Chris’ system.

8 The Implementation

1 `<*kernel>`

8.1 File declaration

```

2 \ProvidesFile{latex-lab-math.ltx}
3           [\ltxlabmathdate\space
4           v\ltxlabmathversion\space
5           Grab all the math(s) and tag it]
6 <@@=math>
7 \ExplSyntaxOn
```

8.2 Setup

Loading `amsmath` is an absolute requirement: this avoids needing to have conditional definitions and deals with how to define `\[/\]` neatly. The package is loaded before `lua-unicode-math` or at begin document to allow user to load it with options.

```

8 \AddToHook{package/lua-unicode-math/before}{\RequirePackage { amsmath }}
9 \AddToHook{begindocument/before}{ \RequirePackage { amsmath } }
```

8.3 Debugging

```

\g__math_debug_bool
10 \bool_new:N \g__math_debug_bool

\__math_debug:n
\__math_debug_typeout:n
11 \cs_new_eq:NN \__math_debug:n \use_none:n
12 \cs_new_eq:NN \__math_debug_typeout:n \use_none:n

(End of definition for \__math_debug:n and \__math_debug_typeout:n.)

\math_debug_on:
\math_debug_off:
\__math_debug_gset:
13 \cs_new_protected:Npn \math_debug_on:
14 {
15   \bool_gset_true:N \g__math_debug_bool
16   \__math_debug_gset:
17 }

18 \cs_new_protected:Npn \math_debug_off:
19 {
20   \bool_gset_false:N \g__math_debug_bool
21   \__math_debug_gset:
22 }

23 \cs_new_protected:Npn \__math_debug_gset:
24 {
25   \cs_gset_protected:Npe \__math_debug:n ##1
26   { \bool_if:NT \g__math_debug_bool {##1} }
27   \cs_gset_protected:Npe \__math_debug_typeout:n ##1
28   { \bool_if:NT \g__math_debug_bool { \typeout{[Math]~ ==>~ ##1} } }
29 }

(End of definition for \math_debug_on:, \math_debug_off:, and \__math_debug_gset:. These functions
are documented on page 11.)

```

\DebugMathOn If we are debugging blocks we also want to know about template instances, so we turn
\DebugMathOff the debugging for templates as well (for now).

```

30 \cs_new_protected:Npn \DebugMathOn { \math_debug_on: }
31 \cs_new_protected:Npn \DebugMathOff { \math_debug_off: }

32 \DebugMathOff

```

(End of definition for \DebugMathOn and \DebugMathOff. These functions are documented on page 11.)

8.4 Data structures

\l__math_collected_bool Tracks whether math mode material has been collected, which happens inside `amsmath` environments as well as those handled directly here. If true following math will not grab and/or process. See 2 for details.

```

33 \bool_new:N \l__math_collected_bool

```

<code>\l__math_fakemath_bool</code>	Tracks whether math mode material has been identified as fake math during the grabbing phase, which happens currently if the grabbed contents contains the <code>\m@th</code> token.
-------------------------------------	--

```
34 \bool_new:N \l__math_fakemath_bool
```

<code>\l__math_math_content_artifact_bool</code>	Tracks whether math content should be marked as artifact by default.
--	--

```
35 \bool_new:N \l__math_math_content_artifact_bool
```

Change first tl name below: 'env' => 'info'?

Or do we need an extra

storage

`\g__math_grabbed_env_tl` contains the name of the math environment (`math` in the case of inline math, `\g__math_grabbed_math_tl` the math content).

```
36 \tl_new:N \g__math_grabbed_env_tl
```

```
37 \tl_new:N \g__math_grabbed_math_tl
```

<code>\l__math_tmpa_tl</code>	Temporary variables
<code>\l__math_tmpa_skip</code>	
<code>\l__math_tmpa_str</code>	

```
38 \tl_new:N \l__math_tmpa_tl
```

```
39 \skip_new:N \l__math_tmpa_skip
```

```
40 \str_new:N \l__math_tmpa_str
```

<code>\l__math_content_alt_tl</code>	Temporary variables to hold math content that should be used in actual or alt text and stored as AF.
<code>\l__math_content_actual_tl</code>	
<code>\l__math_content_AF_tl</code>	

```
41 \tl_new:N \l__math_content_alt_tl
```

```
42 \tl_new:N \l__math_content_actual_tl
```

```
43 \tl_new:N \l__math_content_AF_source_tl
```

```
44 \tl_new:N \l__math_content_AF_source_tmpa_tl
```

```
45 \tl_new:N \l__math_content_AF_mathml_tl
```

8.5 Math attributes

The attributes `inline` and `display` are declared in `latex-lab-namespace`. The variable will hold the attribute class.

<code>\l__math_attribute_class_tl</code>
--

```
46 \tl_new:N \l__math_attribute_class_tl
```

8.6 Code related to AF

Booleans to handle the options.

```

\l__tag_math_texsource_AF_bool
\l__tag_math_texsource_pane_bool
\l__tag_math_mathml_AF_bool
\g__tag_math_mathml_AF_bool
\l__tag_math_mathml_pane_bool
\l__tag_math_alt_bool
\g__tag_math_luamml_tl
\l__tag_math_MSFT_bool

```

The variable `\g__tag_math_luamml_tl` is initially 0 and the user key can set it to -1 or 1. This allows to distinguish the unset case from a value set by the user.

```

47 \bool_new:N\l__tag_math_texsource_AF_bool
48 \bool_new:N\l__tag_math_texsource_pane_bool
49 \bool_new:N\l__tag_math_mathml_AF_bool
50 \bool_new:N\g__tag_math_mathml_AF_bool
51 \bool_new:N\l__tag_math_mathml_pane_bool
52 \bool_new:N\l__tag_math_alt_bool
53 \bool_new:N\l__tag_math_MSFT_bool
54 \tl_new:N\g__tag_math_luamml_tl
55 \tl_gset:Nn\g__tag_math_luamml_tl {0}

```

```

\g__math_mathml_total_int
\g__math_mathml_int
\g__math_math_total_int
\g__math_mathml_AF_found_int
\g__math_mathml_AF_attached_int

```

`\g__math_mml_total_int` records the MathML fragments read in. `\g__math_mml_int` records the MathML fragments read in with a different hash. `\g__math_AF_total_int` records the number of math structures that try to attach a mathml AF. `\g__math_AF_found_int` records the number of math structures for which a fitting mathml is found. `\g__math_AF_attached_int` records the number of math structures which got a MathML fragment (if mathml-AF are not disabled locally this should be the equal to the previous number).

```

56 \int_new:N\g__math_mathml_total_int
57 \int_new:N\g__math_mathml_int
58 \int_new:N\g__math_math_total_int
59 \int_new:N\g__math_mathml_AF_found_int
60 \int_new:N\g__math_mathml_AF_attached_int

```

```

\l__tag_math_mathml_files_clist

```

A sequence to store the file list for the MathML. We also check the luamml file.

```

61 \clist_new:N\l__tag_math_mathml_files_clist
62 \clist_put_right:Ne\l__tag_math_mathml_files_clist
63 {\c_sys_jobname_str-mathml,\c_sys_jobname_str-luamml-mathml}

```

This is the internal variant of the `\mml` command.

```

\__math_AF_mml:nnnn

```

```

64 \cs_new_protected:Npn \__math_AF_mml:nnnn #1 #2 #3 #4

```

```

65 %1 number, #2 tex source for debugging, #3 hash, #4 mathml
66 {
67   \int_gincr:N \g__math_mathml_total_int

```

MathML with the same hash should be included only once:

```

68   \tl_if_exist:cF { g__math_mathml_#3_tl }
69   {
70     \int_gincr:N \g__math_mathml_int

```

a simple Desc key, take care that it is a valid string!

```

71     \pdfdict_put:nne {l_pdffile/Filespec} {Desc}{(mathml-#1)}
72     \pdffile_embed_stream:nnN {#4}{mathml-#1.xml}\l__math_tmpa_tl

```

not strictly necessary but makes the files visible in the file attachment page

```

73     \bool_if:NT \l__tag_math_mathml_pane_bool
74     {\pdfmanagement_add:nne {Catalog/Names}{EmbeddedFiles}{\l__math_tmpa_tl}}
75     \tl_new:c{g__math_mathml_#3_tl}
76     \tl_gset_eq:cN{g__math_mathml_#3_tl}\l__math_tmpa_tl
77   }
78 }

```

(End of definition for __math_AF_mml:nnnn.)

This is a version of the previous command which writes MSFT-attributes

```

\__math_MSFT_mml:nn
\__math_MSFT_mml_aux:nn

```

```

79 \cs_new_eq:NN \__math_MSFT_mml:nn \use_none:nn
80 \cs_new_protected:Npn \__math_MSFT_mml_aux:nn #1 #2
81 %1 hash, #2 mathml
82 {

```

mathml with the same hash should be included only once:

```

83   \tl_if_exist:cF { g__math_mathml_MSFT_#1_tl }
84   {
85     \pdf_string_from_unicode:nnN{utf16/hex}{#2}\l__math_tmpa_tl
86     \tag_attr_new:ee
87     {MSFT-#1}
88     {/O /MSFT_Office /MSFT_MathML \l__math_tmpa_tl }
89     \tl_new:c {g__math_mathml_MSFT_#1_tl}
90     \tl_gset:cn {g__math_mathml_MSFT_#1_tl} {MSFT-#1}
91   }
92 }

```

(End of definition for __math_MSFT_mml:nn and __math_MSFT_mml_aux:nn.)

The html reader.

```

93 \cs_new_protected:Npn \__math_AF_html_reader:w#1</h2>#2<p>#3</p>#4<p>#5</p>#6<math{
94   \begingroup
95   \char_set_catcode_other:N{
96   \char_set_catcode_other:N\}
97   \char_set_catcode_other:N\#
98   \char_set_catcode_other:N\%
99   \char_set_catcode_other:N\^
100   \__math_AF_html_reader_verb:w{#1}{#3}{#5}<math
101 }

```

```

102 \cs_new_protected:Npn\__math_AF_html_reader_verb:w#1#2#3#4~</div>{
103   \endgroup
104   \__math_AF_mml:nnnn{#1}{#2}{#3}{#4}
105   \__math_MSFT_mml:nn {#3}{#4}
106   }

```

As with luatex we write two files we define a few constants for the shared texts.

```

\c_math_mathml_write_init_tl
\l_math_mathml_write_before_tl
\c_math_mathml_write_after_tl
\c_math_mathml_write_final_tl
107 \tl_const:Nn \c_math_mathml_write_init_tl
108 {
109   <!DOCTYPE~html>
110   \iow_newline:
111   <html~ xmlns="http://www.w3.org/1999/xhtml">
112   \iow_newline:
113   }
114 \tl_new:N \l_math_mathml_write_before_tl
115 \tl_const:Nn \c_math_mathml_write_after_tl
116 {
117   \iow_newline:
118   </div>
119   \iow_newline:
120   }
121 \tl_const:Nn \c_math_mathml_write_final_tl
122 {
123   </html>
124   }

```

(End of definition for `\c_math_mathml_write_init_tl` and others.)

`mathml/write/prepare` (*tag socket*) To prepare the hash and the starting command we use a socket, so that both the dummy and luamml can make use of it.

```

125 \NewTaggingSocket{math/mathml/write/prepare}{0}

```

On (*plug*)

```

126 \NewTaggingSocketPlug{math/mathml/write/prepare}{0n}
127 {
128   \str_set:NV\l_math_tmpa_str\l_math_content_AF_source_tl
129   \str_replace_all:Nnn\l_math_tmpa_str{&}{&amp;}
130   \str_replace_all:Nnn\l_math_tmpa_str{<}{&lt;}
131   \tl_set:Nn \l_math_mathml_write_before_tl
132   {
133     <div>
134     \iow_newline:
135     <h2>\c_backslash_str mml\c_space_tl \int_use:N \g_math_math_total_int </h2>
136     \iow_newline:
137     <p>\l_math_tmpa_str</p>
138     \iow_newline:
139     <p>\l_math_content_hash_tl </p>
140     \iow_newline:
141   }
142 }

```

With luatex we automatically generate MathML with luamml if the package can be loaded and unicode-math or lua-unicode-math are detected. We start the process in the begindocument/end hook so that the reading from a previous compilation can happen before!

For other engines, for future name changes and in case luamml is not loaded we provide some commands

```

143 \cs_new_protected:Npn\__math_provide_luamml_commands:
144 {
145   \providecommand\luamml_flag_structelem:{}
146   \cs_if_free:NT \luamml_structelem:
147   {
148     \cs_set_eq:NN\luamml_structelem:\luamml_flag_structelem:
149   }
150   \providecommand\luamml_flag_process:{}
151   \cs_if_free:NT \luamml_process:
152   {
153     \cs_set_eq:NN\luamml_process:\luamml_flag_process:
154   }
155   \providecommand\luamml_flag_ignore:{}
156   \cs_if_free:NT \luamml_ignore:
157   {
158     \cs_set_eq:NN\luamml_ignore:\luamml_flag_ignore:
159   }
160 }

161 \sys_if_engine luatex:TF
162 {
163   \AddToHook{begindocument/before}
164   {
165     \str_case:on \g__math_luamml_load_tl
166     {
167       { 1 } {
168         \RequirePackage { luamml }
169         \AddToHook{begindocument/end}
170         {
171           \__math_luamml_activate_write:
172         }
173       }
174       {-1 } {
175         \AddToHook{begindocument/end}
176         {
177           \msg_note:nnnn { tag }
178           { luamml-status }{ disabled }{ not~create }
179         }
180       }
181       { 0 }
182       {
183         \bool_lazy_or:nnTF
184         {\cs_if_exist_p:c{ver@unicode-math.sty}}
185         {\cs_if_exist_p:c{ver@lua-unicode-math.sty}}
186         {
187           \RequirePackage { luamml }
188           \AddToHook{begindocument/end}
189           {

```

```

190         \_math_luaamml_activate_write:
191     }
192 }
193 { \msg_warning:nn { tag }{ unicode-math-missing } }
194 }
195 }
196 \_math_provide_luaamml_commands:
197 }
198 }
199 {
200     \AddToHook{begindocument/before}
201     {
202         \_math_provide_luaamml_commands:
203     }
204 }
205 \msg_new:nnn { tag }{ luaamml-status }
206 {
207     luaamml-has-been-#1-and-will-#2-a-MathML-file.
208 }
209
210 \msg_new:nnn { tag }{ unicode-math-missing }
211 {
212     Neither~unicode-math~nor~lua-unicode-math~has~been~loaded.\\
213     luaamml~will~not~create~a~MathML~file.\\
214     To~avoid~this~warning~load~unicode-math~\\
215     or~lua-unicode-math~or~disable~luaamml~with~\\
216     \tl_to_str:n{\tagpdfsetup{math/mathml/luaamml/load=false}}\\
217     or~force~luaamml~with~\\
218     \tl_to_str:n{\tagpdfsetup{math/mathml/luaamml/load=true}}
219 }
220 \cs_new_protected:Npn \_math_luaamml_activate_write:
221 {
222     \bool_if:NT \g__math_luaamml_write_bool
223     {

```

to avoid that nothing is written in the first run, we must activate the sockets:

```

224     \bool_gset_true:N\g__tag_math_mathml_AF_bool
225     \AssignTaggingSocketPlug{math/struct/begin}{mathml-AF}
226     \AssignTaggingSocketPlug{math/struct/end}{mathml-AF}
227     \int_set:Nn \l__luaamml_pretty_int { 7 }
228     \RegisterFamilyMapping\symsymbols{oms}
229     \RegisterFamilyMapping\symletters{oml}
230     \AssignTaggingSocketPlug{math/mathml/write/prepare}{On}
231     \iow_new:N \g__math_luaamml_iow
232     \iow_open:Nn \g__math_luaamml_iow {\c_sys_jobname_str-luaamml-mathml.html}
233     \iow_now:Ne \g__math_luaamml_iow { \c__math_mathml_write_init_tl }
234     \cs_new:Npn \_math_luaamml_output_hook:n ##1
235     {
236         \tl_if_empty:NF \l__math_mathml_write_before_tl
237         {

```

We check here if the current group level is equal to the one stored for the outer math. We only write output if that is the case.

Currently in $\text{\LaTeX}$, the $\text{\math@level}$ is increased for every nested math mode (via $\text{\frozen@everymath}$. However, to make the code below work correctly we undo that in

the case of “fake math”, i.e., in math mode that is only entered to make use of super or subscript positioning of text or for `\vcentering` text, i.e., if it is not really a “math formula”. We therefore provide a special declaration, `\UseMathForPositioningText`, to indicate that the directly following `$` represents such “fake math”.

```

238         \int_compare:nNnT
239         { \@math@level } = { 1 }
240         {
241             \iow_now:Ne \g__math_luamml_iow
242             {
243                 \l__math_mathml_write_before_tl
244                 ##1
245                 \c__math_mathml_write_after_tl
246             }
247         }
248     }
249 }
250 \__luamml_register_output_hook:N \__math_luamml_output_hook:n

```

At the end of the document we must finish and close the file:

```

251 \AddToHook{enddocument/afterlastpage}
252 {
253     \iow_now:Ne \g__math_luamml_iow
254     { \c__math_mathml_write_final_tl }
255     \iow_close:N \g__math_luamml_iow
256 }
257 \msg_note:nnnn { tag }
258 { luamml-status }{ enabled }{ create }
259 }
260 }

```

`\UseMathForPositioningText`

The `\UseMathForPositioningText` command indicates that a directly following `$` is not a real math formula, but that the math mode is only entered to position ordinary text with the help of built in T_EX algorithms normally used for math, e.g., `\vcenter`, super and subscripts.

Signalling that special usage is necessary in the case tagged PDF is produced, because then such math mode should not generate a “formula” structure.

The command requires that it is immediately followed by `$`; anything else will result in a low-level T_EX error. As it is not a user but a developer command, this seems acceptable for the sake of fast runtime execution.

The way it is implemented it can be used in legacy code in front of any `$` that switches to math mode for non-math purposes. If tagging is active it will ensure that this particular math mode content is not treated as math with respect to tagging (though nested math still is). If tagging is inactive the command is simply ignored.

Starting with L^AT_EX 2026-06-01 it is not longer used by the kernel as math is not longer used in tabular and super and subscripts.

```

261 \cs_set_protected:Npn \UseMathForPositioningText $ {

```

Before the math mode is started we ensure that its content is not collected by the grabber.

```

262     \bool_if:NTF \l__math_collected_bool
263     { $ }

```

```

264     {
265         \bool_set_true:N \l__math_collected_bool
266         $

```

Once inside math mode we reenable grabbing, so that any nested math (within text) is grabbed.

```

267         \bool_set_false:N \l__math_collected_bool

```

In addition we decrement the math level (which was automatically incremented when entering math mode) so that this math mode is transparent in `\__math_luamml_activate_write:.`

```

268         \int_decr:N \@math@level
269     }
270 }

```

(End of definition for \UseMathForPositioningText. This function is documented on page ??.)

And now keys to activate/deactivate luamml feature

`\g__math_luamml_load_tl` This variable will be used to suppress the loading of luamml altogether.

```

271 \tl_new:N \g__math_luamml_load_tl
272 \tl_gset:Nn \g__math_luamml_load_tl {0}

```

`\g__math_luamml_write_bool` This variable decides if luamml writes a mathml altogether.

```

273 \bool_new:N \g__math_luamml_write_bool
274 \bool_gset_true:N \g__math_luamml_write_bool

```

`\__math_luamml_ignore:` Internal variants of the luamml commands, that can be remapped if needed.
`\__math_luamml_structelem:`

```

275 \cs_new:Npn \__math_luamml_structelem: {}
276 \cs_new:Npn \__math_luamml_ignore: {}

```

(End of definition for __math_luamml_ignore: and __math_luamml_structelem:.)

```

277 \msg_new:nnn { tag }{ PDF-2.0-recommended }
278 {
279     The~key~#1~will~not~work~properly~with~PDF~#2.\\
280     Switching~to~PDF~2.0~is~recommended.
281 }
282 \keys_define:nn { __tag / setup }
283 {

```

At first a key to suppress the loading altogether

```

284     math/mathml/luamml/load .choice: ,
285     math/mathml/luamml/load/true .code:n = {\tl_gset:Nn \g__math_luamml_load_tl{1}},
286     math/mathml/luamml/load/false .code:n = {\tl_gset:Nn \g__math_luamml_load_tl{-
287     1}},
287     math/mathml/luamml/load .default:n = true,
288     math/mathml/luamml/load .usage:n=preamble,

```

A key to activate math structure elements.

```

289     math/mathml/structelem .choice:,
290     math/mathml/structelem/true .code:n =
291     {
292         \pdf_version_compare:NnT < {2.0}
293         {
294             \msg_warning:nnne { tag }{ PDF-2.0-recommended }
295             { math/mathml/structelem }{ \pdf_version: }
296         }
297         \cs_set:Npn\__math_luaamml_structelem:{\luaamml_structelem:}
298         \cs_set:Npn\__math_luaamml_ignore:{\luaamml_ignore:}
299         \sys_if_engine luatex:T {
300             \bool_set_true:N \l__math_math_content_artifact_bool
301         }
302     },
303     math/mathml/structelem/false .code:n =
304     {
305         \cs_set_eq:NN\__math_luaamml_structelem:\prg_do_nothing:
306         \cs_set_eq:NN\__math_luaamml_ignore:\prg_do_nothing:
307         \bool_set_false:N \l__math_math_content_artifact_bool
308     },
309     math/mathml/structelem .default:n = true,

```

and a key to call the ignore flag. This should only be used locally.

```

310     math/mathml/ignore .code:n = {\luaamml_ignore:},
311
312     math/mathml/luamml/write .choice:,
313     math/mathml/luamml/write/true .code:n =
314     {
315         \tl_gset:Nn \g__math_luaamml_load_tl{1}
316         \bool_gset_true:N \g__math_luaamml_write_bool
317     },
318     math/mathml/luamml/write/false .code:n =
319     {
320         \bool_gset_false:N \g__math_luaamml_write_bool
321     },
322     math/mathml/luamml/write .default:n = true,
323     math/mathml/luamml/write .usage:n=preamble,

```

alias keys for compatibility

```

323     math/mathml/luamml .bool_gset:N = \g__math_luaamml_write_bool,
324     math/mathml/luamml .usage:n=preamble
325 }

```

`math/mathml/write` (*tag socket*) This writes a html-dummy with the hash and the math content. This should be optional, so it uses a socket that can be disabled

```

326 \NewTaggingSocket{math/mathml/write}{0}

```

On (*plug*)

```

327 \NewTaggingSocketPlug{math/mathml/write}{0n}
328 {
329     \iow_now:Ne \g__math_writedummy_iow
330     {
331         \l__math_mathml_write_before_tl

```

```

332     <math~ xmlns="http://www.w3.org/1998/Math/MathML"></math>
333     \c__math_mathml_write_after_tl
334   }
335 }

```

And now a key to activate the socket.

```

336 \keys_define:nn { __tag / setup }
337 {
338   math/mathml/write-dummy .code:n =
339   {
340     \bool_gset_true:N \g__tag_math_mathml_AF_bool
341     \tl_if_exist:NF\g__math_writedummy_iow
342     {
343       \iow_new:N \g__math_writedummy_iow
344       \iow_open:Nn \g__math_writedummy_iow
345       {
346         \c_sys_jobname_str-mathml-dummy.html
347       }
348       \iow_now:Ne \g__math_writedummy_iow
349       {
350         \c__math_mathml_write_init_tl
351       }
352       \AssignTaggingSocketPlug{math/mathml/write/prepare}{On}
353       \AssignTaggingSocketPlug{math/mathml/write}{On}
354       \AddToHook{enddocument/afterlastpage}
355       {
356         \iow_now:Ne \g__math_writedummy_iow
357         { \c__math_mathml_write_final_tl }
358         \iow_close:N \g__math_writedummy_iow
359       }
360     }
361   },
362   math/mathml/write-dummy .usage:n=preamble
363 }

```

_math_AF_process_mathml_files:

```

364 \box_new:N\l__math_tmpa_box
365 \cs_new_protected:Npn \__math_AF_process_mathml_files:
366 {
367   \hbox_set:Nn \l__math_tmpa_box
368   {
369     \pdfdict_put:nnn { l_pdffile/Filespec }{AFRelationship} { /Supplement }
370     \pdfdict_put:nne
371     { l_pdffile }{Subtype}
372     { \pdf_name_from_unicode_e:n{application/mathml+xml} }
373     \char_set_catcode_other:N \#
374     \cs_set_eq:NN\mml \__math_AF_html_reader:w
375     \clist_map_inline:Nn \l__tag_math_mathml_files_clist
376     {
377       \file_if_exist:nTF {##1.html}
378       {
379         \typeout{Info:~reading~mathml~file~##1}
380         \file_input:n {##1.html}
381         \bool_gset_true:N\g__tag_math_mathml_AF_bool

```

```

382         }
383         {
384             \typeout{Info:~mathml~file~##1~does~not~exist}%info message
385         }
386     }
387 }
388 \box_clear:N \l__math_tmpa_box
389 \bool_if:NT\g__tag_math_mathml_AF_bool
390 {
391     \typeout{Info:~Activating~mathml~support}
392     \AssignTaggingSocketPlug{math/struct/begin}{mathml-AF}
393     \AssignTaggingSocketPlug{math/struct/end}{mathml-AF}
394     \AddToHook{enddocument/info}
395     {
396         \iow_term:n{MathML~statistic}
397         \iow_term:n{=====}
398         \iow_term:e{==>~\int_use:N\g__math_mathml_total_int\c_space_tl
399         MathML~fragments~read}
400         \iow_term:e{==>~\int_use:N\g__math_mathml_int\c_space_tl
401         different~MathML~fragments}
402         \iow_term:e{==>~\int_use:N\g__math_math_total_int\c_space_tl
403         math~fragments~found}
404         \iow_term:e{==>~\int_use:N\g__math_mathml_AF_found_int\c_space_tl
405         fitting~MathML~AF~found}
406         \iow_term:e{==>~\int_use:N\g__math_mathml_AF_attached_int\c_space_tl
407         MathML~AF~attached}
408     }
409 }
410 }
411 \AddToHook{begindocument}{\__math_AF_process_mathml_files:}

```

(End of definition for `\__math_AF_process_mathml_files:.`)

8.7 Mathstyle detection

In some cases we need to detect the mathstyle used in a `\mathchoice` command and to disable/enable tagging in the unused branches. This is currently only used in the `amstext` command `\text` but is perhaps also needed in other cases, so we create a general command.

```

\l__math_mathstyle_int
\g__math_mathchoice_int
mathstyle
412 \int_new:N \l__math_mathstyle_int
413 \int_new:N \g__math_mathchoice_int
414 \property_new:nnnn{mathstyle}{now}{-1}{\int_use:N \l__math_mathstyle_int }

```

(End of definition for `\l__math_mathstyle_int`, `\g__math_mathchoice_int`, and `mathstyle`.)

For now internal, but perhaps will need a public version. The command should be used in every branch of a `\mathchoice` (with the correct mathstyle number) and with an unique label (which should be the same in every branch). `\g__math_mathchoice_int` can be, e.g., increased before the mathchoice and then used.

```

\__math_tag_if_mathstyle:nn
415 \cs_new_protected:Npn \__math_tag_if_mathstyle:nn #1 #2

```

```

416 %\#1 refers to label
417 %\#2 is a number for the mathstyle (typically 0,2,4,6)
418 {
419   \int_set:Nn \l__math_mathstyle_int {#2}
420   \property_record:nn {#1} { mathstyle }
421   \int_compare:nNnTF { \property_ref:nn {#1}{ mathstyle} } = { #2 }
422     { \tag_resume:n{\mathchoice} }{ \tag_suspend:n{\mathchoice} }
423 }
424 \cs_generate_variant:Nn \__math_tag_if_mathstyle:nn {en}

```

(End of definition for `\__math_tag_if_mathstyle:nn`.)

8.8 Tagging options

```

425 \keys_define:nn { __tag / setup }
426 {
427   math/mathml/sources .clist_set:N = \l__tag_math_mathml_files_clist,
428   math/alt/use        .bool_set:N = \l__tag_math_alt_bool,
429   math/MS/use         .bool_set:N = \l__tag_math_MSFT_bool,
430   viewer/pane/mathml  .bool_set:N = \l__tag_math_mathml_pane_bool,
431   viewer/pane/mathml  .initial:n = true,
432   viewer/pane/mathsource .bool_set:N = \l__tag_math_texsource_pane_bool,
433   math/mathml/AF .bool_set:N = \l__tag_math_mathml_AF_bool,
434   math/mathml/AF .initial:n = true,
435   math/tex/AF .bool_set:N = \l__tag_math_texsource_AF_bool,
436   math/tex/AF .initial:n = true
437 }

```

alt is required for pdf/UA-1. TODO: l3pdfmeta should support this test.

```

438 \AddToHook{begindocument/end}
439 {
440   \str_if_eq:eeT
441     {1}
442     {
443       \exp_last_unbraced:Ne\use_i:nn
444         {\GetDocumentProperty{document/pdfstandard-UA}}
445       \c_empty_tl\c_empty_tl
446     }
447     {
448       \bool_if:NF \l__tag_math_alt_bool
449       {
450         \typeout{PDF/UA-1~detected.~Enabling~alt~text~on~Formula}
451       }
452       \bool_set_true:N\l__tag_math_alt_bool
453     }
454 }

```

8.8.1 Meta keys

The `math/setup` key accepts a list with the values `mathml-SE`, `mathml-AF` and `tex-AF`. It is a fast way to set the main option. It at first disables them all, to get a clean state.

```

455 \keys_define:nn {__tag / setup}
456 {
457   math/setup .code:n =

```

```

458 {
459   %deactivate loading of luamml
460   \tl_gset:Nn \g__math_luamml_load_tl{-1}
461   \keys_set:nn {__tag / setup}
462   {
463     %deactivate tex source AF
464     math/tex/AF = false,
465     %deactivate reading of mathml-AF
466     math/mathml/sources=,
467     math/mathml/AF=false,
468     %deactivate structelem
469     math/mathml/structelem=false,
470     %handle value
471   }
472   \clist_map_inline:nn { #1}
473   {
474     \keys_set:nn {__tag / setup}{math/__setup/##1}
475   }
476 },
477 math/__setup / mathml-SE .code:n =
478 {
479   \tl_gset:Nn \g__math_luamml_load_tl{1}
480   \keys_set:nn {__tag / setup}
481   {
482     math/mathml/structelem=true
483   }
484 },
485 math/__setup / mathml-AF .code:n =
486 {
487   \tl_gset:Nn \g__math_luamml_load_tl{1}
488   \clist_put_right:Ne\l__tag_math_mathml_files_clist
489   {\c_sys_jobname_str-mathml,\c_sys_jobname_str-luamml-mathml}
490   \keys_set:nn {__tag / setup}
491   {
492     math/mathml/AF=true
493   }
494 },
495 math/__setup / mathml-MS .code:n =
496 {
497   \tl_gset:Nn \g__math_luamml_load_tl{1}
498   \cs_set_eq:NN\__math_MSFT_mml:nn\__math_MSFT_mml_aux:nn
499   \bool_set_true:N\l__tag_math_MSFT_bool
500   \clist_put_right:Ne\l__tag_math_mathml_files_clist
501   {\c_sys_jobname_str-mathml,\c_sys_jobname_str-luamml-mathml}
502 },
503 math/__setup / tex-AF .code:n =
504 {
505   \keys_set:nn {__tag / setup}
506   {
507     math/tex/AF =true
508   }
509 },
510 }

```

8.9 Sockets

8.9.1 Main inline math sockets

`math/inline/begin` (*tag socket*) These sockets are already declared in `ltagging` and only documented here. The first two sockets are meant to embed inline math into the surrounding (so to close/reopen, e.g., MC-chunks). The other two implement the actual formula structure. The formula sockets are despite their naming not symmetric: the begin socket is issued after the math has started, while the end socket is after the math!

```
511 %\NewTaggingSocket{math/inline/begin}{0}
512 %\NewTaggingSocket{math/inline/end}{0}
513 %\NewTaggingSocket{math/inline/formula/begin}{2} %
514 %\NewTaggingSocket{math/inline/formula/end}{0}
```

MC (*plug*)

```
515 \NewTaggingSocketPlug
516   {math/inline/begin}
517   {MC}
518   {\tag_mc_end_push:}
519 \NewTaggingSocketPlug
520   {math/inline/end}
521   {MC}
522   {\tag_mc_begin_pop:n{}}
```

We probably will want to test different tagging recipes.

default (*plug*)

```
523 \NewTaggingSocketPlug
524   {math/inline/formula/begin}
525   {default}

526 { \tagpdfparaOff
527   \__math_lua_mml_structelem:
528   \tag_socket_use:n{math/content}
529   \tag_socket_use:n{math/struct/begin}
530   #2
531   \tag_socket_use:n{math/end}
532 }
533 \NewTaggingSocketPlug
534   {math/inline/formula/end}
535   {default}
536 {
537   \tag_socket_use:n{math/struct/end}
538 }
```

8.9.2 Main display math sockets

`math/display/begin` (*tag socket*) These sockets are already declared in `ltagging` and only documented here. The first two sockets are meant to embed display math into the surrounding (so to close/reopen, e.g., MC-chunks and P-structure). The other two implement the actual formula structure. The formula sockets are despite their naming not symmetric: the begin socket is issued after the math has started, while the end socket is after the math!

```
539 %\NewTaggingSocket{math/display/begin}{0}
```

```

540 %\NewTaggingSocket{math/display/end}{0}
541 %\NewTaggingSocket{math/display/formula/begin}{2} %
542 %\NewTaggingSocket{math/display/formula/end}{0}

```

default (*plug*)

```

543 \NewTaggingSocketPlug
544   {math/display/begin}
545   {default}
546   {
547     \UseTaggingSocket{para/textblock/end}
548   }
549 \NewTaggingSocketPlug
550   {math/display/end}
551   {default}
552   {
553   }

```

default (*plug*)

```

554 \NewTaggingSocketPlug
555   {math/display/formula/begin}
556   {default}
557   {
558     \tagpdfparaOff
559     \__math_luauml_structelem:
560     \tag_socket_use:n{math/content}
561     \tag_socket_use:n{math/struct/begin}
562     #2
563     \tag_socket_use:n{math/end}
564   }
565 \NewTaggingSocketPlug
566   {math/display/formula/end}
567   {default}
568   {
569     \tag_socket_use:n{math/struct/end}
570   }

```

8.9.3 Sockets plugs for tags (labels)

$\hdisplay/tag/begin$ (*tag socket*) These sockets are already declared in `ltagging` and only documented here. These sockets
 $\mathdisplay/tag/end$ (*tag socket*) are used in `\maketag__math@` to tag labels as `Lbl`. `luamml` changes the plug to move the `Lbl` into the math structure with an intent.

```

571 %\NewTaggingSocket{math/display/tag/begin}{0}
572 %\NewTaggingSocket{math/display/tag/end}{0}

```

default (*plug*)

```

573 \NewTaggingSocketPlug
574   {math/display/tag/begin}
575   {default}
576   {
577     \tag_mc_end:
578     \tag_struct_begin:n {tag=\UseStructureName{math/label}}
579     \tag_mc_begin:n {}
580   }

```

```

581 \NewTaggingSocketPlug
582   {math/display/tag/end}
583   {default}
584   {
585     \tag_mc_end:
586     \tag_struct_end:
587     \tag_mc_begin:n{ }
588   }
589 \AssignTaggingSocketPlug{math/display/tag/begin}{default}
590 \AssignTaggingSocketPlug{math/display/tag/end}{default}

```

8.9.4 Internal sockets

`\l__math_content_template_tl`

The default text used as alt or actual text.

```

591 \tl_new:N\l__math_content_template_tl
592 \tl_set:Nn \l__math_content_template_tl
593   {
594     LaTeX~ formula~ starts~
595     \exp_not:N\begin{\g__math_grabbed_env_tl}
596     \c_space_tl
597     \exp_not:V\g__math_grabbed_math_tl
598     \c_space_tl
599     \exp_not:N\end{\g__math_grabbed_env_tl}
600     \c_space_tl LaTeX~ formula~ ends~
601   }

```

`\l__math_texsource_template_tl`

The default text used as texsource

```

602 \tl_new:N\l__math_texsource_template_tl
603 \tl_const:Nn\c__math_inline_env_tl {math}
604 \tl_set:Nn \l__math_texsource_template_tl
605   {
606     \tl_if_eq:NNTF\g__math_grabbed_env_tl\c__math_inline_env_tl
607     {
608       $
609       \exp_not:V\g__math_grabbed_math_tl
610       $
611     }
612     {
613       \exp_not:N\begin{\g__math_grabbed_env_tl}
614       \exp_not:V\g__math_grabbed_math_tl
615       \exp_not:N\end{\g__math_grabbed_env_tl}
616     }
617   }

```

math/content (*tag socket*) The math content is stored in associated files and used for actual and alternative text. As the exact text is still unclear we use a socket to be able to test variants. The socket

should set all four `tl` vars above, if needed to identical values. It can use the two variables `\g__math_grabbed_env_tl` and `\g__math_grabbed_math_tl`

```
618 \NewTaggingSocket{math/content}{0}
```

Some default sockets to set the contents. TODO: think about naming convention. TODO: think how this should organized so that one has options to change from the outside and so that there are less repetitions.

`actual+source (plug)`

```
619 \NewTaggingSocketPlug
620 {math/content}
621 {actual+source}
622 {
623   \tl_set:N\l__math_content_actual_tl
624   {
625     \l__math_content_template_tl
626   }
627   \tl_set:N\l__math_content_AF_source_tl
628   {
629     \l__math_texsource_template_tl
630   }
631   \tl_set:Nn \l__math_content_AF_mathml_tl {}
632   \tl_set:Nn \l__math_content_alt_tl {}
633 }
```

`alt+source (plug)`

```
634 \NewTaggingSocketPlug
635 {math/content}
636 {alt+source}
637 {
638   \tl_set:N\l__math_content_alt_tl
639   {
640     \l__math_content_template_tl
641   }
642   \tl_set:N\l__math_content_AF_source_tl
643   {
644     \l__math_texsource_template_tl
645   }
646   \tl_set:Nn \l__math_content_AF_mathml_tl {}
647   \tl_set:Nn \l__math_content_actual_tl {}
648 }
```

```
649 \AssignTaggingSocketPlug{math/content}{alt+source}
```

`math/struct/begin (tag socket)` For the main structure we use a socket too. This allows, e.g., to create a special one
`math/struct/end (tag socket)` for luamml which setups additional objects. The begin socket can use the two variables `\g__math_grabbed_env_tl` and `\g__math_grabbed_math_tl`

```
650 \NewTaggingSocket{math/struct/begin}{0}
651 \NewTaggingSocket{math/struct/end}{0}
```

`default (plug)` TODO: think about some naming convention ...

```
652 \NewTaggingSocketPlug
653 {math/struct/begin}
```

```

654 {default}
655 {
656   \bool_if:NTF\l__tag_math_texsource_AF_bool
657   { \tl_set_eq:NN \l__math_content_AF_source_tmpa_tl \l__math_content_AF_source_tl }
658   { \tl_clear:N \l__math_content_AF_source_tmpa_tl }
659   \tl_if_eq:NnTF\g__math_grabbed_env_tl {math}
660   {
661     \tl_set:Nn\l__math_attribute_class_tl{inline}
662   }
663   {
664     \tl_set:Nn\l__math_attribute_class_tl{display}
665   }
666   \bool_if:NF\l__tag_math_alt_bool
667   { \tl_set:Nn \l__math_content_alt_tl{} }
668   \tag_struct_begin:n
669   {
670     tag=\UseStructureName{math/\l__math_attribute_class_tl},
671     attribute-class=\l__math_attribute_class_tl,
672     texsource      = \l__math_content_AF_source_tmpa_tl,
673     title-o        = \g__math_grabbed_env_tl,
674     actualtext     = \l__math_content_actual_tl,
675     alt            = \l__math_content_alt_tl
676   }
677
678   \__math_debug_typeout:n{grabbed-math=\meaning\g__math_grabbed_math_tl}
679   \bool_if:NTF \l__math_math_content_artifact_bool
680   {
681     \tag_mc_begin:n{ artifact }
682   } {
683     \tag_mc_begin:n{}
684   }
685   \NewTaggingSocketPlug
686   {math/struct/end}
687   {default}
688   { \tag_mc_end: \tag_struct_end: }
689
690   \AssignTaggingSocketPlug{math/struct/begin}{default}
691   \AssignTaggingSocketPlug{math/struct/end}{default}

```

mathml-AF (plug) This socket tries to add a mathml-AF to formula. It is activated if a mathml.html has been found and loaded. As it disturbs the reading of the AF it currently deactivates the /Alt key, unless it has been reenabled with `math/alt/use=true`

```

692 \cs_generate_variant:Nn \str_mdfive_hash:n {o}
693 \tl_new:N\l__math_content_hash_tl

```

we need to save the grabbed math:

```

694 \tl_new:N\l__math_grabbed_math_tl

```

the socket definition

```

695 \NewTaggingSocketPlug
696 {math/struct/begin}
697 {mathml-AF}
698 {

```

```

699 \int_gincr:N\g__math_math_total_int
700 \tl_set:N\l__math_content_hash_tl
701   {\str_mdfive_hash:o { \l__math_content_AF_source_tl }}
702 \tl_set_eq:NN\l__math_grabbed_math_tl\g__math_grabbed_math_tl
703 \tl_if_eq:NnTF\g__math_grabbed_env_tl {math}
704   {
705     \tl_set:Nn\l__math_attribute_class_tl{inline}
706   }
707   {
708     \tl_set:Nn\l__math_attribute_class_tl{display}
709   }
710 \bool_if:NF\l__tag_math_alt_bool
711   { \tl_set:Nn \l__math_content_alt_tl{} }

```

debugging option. TODO: hide in debug key.

```

712 \tl_if_exist:cTF { g__math_mathml_ \l__math_content_hash_tl _tl }
713   {
714     \int_gincr:N\g__math_mathml_AF_found_int
715     \bool_if:NTF \l__tag_math_mathml_AF_bool
716       {
717         \int_gincr:N\g__math_mathml_AF_attached_int
718
719         \__math_debug_typeout:n {Inserting~mathml~with~Hash~\l__math_content_hash_tl}
720       }
721       {
722         \__math_debug_typeout:n {Ignoring~mathml~with~Hash~\l__math_content_hash_tl}
723       }
724   }
725   {
726     \bool_if:NT \l__tag_math_mathml_AF_bool
727       {
728         \typeout {WARNING:~mathml~missing~for~hash~\l__math_content_hash_tl}
729       }
730   }
731 \tag_socket_use:n {math/mathml/write/prepare}
732 \tag_socket_use:n {math/mathml/write} % write hash if request
733 \bool_if:NTF\l__tag_math_texsource_AF_bool
734   { \tl_set_eq:NN \l__math_content_AF_source_tmpa_tl \l__math_content_AF_source_tl }
735   { \tl_clear:N \l__math_content_AF_source_tmpa_tl }
736 \tag_struct_begin:n
737   {
738     tag=\UseStructureName{math/\l__math_attribute_class_tl},
739     attribute-class=\l__math_attribute_class_tl, %
740     attribute      =
741       \bool_if:NT \l__tag_math_MSFT_bool
742       {
743         \cs_if_exist_use:c {g__math_mathml_MSFT_ \l__math_content_hash_tl _tl}
744       },
745     AFref          =
746       \bool_if:NT\l__tag_math_mathml_AF_bool
747       {
748         \cs_if_exist_use:c {g__math_mathml_ \l__math_content_hash_tl _tl}
749       },
750     texsource      = \l__math_content_AF_source_tmpa_tl, % should be after mathml AF!

```

```

750         title-o      = \g__math_grabbed_env_tl,      %
751         alt          = \l__math_content_alt_tl
752     }
753     \__math_debug_typeout:n {grabbed~math=\meaning\g__math_grabbed_math_tl}
754     \tag_mc_begin:n{ }
755 }

```

not really needed but looks more symmetric:

```

756 \NewTaggingSocketPlug
757 {math/struct/end}
758 {mathml-AF}
759 {
760     \tag_mc_end:
761     \tag_struct_end:
762 }

```

math/end (*tag socket*) A socket used at the end of the math (before the closing dollar(s)) which can, e.g., set a flag for luamml.

```

763 \NewTaggingSocket{math/end}{0}

764 \AssignTaggingSocketPlug{math/inline/begin}{MC}
765 \AssignTaggingSocketPlug{math/inline/end}{MC}
766 \AssignTaggingSocketPlug{math/inline/formula/begin}{default}
767 \AssignTaggingSocketPlug{math/inline/formula/end}{default}
768 \AssignTaggingSocketPlug{math/display/begin}{default}
769 \AssignTaggingSocketPlug{math/display/end}{default}
770 \AssignTaggingSocketPlug{math/display/formula/begin}{default}
771 \AssignTaggingSocketPlug{math/display/formula/end}{default}

```

8.10 Interface commands

```

\__math_process:nn A no-op place-holder; the internal wrapper means that it does not need to be concerned
\__math_process:Vn with internals.
\__math_process_auxi:nn
\__math_process_auxii:nn
772 \newif\ifmeasuring@
773 \cs_new_protected:Npn \__math_process:nn #1#2
774 {
775     \legacy_if:nF { measuring@ }
776     {
777         \tl_if_in:nnTF {#2} { \m@th }
778         { \bool_set_true:N\l__math_fakemath_bool }
779         { \tl_trim_spaces_apply:nN {#2} \__math_process_auxi:nn {#1} }
780     }
781 }
782 \cs_generate_variant:Nn \__math_process:nn { V }
783 \cs_new_protected:Npn \__math_process_auxi:nn #1#2
784 {
785     \tl_gset:Nn \g__math_grabbed_env_tl {#2}
786     \tl_gset:Nn \g__math_grabbed_math_tl {#1}
787     \__math_process_auxii:nn {#2} {#1}
788 }
789 \cs_new_protected:Npn \__math_process_auxii:nn #1#2 { }

```

(End of definition for __math_process:nn, __math_process_auxi:nn, and __math_process_auxii:nn.)

`\math_processor:n` A simple installer

```
790 \cs_new_protected:Npn \math_processor:n #1
791 { \cs_set_protected:Npn \__math_process_auxii:nn ##1##2 {#1} }
```

(End of definition for `\math_processor:n`. This function is documented on page 5.)

8.11 Content grabbing

`\MathCollectTrue`
`\MathCollectFalse`

```
792 \cs_set_protected:Npn \MathCollectTrue{\bool_set_false:N \l__math_collected_bool}
793 \cs_set_protected:Npn \MathCollectFalse{\bool_set_true:N \l__math_collected_bool}
```

(End of definition for `\MathCollectTrue` and `\MathCollectFalse`. These functions are documented on page 6.)

`\__math_grab_dollar:w`
`\__math_grab:n`

Top-level function to handle grabbing of inline math mode delimited by `$` tokens. We provide two different ways to do that: a token-by-token one that can be used everywhere, and a fast delimited one that does not work anywhere that the end `$` token may be hidden, most obviously in tabulars. The function here is therefore set up as a variable starting point.

```
794 \cs_new_protected:Npn \__math_grab_dollar:w { \__math_grab_dollar_delim:w }
```

After grabbing inline math material, there is again common processing independent of mechanism of collection.

```
795 \cs_new_protected:Npn \__math_grab:n #1
796 {
```

We need to do processing first as this picks up “fake” math mode: that information is needed below. The captured material *could* contain `&` tokens, so we ensure that these do no cause an issue if we are inside an `\halign`.

```
797 \group_align_safe_begin:
798 \__math_process:nn { math } {#1}
799 \group_align_safe_end:
```

We do not want math tagging in fakemath or when measuring, We also do not want math tagging if tagging has been suspended.

```
800 \bool_lazy_any:nTF
801 {
802   {\legacy_if_p:n { measuring@ }}
803   { \l__math_fakemath_bool }
804   { \tl_if_blank_p:n {#1} }
805 }
806 {
807   \__math_luamml_ignore:
808   #1 $ % $
809 }
810 {
811   \tag_socket_use:n {math/inline/begin} %end P-MC
```

We do no use a tagging socket here, so that the argument (the math) is not lost, tagging-project issue 661.

```
812 \tag_socket_use:nnn {math/inline/formula/begin}{-}{#1}
813 $ % $
814 \tag_socket_use:n {math/inline/formula/end}
```

```

815         \tag_socket_use:n {math/inline/end} % restart P-MC
816     }
817 }

```

(End of definition for __math_grab_dollar:w and __math_grab:n.)

__math_grab_dollar_delim:w Grab up to a single \$, for inline math mode, suppressing any processing if the token is \m@th found in the content.

```

818 \cs_new_protected:Npn \__math_grab_dollar_delim:w #1 $ % $
819 { \__math_grab:n {#1} }

```

(End of definition for __math_grab_dollar_delim:w.)

__math_grab_dollardollar:w And for the classical T_EX display structure.

```

820 \cs_new_protected:Npn \__math_grab_dollardollar:w #1 $$ {
821     \tl_if_blank:nF {#1}
822     {
823         \__math_process:nn { equation* } {#1}
824         \tag_socket_use:n {math/display/begin}
825         \tag_socket_use:nn{math/display/formula/begin}-{#1}
826     }

```

Prepare to finish the display math formula and let T_EX do its job; then regain control after the formula has been contributed to the page, in order to add the tagging followed by the correct penalty and skip.

```

827     \__math_prepare_display_end:
828     $$
829 }

```

(End of definition for __math_grab_dollardollar:w.)

__math_grab_inline_delim:w Collect inline math content and deal with the need to move to math mode.

```

830 \cs_new_protected:Npn \__math_grab_inline_delim:w % \ (
831     #1 \)
832     {
833         \tl_if_blank:nF {#1}
834         {
835             $ #1 $
836         }
837         \bool_set_false:N \l__math_collected_bool
838     }

```

(End of definition for __math_grab_inline_delim:w.)

__math_grab_inline:w See @@_grab_dollar:w.

```

839 \cs_new_protected:Npn \__math_grab_inline:w { \__math_grab_inline_delim:w }

```

(End of definition for __math_grab_inline:w.)

__math_grab_eqn:w For the most common use of \[/\]: turn into an environment.

```

840 \cs_new_protected:Npn \__math_grab_eqn:w % \[
841     #1 \]
842     {
843         % \typeout{collected? = \bool_if:NTF \l__math_collected_bool {true}{false}}
844         \begin { equation* } #1 \end { equation* }
845     }

```

(End of definition for __math_grab_eqn:w.)

8.12 Token-by-token inline grabbing

Grabbing inline math token-by-token is more involved. The mechanism here is essentially a simplified version of that originally seen in `collcell` and refined in `siunitx`. We make use of the fact that in math mode spaces are ignored, so we have to deal with only N-type tokens and groups. Furthermore, there is no need to look inside groups, so the only special cases are a small selection of N-type tokens.

`\l__math_grabbed_tl` For collection of the material piecewise.

```
846 \tl_new:N \l__math_grabbed_tl
```

`\l__math_grab_env_int` Needed to count up the number of nested environments encountered.

```
847 \int_new:N \l__math_grab_env_int
```

`\__math_grab_loop_init:` The lead-off here establishes a group: we need that as we will have to be careful in the way `\cr` is handled and ensure this is only manipulated whilst grabbing. The main loop is then started.

```
848 \cs_new_protected:Npn \__math_grab_loop_init:
849 {
850   \group_begin:
851   \tl_clear:N \l__math_grabbed_tl
852   \__math_grab_loop:
853 }
854 \cs_new_protected:Npn \__math_grab_loop:
855 {
856   \peek_remove_spaces:n
857   {
858     \peek_meaning:NTF \c_group_begin_token
859     { \__math_grab_loop_group:n }
860     { \__math_grab_loop_token:N }
861   }
862 }
```

(End of definition for `\__math_grab_loop_init:` and `\__math_grab_loop:`.)

`\__math_grab_loop_group:n` Handling of grabbed groups is pretty easy.

```
\__math_grab_loop_store:n
863 \cs_new_protected:Npn \__math_grab_loop_group:n #1
864 { \__math_grab_loop_store:n { {#1} } }
865 \cs_new_protected:Npn \__math_grab_loop_store:n #1
866 {
867   \tl_put_right:Nn \l__math_grabbed_tl {#1}
868   \__math_grab_loop:
869 }
```

(End of definition for `\__math_grab_loop_group:n` and `\__math_grab_loop_store:n`.)

`\__math_grab_loop_token:N` Filter out the special cases: for performance reasons, use a hash table approach rather than a loop (*cf.* `collcell`). To handle the case of a nested array or similar, an alignment safe group is added around the first occurrence of a nested environment.

```
\__math_grab_loop_$:
\__math_grab_loop_\:
\__math_grab_loop_\begin:
\__math_grab_loop_\end:
\__math_grab_loop_\ignorespaces:
\__math_grab_loop_\unskip:
\__math_grab_loop_\textonly@unskip:
\__math_grab_loop_\end:n
870 \cs_new_protected:Npn \__math_grab_loop_token:N #1
```

```

871 {
872   \cs_if_exist_use:cF
873     { __math_grab_loop_ \token_to_str:N #1 : }
874     { \__math_grab_loop_store:n {#1} }
875 }
876 \cs_new_protected:cpn { __math_grab_loop_ \token_to_str:N $ : }
877 { \__math_grab_loop_end: }
878 \cs_new_protected:cpn { __math_grab_loop_ \token_to_str:N \) : }
879 { \__math_grab_loop_end: }
880 \cs_new_protected:cpn { __math_grab_loop_ \token_to_str:N \\ : }
881 {
882   \int_compare:nNnTF \l__math_grab_env_int = 0
883     { \__math_grab_loop_newline: }
884     { \__math_grab_loop_store:n { \\ } }
885 }

```

In contrast to `colcell`, nesting is tracked by counting `\begin/\end` pairs: this is needed in case there is a tabular-like construct containing `\\` inside a cell. As a result, the end-of-tabular can be detected without checking the name argument: if `\end` is encountered at nesting level 0, we've hit the end of a cell. In that case, end the row and leave the environment to clean up.

```

886 \cs_new_protected:cpn { __math_grab_loop_ \token_to_str:N \begin : }
887 {
888   \int_incr:N \l__math_grab_env_int
889   \int_compare:nNnT \l__math_grab_env_int = 1
890     { \group_align_safe_begin: }
891   \__math_grab_loop_store:n { \begin }
892 }
893 \cs_new_protected:cpe { __math_grab_loop_ \token_to_str:N \end : }
894 {
895   \exp_not:N \int_compare:nNnTF \exp_not:N \l__math_grab_env_int = 0
896     {
897       \exp_not:N \__math_grab_loop_newline:
898       \exp_not:N \end
899     }
900     { \exp_not:c { __math_grab_loop_ \token_to_str:N \end :n } }
901 }
902 \cs_new_protected:cpn { __math_grab_loop_ \token_to_str:N \end :n } #1
903 {
904   \int_decr:N \l__math_grab_env_int
905   \tl_put_right:Nn \l__math_grabbed_tl { \end {#1} }
906   \int_compare:nNnT \l__math_grab_env_int = 0
907     { \group_align_safe_end: }
908   \__math_grab_loop:
909 }
910 \tl_map_inline:nn { \ignorespaces \unskip \textonly@unskip }
911 {
912   \cs_new_protected:cpn { __math_grab_loop_ \token_to_str:N #1 : }
913   { \__math_grab_loop: }
914 }

```

(End of definition for `\__math_grab_loop_token:N` and others.)

`\__math_grab_loop_newline:` To allow collection of tokens in the part of the `\halign` template after `#`, we need `TEX` to see the primitive with the loop token in the right place. That is done by re-defining `\cr`

at present. Ideally there would be a socket in the definition of `tabular`, etc., to handle this: there is also the need to examine in interaction with `longtable`, which also redefines `\cr`.

```

915 \cs_new_protected:Npn \__math_grab_loop_newline:
916 {
917   \if_false: { \fi:
918     \cs_set_protected:Npn \cr
919     {
920       \__math_grab_loop:
921       \tex_cr:D
922     }
923   \if_false: } \fi:
924   \\\
925 }
```

(End of definition for `\__math_grab_loop_newline:.`)

`\__math_grab_loop_end:` Clean up and pass on.

```

926 \cs_new_protected:Npn \__math_grab_loop_end:
927 {
928   \exp_args:NNV \group_end:
929   \__math_grab:n \l__math_grabbed_tl
930 }
```

(End of definition for `\__math_grab_loop_end:.`)

8.13 Marking math environments

A general mechanism for math mode environments that do not grab their content (*cf.* most `amsmath` environments).

`\l__math_env_name_tl` To allow us to carry out “special effects”

```

931 \tl_new:N \l__math_env_name_tl
```

Here we set up specialized handling of environments. The idea for the `arg-spec` key is that if an environment takes arguments, we don’t worry during the main grabbing. Rather, we remove the arguments from the grabbed content and forward only the payload. That is done by (ab)using `ltxcmd`.

```

932 \keys_define:nn { __math }
933 {
934   arg-spec .code:n =
935   {
936     \ExpandArgs { c } \DeclareDocumentCommand
937     { __math_env \l__math_env_name_tl _aux: }
938     {#1}
939     { \__math_env_forward:w }
940   }
941 }
```

`\math_register_env:nn` Set up to capture environment content and make available.
`\math_register_env:n`
`\RegisterMathEnvironment`

```

942 \cs_new_protected:Npn \math_register_env:nn #1#2
943 {
```

```

944 \tl_set:Nn \l__math_env_name_tl {#1}
945 \keys_set:nn { __math } {#2}
946 \cs_gset_eq:cc { __math_env_ #1 _begin: } {#1}
947 \cs_gset_eq:cc { __math_env_ #1 _end: } { end #1 }
948 %
949 \ExpandArgs { nnee } \RenewDocumentEnvironment {#1} { b }
950 {
951   \exp_not:N \bool_if:NTF \exp_not:N \l__math_collected_bool
952   {
953     \typeout{==>B1}
954   }
955   {
956     \typeout{==>B2}
957     \cs_if_exist:cTF { __math_env_ #1 _aux: }
958     {
959       \exp_not:c { __math_env_ #1 _aux: }
960       ##1 \exp_not:N \__math_env_end: {#1}
961     }
962     { \exp_not:N \__math_process:nn {#1} {##1} }
963     \exp_not:n { \@kernel@math@registered@begin }
964     \bool_set_true:N \exp_not:N \l__math_collected_bool
965   }
966   % \exp_not:N \tracingall
967   \exp_not:c { __math_env_ #1 _begin: }
968   ##1
969   % \exp_not:N \tracingnone
970 }
971 {
972   \exp_not:c { __math_env_ #1 _end: }
973 }
974 }

975 \cs_new:Npn \@kernel@math@registered@begin {
976 % \ShowTagging{struct-stack}
977 %\typeout{==>A1}\ShowTagging{struct-stack,mc-current}
978 \mode_if_vertical:TF
979 {
980   \__block_beginpar_vmode:
981 }
982 {
983   \UseTaggingSocket{para/textblock/end}
984 }
985 \tag_socket_use:nn{math/display/formula/begin}{-}{-}
986 % \typeout{==>MC1}\ShowTagging{mc-current}
987 }

988
989 \cs_new_protected:Npn \math_register_env:n #1
990 { \math_register_env:nn {#1} { } }
991
992 \NewDocumentCommand \RegisterMathEnvironment { 0{ } m }
993 { \math_register_env:nn {#2} {#1} }

```

(End of definition for `\math_register_env:nn`, `\math_register_env:n`, and `\RegisterMathEnvironment`.
These functions are documented on page 5.)

```
\__math_env_forward:w
```

```
994 \cs_new_protected:Npn \__math_env_forward:w #1 \__math_env_end: #2
995 { \__math_process:nn {#2} {#1} }
```

(End of definition for __math_env_forward:w.)

8.14 Regaining control after a display has finished with \$\$

The tagging structures have to be added after the formula content but before $\TeX$ is allowed to break a page. But $\TeX$ will automatically add `\postdisplaypenalty` followed by `\belowdisplayskip` or `\belowdisplayshortskip` without giving us any chance to take control before these are added.

We therefore implement the following approach:

- Just before we end the formula we save away the current value of `\postdisplaypenalty` in case it was altered within the formula. Then we set it to 10000 so that what is inserted by $\TeX$ is not allowing for a page break at this point
- At this point we also normally flip the sign of `\belowdisplayskip` and `\belowdisplayshortskip` so that they become negative and record that we have done this, by setting the token list `\g__math_skip_sign_tl` to `-`.
- However, we only do that if both are non-negative. If either of them is negative we assume that this was deliberately done by the user to adjust the spacing around this particular display. Again, we record in `\g__math_skip_sign_tl` that we haven't done the sign flip by setting the variable to empty.
- Making these two skips negative is essential, because the formula will be added using the special `\postdisplaypenalty` value that doesn't allow a page break even though the real value (that we use later on) will probably do that. Thus the below skip added by $\TeX$ will add to the size of the display and not vanish into the bottom margin and if it is positive $\TeX$ might decide to move the whole formula onto the next page even though it might well fit.
- Once $\TeX$ has finished processing the closing `$$` it has added something like

```
\penalty 100000          % our special value for \postdisplaypenalty
\glue -10.0 plus -3pt    % the \belowdisplayskip (with sign flipped
```

after the formula. This means that at this that point we can retrieve this skip by using `\lastskip` and we just have to flip the sign again to know how to correct it (no need to know whether the normal or the short skip was added by $\TeX$) and the right penalty is available to us too as we have saved it away in `\g__math_postdisplaypenalty_int`.

```
\__math_prepare_display_end:
```

Prepare to finish the formula and give control to $\TeX$. This is normally used directly in front of `$$` (`\eqno` or `\leqno`).

```
996 \cs_new_protected:Npn \__math_prepare_display_end: {
997   \bool_lazy_or:nnTF
998     { \dim_compare_p:nNn \belowdisplayskip < {0pt} }
999     { \dim_compare_p:nNn \belowdisplayshortskip < {0pt} } }
```

No sign flipping if one or both of the skips are already negative.

```
1000 { \tl_gclear:N \g__math_skip_sign_tl }
```

Otherwise flip the sign of both.

```

1001 {
1002   \tl_gset:Nn \g__math_skip_sign_tl {-}
1003   \skip_set:Nn \belowdisplayskip {-\belowdisplayskip}
1004   \skip_set:Nn \belowdisplayshortskip {-\belowdisplayshortskip}
1005 }

```

For the skip it is enough to flip the sign, because we get the value back through `\lastskip` but for the `\postdisplaypenalty` change we have to save the current value somewhere, because it may have been changed within the display formula. This has to be done globally, because it is used after the formula has ended.

```

1006 \int_gset_eq:NN \g__math_postdisplaypenalty_int \postdisplaypenalty
1007 \int_set_eq:NN \postdisplaypenalty \@M
1008 }

```

(End of definition for `\__math_prepare_display_end:`)

`\__math_tag_dollardollar_display_end:`

The code to be executed after the formula has ended is injected using `\group_insert_after:N` inside of `\tex_everydisplay:D` (i.e., when the formula starts but inside the group started by the display). As `\group_insert_after:N` expects a single token we have to store that code in a command.

```

1009 \cs_new_protected:Npn \__math_tag_dollardollar_display_end:
1010 {
1011   % \typeout{== tag dollarldollar display end}
1012   % \ShowTagging{struct-stack}
1013   \para_raw_end:

```

The `\postdisplaypenalty` was temporarily set to 10000 inside the display and both the `\belowdisplayskip` and the `\belowdisplayshortskip` were negated (with some exceptions, see above), so whatever was inserted it should have been a negative or possibly zero skip. Whatever was added, we pick up the value, so that we can correct the spacing after the tagging code has been inserted.

```

1014 \l__math_tmpa_skip \lastskip
1015 \tag_socket_use:n{math/display/formula/end}

```

Now we add a skip without introducing a page break possibility, that should bring the current vertical position back to the point where T_EX has added the penalty and the “below skip”.

```

1016 \nobreak
1017 \skip_vertical:n { -\l__math_tmpa_skip }

```

Then we finally add the real stuff: the true `\postdisplaypenalty` (saved in `\g__math_postdisplaypenalty_int` earlier) and the negated value of the skip value we saved in `\l__math_tmpa_skip`. It may look strange that we have two identical negated skips next to each other, but if you think about it, that is correct: the first cancels the “below skip” that T_EX has added and the second puts the same amount of skip after the penalty (which is where it should be).

```

1018 \penalty \g__math_postdisplaypenalty_int

```

As we are now in vertical mode the situation is different from the way $\TeX$ would handle things after a display: $\TeX$ would internally switch to horizontal mode without adding a `\parskip`. But this is not possible to do on the macro level. Therefore we have to neutralize the upcoming `\parskip` since we can't prevent it from being added. We have to do this before re-adding the saved skip, because we want to make this skip seen by any following `\addvspace`. Otherwise, we might end up with too much space in such a situation. .

```
1019     \skip_vertical:n { -\tex_parskip:D }
1020     \skip_vertical:n
```

Actually, we use the skip without flipping the sign when our records show that it wasn't flipped earlier i.e., when the negative value was due to the user specifying it inside the formula.

```
1021     { \g__math_skip_sign_tl \l__math_tmpa_skip } }
```

We also set the `@domathendpetrue` flag to signal that paragraph continuation should happen after such a display.

```
1022     \@domathendpetrue
1023     \@doendpe           % this has no \end{...} to take care of it
1024 }
```

(End of definition for __math_tag_dollardollar_display_end:.)

`\g__math_postdisplaypenalty_int` This is the internal variable in which we save the `\postdisplaypenalty`. It is global because we use it immediately after the formula has ended.

```
1025 \int_new:N \g__math_postdisplaypenalty_int
```

(End of definition for \g__math_postdisplaypenalty_int.)

`\g__math_skip_sign_tl` We record whether we have flipped the signs of `\belowdisplayskip`, etc., so that we know what to do after the formula has ended. If we have it flipped the signs then variable holds `-`, whilst otherwise it is empty. This means we can flip the signs again by simply prefixing the value with this token list variable.

```
1026 \tl_new:N \g__math_skip_sign_tl
```

(End of definition for \g__math_skip_sign_tl.)

`\dollardollar@end` When we do tagging we augment the kernel definition for `\dollardollar@end` in order to regain control at the very end of the formula (after the user may have altered `\postdisplaypenalty` or `\belowdisplayskip`).

```
1027 \def \dollardollar@end { \__math_prepare_display_end: $$ }
```

(End of definition for \dollardollar@end.)

`\eqno` `\leqno` However, in case of a formula using the primitives `\eqno` or `\leqno` the `\dollardollar@end` comes too late as it is executed in the context of the equation number; and the values for `\postdisplaypenalty`, etc. set at this point are not the ones used for the formula. We therefore have to execute `\__math_prepare_display_end:` just in front of the primitive.

```
1028 \protected\def\eqno{
1029   \__math_prepare_display_end:
```

Inside the equation we set it temporarily to do nothing, because otherwise it would be executed again when `\dollar\dollar@end` is reached (not that this would matter, but it would just take unnecessary extra time).

```

1030 \kernel@eqno
1031 \cs_set_eq:NN \__math_prepare_display_end: \prg_do_nothing:
1032 \aftergroup\ignorespaces
1033 }

```

Same game for `\leqno`.

```

1034 \protected\def\leqno{
1035   \__math_prepare_display_end:
1036   \kernel@leqno
1037   \cs_set_eq:NN \__math_prepare_display_end: \prg_do_nothing:
1038   \aftergroup\ignorespaces
1039 }

```

(End of definition for `\eqno` and `\leqno`.)

8.15 Document commands

Add one more here: `displaymath`, which is equivalent to `\[ , \]` and hence to the basic `equation*`.
Added in more recent branch.

`\equation`

`\__math_equation_begin:`

`\equation*`

`\__math_equation_star_begin:`

`\endequation`

`\__math_equation_end:`

`\endequation*`

`\__math_equation_star_end:`

These environments are not set up by `amsmath` to collect their body, so we do that here. This has to be done *after* we can be sure `amsmath` is loaded. Note that with `amsmath` loaded, `equation*` and `equation` are the two basics: they are used to define the other single-row display environments, etc.

```

1040 \tl_gput_right:Nn \kernel@before@begindocument
1041 {
1042   \math_register_env:n { equation }
1043   \math_register_env:n { equation* }
1044   % at the moment register_env can only do display math
1045   %   \math_register_env:n { math }
1046   \RenewDocumentEnvironment{math} {b}{\$#1\$}{}
1047   % and this one doesn't work either
1048   %   \math_register_env:n { displaymath }
1049   \RenewDocumentEnvironment{displaymath} {b}{\[#1\]}{}
1050 }

```

(End of definition for `\equation` and others.)

- \(If math mode has not been collected, we need to do that; otherwise, worry about whether
- \) we are in math mode or not. The closing command here can only occur inside a collected math block: otherwise it will be simply used as a delimiter.

```

1051 \cs_gset_protected:Npn \( % \)
1052 {
1053   \bool_if:NTF \l__math_collected_bool
1054   {
1055     \mode_if_math:TF
1056     { \badmath }
1057     { $ }
1058   }

```

```

1059     {
1060       \__math_grab_inline:w
1061     }
1062   } % \(\
1063   \cs_gset_protected:Npn \)
1064   {
1065     \mode_if_math:TF
1066     { $ }
1067     { \@badmath }
1068   }

```

(End of definition for \(\ and \).)

\[Again, we need to watch for when `amsmath` is loaded after this code. The flag usage here
\] is to cover the case where `\[/\]` is hidden inside another environment. In this case the grabbing happens on the outer level and should not be repeated.

```

1069 \tl_gput_right:Nn \@kernel@before@begindocument
1070 {
1071   \cs_gset_protected:Npn \[ % \[
1072   {
1073     \__math_grab_eqn:w
1074   } % \[
1075   \cs_gset_protected:Npn \]
1076   {
1077     \@badmath
1078   }
1079 }

```

(End of definition for \[and \].)

8.16 \everymath and \everydisplay

The business end for grabbing inline math and “raw” $\TeX$ display. Most display math mode is actually handled elsewhere, as we have macro control.

```

1080 \exp_args:No \tex_everymath:D
1081 {
1082   \tex_the:D \tex_everymath:D
1083   \bool_if:NF \l__math_collected_bool
1084   {
1085     \bool_set_true:N \l__math_collected_bool
1086     \__math_grab_dollar:w
1087   }
1088 }
1089
1090 \exp_args:No \tex_everydisplay:D
1091 {
1092   \tex_the:D \tex_everydisplay:D
1093   % \typeout{==>~ in~ everydisplay}

```

We need to attach tagging after the display math has been processed by $\TeX$, and we also need to correct the penalty and spacing that will get added there. This is done by the following code through which we regain control after the display math group ends.

```

1094   \group_insert_after:N \__math_tag_dollardollar_display_end:
1095   \bool_if:NF \l__math_collected_bool

```

```

1096     {
1097       \bool_set_true:N \l__math_collected_bool
1098       \__math_grab_dollardollar:w
1099     }
1100   }

```

8.17 Modifying kernel environments

We need to cover this even though it is, of course, not encouraged. Registration is delayed until `begindocument` to avoid error with redefinition in, e.g., `fleqn.clo`.

```

1101 \tl_gput_right:Nn \@kernel@before@begindocument
1102 {
1103   \math_register_env:n { eqnarray }
1104   \math_register_env:n { eqnarray* }
1105 }

```

In `tabulars` we do need to change the grabbing method to the slow loop-method. `\@kernel@tabular@init` is provided in `array.sty` which is loaded by the table code.

```

1106 \providecommand\@kernel@tabular@init{}
1107 \tl_put_right:Nn\@kernel@tabular@init
1108 {
1109   \cs_set_protected:Npn \__math_grab_dollar:w { \__math_grab_loop_init: }
1110   \cs_set_protected:Npn \__math_grab_inline:w { $ }
1111 }

```

`\__math_m@th:` Handle non-math use of math mode. At present nesting isn't supported as `\m@th` pops up in a few places that *are* math mode!

```

1112 \cs_new_eq:NN \__math_m@th: \m@th
1113 \cs_gset_protected:Npn \m@th
1114 {
1115   \bool_set_true:N \l__math_collected_bool
1116   \__math_m@th:
1117 }

```

(End of definition for `\__math_m@th:` and `\m@th`.)

8.18 Modifying kernel commands

`\mathpalette` The `\mathpalette` command is a problem if `lualatex` is used and math structure elements are created as the math is then processed more than once. We therefore redefine it to use `\mathstyle`.

```

1118 \sys_if_engine luatex:T
1119 {
1120   \def\mathpalette#1#2{%
1121     \ifcase\mathstyle
1122       #1\displaystyle{#2}\or
1123       #1\displaystyle{#2}\or
1124       #1\textstyle{#2}\or
1125       #1\textstyle{#2}\or
1126       #1\scriptstyle{#2}\or
1127       #1\scriptstyle{#2}\or
1128       #1\scriptscriptstyle{#2}\or
1129       #1\scriptscriptstyle{#2}

```

```

1130   \fi}
1131 }

```

(End of definition for `\mathpalette`.)

`\mathsm@sh` Similar to the phantom commands in `latex-lab-text`, `\mathsm@sh` must be redefined to include `luamml` sockets.

```

1132 \def\mathsm@sh#1#2{%
1133   \setbox\z@\hbox{$\m@th#1#2}
1134   \UseTaggingSocket{math/luamml/save/nNn}{\mathsmash} #1 {mpadded}}
1135   $}%
1136   \UseTaggingSocket{math/luamml/finsm@sh}{\finsm@sh}}

```

(End of definition for `\mathsm@sh`.)

`\eqnarray` For MathML generation in `eqnarray` we also need to add some sockets.
`\endeqnarray`

```

1137 <@=)
1138 \def\eqnarray{%
1139   \stepcounter{equation}%
1140   \def\@currentlabel{\p@equation\theequation}%
1141   \def\@currentcounter{equation}%
1142   \global\@eqnswtrue
1143   \m@th
1144   \global\@eqcnt\z@
1145   \tabskip\@centering
1146   \let\\\@eqncr
1147   \dollar\dollar@begin\everycr{}\halign to\displaywidth\bgroup
1148     \hskip\@centering
1149     $\displaystyle\tabskip\z@skip{##}%
1150     \UseTaggingSocket{math/luamml/save/nNn}{\displaystyle {mtd}}%
1151     $%
1152     \UseTaggingSocket{math/luamml/mtable/finalizecol}{last}%
1153     \@eqnset
1154     &\global\@eqcnt\@ne\hskip \tw@\arraycolsep \hfil${##}%
1155     \UseTaggingSocket{math/luamml/save/nNn}{\displaystyle {mtd}}%
1156     $%
1157     \UseTaggingSocket{math/luamml/mtable/finalizecol}{last}%
1158     \hfil
1159     &\global\@eqcnt\tw@ \hskip \tw@\arraycolsep $\displaystyle{##}%
1160     \UseTaggingSocket{math/luamml/save/nNn}{\displaystyle {mtd}}%
1161     $%
1162     \UseTaggingSocket{math/luamml/mtable/finalizecol}{last}%
1163     \hfil\tabskip\@centering
1164     &\global\@eqcnt\thr@@ \hbxt@\z@\bgroup\hss##\egroup
1165     \UseTaggingSocket{math/luamml/mtable/tag/set}
1166     \tabskip\z@skip
1167   \cr
1168 }
1169 \def\endeqnarray{%
1170   \@eqncr
1171   \UseExpandableTaggingSocket{math/luamml/mtable/finalize} {eqnarray}%
1172   \egroup
1173   \global\advance\c@equation\@ne
1174   \dollar\dollar@end\@ignoretrue
1175 }

```

```

1176 \AddToHook{file/fleqn.clo/after}{
1177   \renewenvironment{eqnarray}{%
1178     \stepcounter{equation}%
1179     \def\@currentlabel{\p@equation\theequation}%
1180     \def\@currentcounter{equation}%
1181     \global\@eqnswtrue\m@th
1182     \global\@eqcnt\z@
1183     \tabskip\mathindent
1184     \let\@=\@eqncr
1185     \setlength\abovedisplayskip{\topsep}%
1186     \ifvmode
1187       \addtolength\abovedisplayskip{\partopsep}%
1188     \fi
1189     \addtolength\abovedisplayskip{\parskip}%
1190     \setlength\belowdisplayskip{\abovedisplayskip}%
1191     \setlength\belowdisplayshortskip{\abovedisplayskip}%
1192     \setlength\abovedisplayshortskip{\abovedisplayskip}%
1193     \dollar\dollar@begin\everycr{}\halign to\linewidth%
1194     \bgroup
1195     \hskip\@centering
1196     $\displaystyle\tabskip\z@skip{##}%
1197     \UseTaggingSocket{math/luamml/save/nNn}{\displaystyle {mtd}}%
1198     $%
1199     \UseTaggingSocket{math/luamml/mtable/finalizecol}{last}%
1200     \@eqnse1%
1201     \global\@eqcnt\@ne \hskip \tw@\arraycolsep \hfil${##}%
1202     \UseTaggingSocket{math/luamml/save/nNn}{\displaystyle {mtd}}%
1203     $%
1204     \UseTaggingSocket{math/luamml/mtable/finalizecol}{last}%
1205     \hfil%
1206     \global\@eqcnt\tw@ \hskip \tw@\arraycolsep $\displaystyle{##}%
1207     \UseTaggingSocket{math/luamml/save/nNn}{\displaystyle {mtd}}%
1208     $%
1209     \UseTaggingSocket{math/luamml/mtable/finalizecol}{last}%
1210     \hfil \tabskip\@centering%
1211     \global\@eqcnt\thr@@
1212     \hb@xt@\z@\bgroup\hss##\egroup
1213     \UseTaggingSocket{math/luamml/mtable/tag/set}
1214     \tabskip\z@skip\cr
1215   }%
1216   {%
1217     \@@eqncr
1218     \UseExpandableTaggingSocket{math/luamml/mtable/finalize} {eqnarray}%
1219     \egroup
1220     \global\advance\c@equation\m@ne
1221     \dollar\dollar@end
1222     \@ignoretrue
1223   }
1224 }
1225 \@@=math

```

(End of definition for \eqnarray and \endeqnarray.)

8.19 Disable math grabbing in the begindocument hook

For example amsart uses math to measure text there.

```

1226 \tl_gput_right:Nn\@kernel@before@begindocument
1227 {
1228   \bool_set_true:N\l__math_collected_bool
1229 }
1230 \tl_gput_right:Nn\@kernel@after@begindocument
1231 {
1232   \bool_set_false:N\l__math_collected_bool
1233 }

1234 \ExplSyntaxOff

1235 <@@=)

1236 </kernel>

```

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.

Symbols	
\#	97, 373
\%	98
\(	830
\(	<u>1051</u>
\)	831, 878
\)	<u>1051</u>
@@ commands:	
\l_@@_collected_bool	3, 12
\l_@@_content_template_tl	9
\l_@@_fakemath_bool	12
\[	840, 1049
\[	13, 36, 44, 45, <u>1069</u>
\]	212, 213, 214, 215,
	216, 217, 279, 880, 884, 924, 1146, 1184
\{	95
\}	96
\]	841, 1049
\]	13, 36, 44, 45, <u>1069</u>
\^	99
A	
\abovedisplayshortskip	1192
\abovedisplayskip	
..	1185, 1187, 1189, 1190, 1191, 1192
actual+source (plug)	<u>619</u>
\AddToHook	8, 9, 163, 169, 175,
	188, 200, 251, 354, 394, 411, 438, 1176
\addtolength	1187, 1189
\addvspace	43
\advance	1173, 1220
\aftergroup	1032, 1038
alt+source (plug)	<u>634</u>
\arraycolsep	1154, 1159, 1201, 1206
\AssignTaggingSocketPlug	
	225, 226, 230, 352, 353,
	392, 393, 589, 590, 649, 690, 691,
	764, 765, 766, 767, 768, 769, 770, 771
B	
\begin	38, 595, 613, 844, 886, 891
\begingroup	94
\belowdisplayshortskip	
	41, 42, 999, 1004, 1191
\belowdisplayskip	41–43, 998, 1003, 1190
\bgroup	1147, 1164, 1194, 1212
block internal commands:	
_block_beginpar_vmode:	980
bool commands:	
\bool_gset_false:N	20, 319
\bool_gset_true:N	
	15, 224, 274, 315, 340, 381
\bool_if:NTF	
	26, 28, 73, 222, 262, 389, 448,
	656, 666, 678, 710, 715, 725, 732,
	740, 745, 843, 951, 1053, 1083, 1095
\bool_lazy_any:nTF	800
\bool_lazy_or:nnTF	183, 997

<code>\bool_new:N</code>	10, 33, 34, 35, 47, 48, 49, 50, 51, 52, 53, 273
<code>\bool_set_false:N</code>	267, 307, 792, 837, 1232
<code>\bool_set_true:N</code> 265, 300, 452, 499, 778, 793, 964, 1085, 1097, 1115, 1228	
bool internal commands:	
<code>\l_math_collected_bool</code>	14, 33, 262, 265, 267, 792, 793, 837, 843, 951, 964, 1053, 1083, 1085, 1095, 1097, 1115, 1228, 1232
<code>\g_math_debug_bool</code>	14, 10, 15, 20, 26, 28
<code>\l_math_fakemath_bool</code> 15, 34, 778, 803	
<code>\g_math_luaaml_write_bool</code>	22, 222, 273, 274, 315, 319, 323
<code>\l_math_math_content_artifact_-bool</code>	15, 35, 300, 307, 678
box commands:	
<code>\box_clear:N</code>	388
<code>\box_new:N</code>	364
box internal commands:	
<code>\l_math_tmpa_box</code>	364, 367, 388
<code>\boxed</code>	12
C	
char commands:	
<code>\char_set_catcode_other:N</code>	95, 96, 97, 98, 99, 373
clist commands:	
<code>\clist_map_inline:Nn</code>	375
<code>\clist_map_inline:nn</code>	472
<code>\clist_new:N</code>	61
<code>\clist_put_right:Nn</code>	62, 488, 500
<code>\cr</code>	918, 1167, 1214
cs commands:	
<code>\cs_generate_variant:Nn</code> 424, 692, 782	
<code>\cs_gset_eq:NN</code>	946, 947
<code>\cs_gset_protected:Npe</code>	25, 27
<code>\cs_gset_protected:Npn</code>	1051, 1063, 1071, 1075, 1113
<code>\cs_if_exist:NTF</code>	957
<code>\cs_if_exist_p:N</code>	184, 185
<code>\cs_if_exist_use:N</code>	742, 747
<code>\cs_if_exist_use:NTF</code>	872
<code>\cs_if_free:NTF</code>	146, 151, 156
<code>\cs_new:Npn</code>	234, 275, 276, 975
<code>\cs_new_eq:NN</code>	11, 12, 79, 1112
<code>\cs_new_protected:Npe</code>	893
<code>\cs_new_protected:Npn</code>	13, 18, 23, 30, 31, 64, 80, 93, 102, 143, 220, 365, 415, 773, 783, 789, 790, 794, 795, 818, 820, 830, 839, 840, 848, 854, 863, 865, 870, 876, 878, 880, 886, 902, 912, 915, 926, 942, 989, 994, 996, 1009
<code>\cs_set:Npn</code>	297, 298
<code>\cs_set_eq:NN</code>	148, 153, 158, 305, 306, 374, 498, 1031, 1037
<code>\cs_set_protected:Npn</code>	261, 791, 792, 793, 918, 1109, 1110
D	
<code>\DebugMathOff</code>	11, 30
<code>\DebugMathOn</code>	11, 30
<code>\DeclareDocumentCommand</code>	936
<code>\DeclareMathEnvironment</code>	13
<code>\def</code>	1027, 1028, 1034, 1120, 1132, 1138, 1140, 1141, 1169, 1179, 1180
default (plug)	523, 543, 554, 573, 652
dim commands:	
<code>\dim_compare_p:nNn</code>	998, 999
<code>\displaylines</code>	13
<code>\displaystyle</code>	1122, 1123, 1149, 1150, 1155, 1159, 1160, 1196, 1197, 1202, 1206, 1207
<code>\displaywidth</code>	1147
E	
<code>\egroup</code>	1164, 1172, 1212, 1219
<code>\end</code>	38, 599, 615, 844, 893, 898, 900, 902, 905, 1023
<code>\endeqnarray</code>	1137
<code>\endequation</code>	1040
<code>\endequation*</code>	1040
<code>\endgroup</code>	103
<code>\ensuremath</code>	13
<code>\equalign</code>	13
<code>\eqnarray</code>	1137
<code>\eqno</code>	41, 43, 1028
<code>\equation</code>	1040
<code>\equation*</code>	1040
<code>\everycr</code>	1147, 1193
<code>\everydisplay</code>	45
<code>\everymath</code>	3, 45
exp commands:	
<code>\exp_args:NNV</code>	928
<code>\exp_args:No</code>	1080, 1090
<code>\exp_last_unbraced:Ne</code>	443
<code>\exp_not:N</code>	595, 599, 613, 615, 895, 897, 898, 900, 951, 959, 960, 962, 964, 966, 967, 969, 972
<code>\exp_not:n</code>	597, 609, 614, 963
<code>\ExpandArgs</code>	936, 949
<code>\ExplSyntaxOff</code>	1234
<code>\ExplSyntaxOn</code>	7
F	
<code>\fi</code>	1130, 1188

fi commands:		<code>\g__math_mathchoice_int</code> 25, 412
<code>\fi:</code>	917, 923	<code>\g__math_mathml_AF_attached_int</code>
file commands:		 16, 60, 406, 717
<code>\file_if_exist:nTF</code>	377	<code>\g__math_mathml_AF_found_int</code> ...
<code>\file_input:n</code>	380	 16, 59, 404, 714
G		<code>\g__math_mathml_int</code> ... 16, 57, 70, 400
<code>\GetDocumentProperty</code>	444	<code>\g__math_mathml_total_int</code>
<code>\global</code>		 16, 56, 67, 398
..... 1142, 1144, 1154, 1159, 1164, 1173,		<code>\l__math_mathstyle_int</code> 412, 419
..... 1181, 1182, 1201, 1206, 1211, 1220		<code>\g__math_mml_int</code>
group commands:		 16
<code>\group_align_safe_begin:</code> ..	797, 890	<code>\g__math_mml_total_int</code>
<code>\group_align_safe_end:</code>	799, 907	<code>\g__math_postdisplaypenalty_int</code>
<code>\group_begin:</code>	850	 41, 42, 1006, 1018, 1025
<code>\c_group_begin_token</code>	858	<code>\intertext</code>
<code>\group_end:</code>	928	13
<code>\group_insert_after:N</code>	42, 1094	iow commands:
H		<code>\iow_close:N</code>
<code>\halign</code>	1147, 1193	 255, 358
<code>\hbox</code>	1133	<code>\iow_new:N</code>
hbox commands:		 231, 343
<code>\hbox_set:Nn</code>	367	<code>\iow_newline:</code>
<code>\hfil</code> ... 1154, 1158, 1163, 1201, 1205, 1210		 110, 112, 117, 119, 134, 136, 138, 140
<code>\hskip</code> .. 1148, 1154, 1159, 1195, 1201, 1206		<code>\iow_now:Nn</code> 233, 241, 253, 329, 348, 356
<code>\hss</code>	1164, 1212	<code>\iow_open:Nn</code>
I		 232, 344
if commands:		<code>\iow_term:n</code>
<code>\if_false:</code>	917, 923	 396, 397, 398, 400, 402, 404, 406
<code>\ifcase</code>	1121	iow internal commands:
<code>\ifvmode</code>	1186	<code>\g__math_luamml_iow</code>
<code>\ignorespaces</code>	910, 1032, 1038	 231, 232, 233, 241, 253, 255
int commands:		<code>\g__math_writedummy_iow</code>
<code>\int_compare:nNnTF</code>		 329, 341, 343, 344, 348, 356, 358
..... 238, 421, 882, 889, 895, 906		K
<code>\int_decr:N</code>	268, 904	keys commands:
<code>\int_gincr:N</code> 67, 70, 699, 714, 717		<code>\keys_define:nn</code> 282, 336, 425, 455, 932
<code>\int_gset_eq:NN</code>	1006	<code>\keys_set:nn</code> 461, 474, 480, 490, 505, 945
<code>\int_incr:N</code>	888	
<code>\int_new:N</code>	56,	L
..... 57, 58, 59, 60, 412, 413, 847, 1025		<code>\lastskip</code>
<code>\int_set:Nn</code>	227, 419	 41, 42, 1014
<code>\int_set_eq:NN</code>	1007	<code>\LaTeXe</code>
<code>\int_use:N</code>		 12
..... 135, 398, 400, 402, 404, 406, 414		<code>\leavevmode</code>
int internal commands:		 9
<code>\g__math_AF_attached_int</code>	16	legacy commands:
<code>\g__math_AF_found_int</code>	16	<code>\legacy_if:nTF</code>
<code>\g__math_AF_total_int</code>	16	 775
<code>\l__math_grab_env_int</code>		<code>\legacy_if_p:n</code>
..... 37, 847, 882, 888, 889, 895, 904, 906		 802
<code>\g__math_math_total_int</code>		<code>\leqno</code>
..... 16, 58, 135, 402, 699		 41, 43, 44, 1028
		<code>\let</code>
		 1146, 1184
		<code>\linewidth</code>
		 1193
		<code>\ltxlabmathdate</code>
		 3
		<code>\ltxlabmathversion</code>
		 4
		luamml commands:
		<code>\luamml_flag_ignore:</code>
		 155, 158
		<code>\luamml_flag_process:</code>
		 150, 153
		<code>\luamml_flag_structelem:</code> ..
		 145, 148
		<code>\luamml_ignore:</code> ... 156, 158, 298, 310
		<code>\luamml_process:</code>
		 151, 153
		<code>\luamml_structelem:</code>
		 146, 148, 297

luamml internal commands:		
\l_luamml_pretty_int	227	
__luamml_register_output_hook:N	250	
M		
\m@th	3	
maketag internal commands:		
\maketag_math@	29	
math commands:		
\math_debug_off:	11, 13, 18, 31	
\math_debug_on:	11, 13, 13, 30	
\math_processor:n	5, 790, 790	
\math_register_env:n	5, 942,	989, 1042, 1043, 1045, 1048, 1103, 1104
\math_register_env:nn		5, 942, 942, 990, 993
math internal commands:		
__math_AF_html_reader:w	93, 374	
__math_AF_html_reader_verb:w		100, 102
__math_AF_mml:nnnn	64, 64, 104	
__math_AF_process_mathml_files:		364, 365, 411
__math_debug:n	11, 11, 25	
__math_debug_gset:	13, 16, 21, 23	
__math_debug_typeout:n		11, 12, 27, 677, 718, 721, 753
__math_env_end:	960, 994	
__math_env_forward:w	939, 994, 994	
__math_equation_begin:	1040	
__math_equation_end:	1040	
__math_equation_star_begin:	1040	
__math_equation_star_end:	1040	
__math_grab:n	794, 795, 819, 929	
__math_grab_dollar:w		794, 794, 1086, 1109
__math_grab_dollar_delim:w		794, 818, 818
__math_grab_dollardollar:w		820, 820, 1098
__math_grab_eqn:w	840, 840, 1073	
__math_grab_inline:w		839, 839, 1060, 1110
__math_grab_inline_delim:w		830, 830, 839
__math_grab_loop:		848, 852, 854, 868, 908, 913, 920
__math_grab_loop_\$:	870	
__math_grab_loop_\:	870	
__math_grab_loop_\begin:	870	
__math_grab_loop_\end:	870	
__math_grab_loop_\end:n	870	
__math_grab_loop_\ignorespaces:	870	
__math_grab_loop_\textonly@unskip:	870	
__math_grab_loop_\unskip:	870	
__math_grab_loop_end:		877, 879, 926, 926
__math_grab_loop_group:n		859, 863, 863
__math_grab_loop_init:	848, 848, 1109	
__math_grab_loop_newline:		883, 897, 915, 915
__math_grab_loop_store:n		863, 864, 865, 874, 884, 891
__math_grab_loop_token:N		860, 870, 870
__math_luamml_activate_write:		22, 171, 190, 220
__math_luamml_ignore:		275, 276, 298, 306, 807
__math_luamml_output_hook:n	234, 250	
__math_luamml_structelem:		275, 275, 297, 305, 527, 559
__math_m@th:	1112, 1112, 1116	
__math_MSFT_mml:nn	79, 79, 105, 498	
__math_MSFT_mml_aux:nn	79, 80, 498	
__math_prepare_display_end:		43, 827,
__math_process:nn		996, 996, 1027, 1029, 1031, 1035, 1037
__math_process_auxi:nn	772, 773, 782, 798, 823, 962, 995	
__math_process_auxii:nn	772, 779, 783	
__math_process_auxiii:nn	772, 787, 789, 791	
__math_provide_luamml_commands:		143, 196, 202
__math_tag_dollardollar_		display_end: 1009, 1009, 1094
__math_tag_if_mathstyle:nn		415, 415, 424
math/content (tag socket)	618	
math/display/begin (tag socket)	539	
math/display/end (tag socket)	539	
math/display/formula/begin (tag socket)	539	
math/display/formula/end (tag socket)	539	
math/display/tag/begin (tag socket)	571	
math/display/tag/end (tag socket)	571	
math/end (tag socket)	763	
math/inline/begin (tag socket)	511	
math/inline/end (tag socket)	511	
math/inline/formula/begin (tag socket)	511	
math/inline/formula/end (tag socket)	511	
math/mathml/write (tag socket)	326	
math/mathml/write/prepare (tag socket)	125	
math/struct/begin (tag socket)	650	

str internal commands:		
\l__math_tmpa_str	15, 40, 128, 129, 130, 137	
\SuspendTagging	3	
\symletters	229	
\symsymbols	228	
sys commands:		
\sys_if_engine_luatex:TF	161, 299, 1118	
\c_sys_jobname_str	63, 232, 346, 489, 501	
T		
\tabskip	1145, 1149, 1163, 1166, 1183, 1196, 1210, 1214	
tag commands:		
\tag_attr_new:nn	86	
\tag_mc_begin:n	579, 587, 680, 682, 754	
\tag_mc_begin_pop:n	522	
\tag_mc_end:	577, 585, 688, 760	
\tag_mc_end_push:	518	
\tag_resume:n	422	
\tag_socket_use:n	528, 529, 531, 537, 560, 561, 563, 569, 730, 731, 811, 814, 815, 824, 1015	
\tag_socket_use:nn	825, 985	
\tag_socket_use:nnn	812	
\tag_struct_begin:n	6, 578, 668, 735	
\tag_struct_end:	586, 688, 761	
\tag_suspend:n	422	
tag internal commands:		
\l__tag_math_alt_bool	16, 52, 428, 448, 452, 666, 710	
\g__tag_math_lua_mml_tl	16, 54, 55	
\g__tag_math_mathml_AF_bool	16, 50, 224, 340, 381, 389	
\l__tag_math_mathml_AF_bool	16, 49, 433, 715, 725, 745	
\l__tag_math_mathml_files_clist	16, 61, 62, 375, 427, 488, 500	
\l__tag_math_mathml_pane_bool	16, 51, 73, 430	
\l__tag_math_MSFT_bool	16, 53, 429, 499, 740	
\l__tag_math_texsource_AF_bool	16, 47, 435, 656, 732	
\l__tag_math_texsource_pane_bool	16, 48, 432	
Tagging Sockets:		
math/content	618	
math/display/begin	539	
math/display/end	539	
math/display/formula/begin	539	
math/display/formula/end	539	
math/display/tag/begin	571	
math/display/tag/end	571	
math/end	763	
math/inline/begin	511	
math/inline/end	511	
math/inline/formula/begin	511	
math/inline/formula/end	511	
math/mathml/write	326	
math/mathml/write/prepare	125	
math/struct/begin	650	
math/struct/end	650	
\tagpdfparaOff	526, 558	
\tagpdfsetup	216, 218	
TeX and L ^A T _E X 2 _ε commands:		
\@eqncr	1170, 1217	
\@M	1007	
\@badmath	1056, 1067, 1077	
\@centering	1145, 1148, 1163, 1195, 1210	
\@currentcounter	1141, 1180	
\@currentlabel	1140, 1179	
\@doendpe	1023	
\@domathendpetrue	1022	
\@eqcnt	1144, 1154, 1159, 1164, 1182, 1201, 1206, 1211	
\@eqncr	1146, 1184	
\@eqnset	1153, 1200	
\@eqnswtrue	1142, 1181	
\@ignoretrue	1174, 1222	
\@iiiparbox	12	
\@kernel@after@begindocument	1230	
\@kernel@before@begindocument	1040, 1069, 1101, 1226	
\@kernel@eqno	1030	
\@kernel@leqno	1036	
\@kernel@math@registered@begin	963, 975	
\@kernel@tabular@init	46, 1106, 1107	
\@math@level	239, 268	
\@ne	1154, 1201	
\c@equation	1173, 1220	
\cr	37–39	
\dollardollar@begin	1147, 1193	
\dollardollar@end	43, 44, 1027, 1174, 1221	
\finism@sh	1136	
\frozen@everymath	20	
\halign	35, 38	
\hb@xt@	1164, 1212	
\ifmeasuring@	5, 12, 772	
\m@ne	1173, 1220	
\m@th	4, 12, 13, 15, 36, 46, 777, 1112, 1133, 1143, 1181	
\math@level	20	
\mathsm@sh	47, 1132	

<code>\p@equation</code>	1140, 1179	<code>\l__math_content_AF_mathml_tl</code> . .	
<code>\textonly@unskip</code>	910		45, 631, 646
<code>\thr@@</code>	1164, 1211	<code>\l__math_content_AF_source_tl</code> . .	
<code>\tw@</code>	1154, 1159, 1201, 1206		43, 128, 627, 642, 657, 701, 733
<code>\z@</code>	1133, 1144, 1164, 1182, 1212	<code>\l__math_content_AF_source_tmpa_</code>	
<code>\z@skip</code>	1149, 1166, 1196, 1214	<code>tl</code> . .	44, 657, 658, 672, 733, 734, 749
tex commands:		<code>\l__math_content_AF_tl</code>	15
<code>\tex_cr:D</code>	921	<code>\l__math_content_alt_tl</code>	
<code>\tex_everydisplay:D</code> . . .	42, 1090, 1092	. .	15, 41, 632, 638, 667, 675, 711, 751
<code>\tex_everymath:D</code>	1080, 1082	<code>\l__math_content_hash_tl</code> . . .	139,
<code>\tex_parskip:D</code>	1019		693, 700, 712, 718, 721, 727, 742, 747
<code>\tex_the:D</code>	1082, 1092	<code>\l__math_content_template_tl</code> . . .	
<code>\text</code>	25		30, 591, 592, 625, 640
<code>\textstyle</code>	1124, 1125	<code>\l__math_env_name_tl</code>	39, 931, 937, 944
<code>\textsubscript</code>	3	<code>\g__math_grabbed_env_tl</code>	
<code>\textsuperscript</code>	3		15, 31, 36, 595, 599,
<code>\theequation</code>	1140, 1179		606, 613, 615, 659, 673, 703, 750, 785
tl commands:		<code>\g__math_grabbed_math_tl</code>	15,
<code>\c_empty_tl</code>	445		31, 37, 597, 609, 614, 677, 702, 753, 786
<code>\c_space_tl</code>	135,	<code>\l__math_grabbed_math_tl</code> . .	694, 702
.	398, 400, 402, 404, 406, 596, 598, 600	<code>\l__math_grabbed_tl</code>	
<code>\tl_clear:N</code>	658, 734, 851		37, 846, 851, 867, 905, 929
<code>\tl_const:Nn</code>	107, 115, 121, 603	<code>\c__math_inline_env_tl</code>	603, 606
<code>\tl_gclear:N</code>	1000	<code>\g__math_luaaml_load_tl</code>	
<code>\tl_gput_right:Nn</code>			22, 165, 271,
.	1040, 1069, 1101, 1226, 1230		272, 285, 286, 314, 460, 479, 487, 497
<code>\tl_gset:Nn</code>	55, 90, 272, 285, 286,	<code>\c__math_mathml_write_after_tl</code> .	
.	314, 460, 479, 487, 497, 785, 786, 1002		107, 245, 333
<code>\tl_gset_eq:NN</code>	76	<code>\l__math_mathml_write_before_tl</code>	
<code>\tl_if_blank:nTF</code>	821, 833		107, 131, 236, 243, 331
<code>\tl_if_blank_p:n</code>	804	<code>\c__math_mathml_write_final_tl</code> .	
<code>\tl_if_empty:NTF</code>	236		107, 254, 357
<code>\tl_if_eq:NNTF</code>	606	<code>\c__math_mathml_write_init_tl</code> . .	
<code>\tl_if_eq:NnTF</code>	659, 703		107, 233, 350
<code>\tl_if_exist:NTF</code>	68, 83, 341, 712	<code>\g__math_skip_sign_tl</code>	
<code>\tl_if_in:nTF</code>	777		41, 1000, 1002, 1021, 1026
<code>\tl_map_inline:nn</code>	910	<code>\l__math_texsource_template_tl</code> .	
<code>\tl_new:N</code>	36, 37, 38, 41,		30, 602, 604, 629, 644
.	42, 43, 44, 45, 46, 54, 75, 89, 114,	<code>\l__math_tmpa_tl</code>	
.	271, 591, 602, 693, 694, 846, 931, 1026		15, 38, 72, 74, 76, 85, 88
<code>\tl_put_right:Nn</code>	867, 905, 1107	token commands:	
<code>\tl_set:Nn</code>	131, 592, 604, 623,	<code>\token_to_str:N</code>	873,
.	627, 631, 632, 638, 642, 646, 647,		876, 878, 880, 886, 893, 900, 902, 912
.	661, 664, 667, 700, 705, 708, 711, 944	<code>\topsep</code>	1185
<code>\tl_set_eq:NN</code>	657, 702, 733	<code>\tracingall</code>	966
<code>\tl_to_str:n</code>	216, 218	<code>\tracingnone</code>	969
<code>\tl_trim_spaces:apply:nN</code>	779	<code>\typeout</code>	28, 379, 384, 391, 450,
tl internal commands:			727, 843, 953, 956, 977, 986, 1011, 1093
<code>\l__math_attribute_class_tl</code>			
.	15, 46,		
.	661, 664, 670, 671, 705, 708, 737, 738		
<code>\l__math_content_actual_tl</code>			
.	15, 42, 623, 647, 674		
		U	
		<code>\unskip</code>	910
		use commands:	
		<code>\use_i:nn</code>	443
		<code>\use_none:n</code>	11, 12

<code>\use_none:nn</code>	79	. 547, 983, 1134, 1136, 1150, 1152,
<code>\UseExpandableTaggingSocket</code> .	1171, 1218	1155, 1157, 1160, 1162, 1165, 1197,
<code>\UseMathForPositioningText</code>	21, <u>261</u>	1199, 1202, 1204, 1207, 1209, 1213
<code>\UseStructureName</code>	578, 670, 737	V
<code>\UseTaggingSocket</code>		<code>\vcenter</code> 3, 12, 21