The `latex-lab-amsmath` code*

LaTeX Project

May 31, 2025

Abstract

Contents

1 Introduction

This file implements adaptations to the amsmath package needed for the tagging project.

2 The Implementation

Better no @@ expansion here .....

```
1 ⟨@@=⟩
2 ⟨*kernel⟩
```

*

2.1 File declaration

```
\ProvidesFile{latex-lab-amsmath.ltx}
        [2025-05-23 v0.1g amsmath adaptions]
```

2.2 Tagpdf support

To make the code independent from tagging being loaded and active we load the tagpdf-base package:

```
\RequirePackage{tagpdf-base}
\ExplSyntaxOn
```

2.3 Measuring

When measuring we neither want tagging nor the luamml processing.

\measuring@true

```
\def\measuring@true{\let\ifmeasuring@\iftrue\tag_suspend:n{\measuring}\luamml_ignore:}
```

(*End of definition for* `\measuring@true`. *This function is documented on page* **??**.)

2.4 Display environments

2.4.1 Tag

The tag/label must be saved, so that it can be reinserted later.

TODO: `\maketag@@@` is perhaps used in places where tagging/luamml handling is not wanted. This must be checked and handled.

\maketag@@@

```
\def\maketag@@@#1
  {%
    \ifmeasuring@
      \hbox{\m@th\normalfont#1}%
    \else
      \UseTaggingSocket{math/display/tag/begin}
      \hbox{\m@th\normalfont#1
      \UseTaggingSocket{math/luamml/mtable/tag/save}
      }%
      \UseTaggingSocket{math/display/tag/end}
    \fi
   }
```

(*End of definition for* `\maketag@@@`. *This function is documented on page* **??**.)

`\eqref` uses `\tagform@` and so `\maketag@@@` but we do not want this tagging there.

\eqref
\maketag@@@notog

```
\def\maketag@@@notag#1{\hbox{\m@th\normalfont#1}}
\DeclareRobustCommand{\eqref}[1]
  {\textup{\let\maketag@@@\maketag@@@notag\tagform@{\ref{#1}}}}
```

(*End of definition for* `\eqref` *and* `\maketag@@@notog`. *These functions are documented on page* **??**.)

2.4.2 align & friends

Most display environment uses a common command for the end which contains the luamml socket to finalize the mtable.

\common@align@ending

```
\def\common@align@ending {
  \math@cr \black@\totwidth@
  \UseExpandableTaggingSocket {math/luamml/mtable/finalize} {\@currenvir}
  \egroup
  \ifingather@
    \restorealignstate@
    \egroup
    \nonumber
    \ifnum0=`{\fi\iffalse}\fi
  \else
    \dollardollar@end
  \fi
  \ignorespacesafterend
}
```

(*End of definition for* `\common@align@ending`*. This function is documented on page* **??**.)

Now we redefine the display alignments to use these ending.

```
\renewenvironment{align}{%
  \start@align\@ne\st@rredfalse\m@ne
}{%
  \common@align@ending
}
\renewenvironment{align*}{%
  \start@align\@ne\st@rredtrue\m@ne
}{%
  \common@align@ending
}
\renewenvironment{alignat}{%
  \start@align\z@\st@rredfalse
}{%
  \common@align@ending
}
\renewenvironment{alignat*}{%
  \start@align\z@\st@rredtrue
}{%
  \common@align@ending
}
\renewenvironment{flalign}{%
  \start@align\tw@\st@rredfalse\m@ne
}{%
  \common@align@ending
}
\renewenvironment{flalign*}{%
  \start@align\tw@\st@rredtrue\m@ne
}{%
  \common@align@ending
}
```

```
\renewenvironment{xalignat}{%
  \start@align\@ne\st@rredfalse
}{%
  \common@align@ending
}
\renewenvironment{xalignat*}{%
  \start@align\@ne\st@rredtrue
}{%
  \common@align@ending
}
\renewenvironment{xxalignat}{%
  \start@align\tw@\st@rredtrue
}{%
  \common@align@ending
}
```

And register these environments for the math grabbing.

```
 \math_register_halign_env:nn {align}{}
 \math_register_halign_env:nn {align*}{}
 \math_register_halign_env:nn {alignat}{}
 \math_register_halign_env:nn {alignat*}{}
 \math_register_halign_env:nn {flalign}{}
 \math_register_halign_env:nn {flalign*}{}
 \math_register_halign_env:nn {xalignat}{}
 \math_register_halign_env:nn {xalignat*}{}
 \math_register_halign_env:nn {xxalignat}{}
```

The align preamble (used in `\align@`) needs code for luamml to save the cells.

`\align@preamble`

```
\def\align@preamble{%
   &\hfil
    \strut@
    \setboxz@h
      {
        \@lign
         $
          \m@th\displaystyle{##}
         \ifmeasuring@
          \luamml_ignore:
         \else
          \UseTaggingSocket{math/luamml/save/nNn}{ {} \displaystyle {mtd} }
         \fi
         $
      }%
    \ifmeasuring@
      \savefieldlength@
    \else
      \UseTaggingSocket{math/luamml/mtable/finalizecol}{box}
    \fi
    \set@field
    \tabskip\z@skip
   &\setboxz@h
```

```
      {
        \@lign
         $
           \m@th\displaystyle{{}##}
           \ifmeasuring@
           \luamml_ignore:
           \else
           \UseTaggingSocket{math/luamml/save/nNn}{ {} \displaystyle {mtd} }
           \fi
         $
      }%
    \ifmeasuring@
      \savefieldlength@
    \else
      \UseTaggingSocket{math/luamml/mtable/finalizecol}{box}
    \fi
    \set@field
    \hfil
    \tabskip\alignsep@
}
```

(*End of definition for* `\align@preamble`. *This function is documented on page* **??**.)

`\math@cr@@@align`

```
\def\math@cr@@@align{%
  \ifst@rred\nonumber\fi
  \if@eqnsw \global\tag@true \fi
  \global\advance\row@\@ne
  \add@amps\maxfields@
  \omit
  \kern-\alignsep@
  \iftag@
    \setboxz@h{\@lign\strut@{\make@display@tag}}%
    \place@tag
  \fi
  \UseTaggingSocket{math/luamml/mtable/tag/set}
  \ifst@rred\else\global\@eqnswtrue\fi
  \global\lineht@\z@
  \cr
}
```

(*End of definition for* `\math@cr@@@align`. *This function is documented on page* **??**.)

2.4.3 gather and gather*

`\gather@`

```
\def\gather@#1{%
    \ingather@true \let\split\insplit@
    \let\tag\tag@in@align \let\label\label@in@display
    \chardef\dspbrk@context\z@
    \intertext@ \displ@y@ \Let@
    \let\math@cr@@@\math@cr@@@gather
    \gmeasure@{#1}%
```

```
    \global\shifttag@false
    \tabskip\z@skip
    \global\row@\@ne
    \halign to\displaywidth\bgroup
        \strut@
        \setboxz@h
          {
           $\m@th\displaystyle{##}
            \UseTaggingSocket{math/luamml/save/nNn}{ {} \displaystyle {mtd} }
           $
          }%
        \UseTaggingSocket{math/luamml/mtable/finalizecol}{box}
        \calc@shift@gather
        \set@gather@field
        \tabskip\@centering
       &\setboxz@h{\strut@{##}}%
        \place@tag@gather
        \UseTaggingSocket{math/luamml/mtable/tag/set}
         \tabskip \iftagsleft@ \gdisplaywidth@ \else \z@skip \span\fi
         \crcr
         #1%
}
```

(*End of definition for* `\gather@`*. This function is documented on page* **??**.)

`\endgather`

```
\def\endgather{
      \math@cr
      \black@ \totwidth@
      \UseExpandableTaggingSocket{math/luamml/mtable/finalize} {gather}
  \egroup
  \dollardollar@end
  \ignorespacesafterend
}
```

(*End of definition for* `\endgather`*. This function is documented on page* **??**.)

The original definition of `gather*` uses `\endgather` but this redirection doesn't work if we alter `gather` so we use the real meaning and add the socket.

`gather*` (*env.*)

```
\renewenvironment{gather*}
 {
   \start@gather\st@rredtrue
 }
 {
    \math@cr
    \black@\totwidth@
    \UseExpandableTaggingSocket{math/luamml/mtable/finalize} {gather}
   \egroup
   \dollardollar@end
   \ignorespacesafterend
 }
```

Register both environments

```
\math_register_halign_env:nn {gather}{}
\math_register_halign_env:nn {gather*}{}
```

2.4.4 gathered, aligned and alignedat

These environments are not grabbed as they are inside other display environments but they need various sockets for luamml support.

\start@aligned

```
\renewcommand{\start@aligned}[2]{
  \RIfM@
  \else
    \nonmatherr@ {\begin{\@currenvir}}
  \fi
  \savecolumn@ % Assumption: called inside a group
  \UseTaggingSocket{math/luamml/annotate/false}{}{ \alignedspace@left }
  \ams@start@box {#1} \bgroup
    \maxfields@ #2 \relax
    \ifnum \maxfields@ > \m@ne
      \multiply \maxfields@ \tw@
      \let \math@cr@@@ \math@cr@@@alignedat
      \alignsep@ \z@skip
    \else
      \let \math@cr@@@ \math@cr@@@aligned
      \alignsep@ \minalignsep
    \fi
    \Let@ \chardef \dspbrk@context \@ne
    \default@tag
    \spread@equation % no-op if already called
    \global \column@ \z@
    \ialign \bgroup
      & \column@plus
        \hfil
        \strut@
        $
          \m@th
          \displaystyle
          {##}
          \UseTaggingSocket{math/luamml/save/nNn}{ {} \displaystyle {mtd} }
        $
        \UseTaggingSocket{math/luamml/mtable/finalizecol}{last}
        \tabskip \z@skip
      & \column@plus
        $
          \m@th
          \displaystyle
          {
            {}
            ##
          }
          \UseTaggingSocket{math/luamml/save/nNn}{ {} \displaystyle {mtd} }
        $
```

```
          \UseTaggingSocket{math/luamml/mtable/finalizecol}{last}
          \hfil
          \tabskip\alignsep@
        \crcr
        \ams@return@opt@arg
}
```

(*End of definition for* `\start@aligned`. *This function is documented on page* **??**.)

`gathered` (*env.*)

```
\renewenvironment{gathered}[1][c]{%
    \RIfM@\else
        \nonmatherr@{\begin{gathered}}%
    \fi
```

This annotates the space

```
    \UseTaggingSocket{ math/luamml/annotate/false } {}{ \alignedspace@left }
    \ams@start@box{#1}\bgroup
        \Let@ \chardef\dspbrk@context\@ne \restore@math@cr
        \spread@equation
        \ialign\bgroup
            \hfil\strut@$\m@th\displaystyle##
```

This save the cell and then finalize it.

```
            \UseTaggingSocket{math/luamml/save/nNn}{ {} \displaystyle {mtd} }
            $
            \UseTaggingSocket{math/luamml/mtable/finalizecol}{last}
            \hfil
            \crcr
    \ams@return@opt@arg
}{%
  \endaligned
}
```

`\endaligned`

```
\def\endaligned
 {
      \crcr
      \UseExpandableTaggingSocket{math/luamml/mtable/innertable/save}
    \egroup
    \restorecolumn@
  \egroup
  \UseTaggingSocket{math/luamml/mtable/innertable/finalize}
 }
```

(*End of definition for* `\endaligned`. *This function is documented on page* **??**.)

2.4.5 multline and multline*

`multline*` needs a redefinition before it is registered as it uses `\endmultline`, this must be replaced by the true code.

`multline*` (*env.*)

```
\renewenvironment{multline*}{\start@multline\st@rredtrue}
 {%
  \iftagsleft@ \@xp\lendmultline@ \else \@xp\rendmultline@ \fi
  \ignorespacesafterend
 }
```

And now we register both

```
\math_register_halign_env:nn {multline}{}
\math_register_halign_env:nn {multline*}{}
```

In the internal commands we have to add sockets for alignment attributes

`\multline@`

```
\def\multline@#1{%
    \Let@
    \@display@init{\global\advance\row@\@ne \global\dspbrk@lvl\m@ne}%
    \chardef\dspbrk@context\z@
    \restore@math@cr
    \let\tag\tag@in@align
    \global\tag@false \global\let\raise@tag\@empty
    \mmeasure@{#1}%
    \let\tag\gobble@tag \let\label\@gobble
    \tabskip \if@fleqn \@mathmargin \else \z@skip \fi
    \totwidth@\displaywidth
    \if@fleqn
        \advance\totwidth@-\@mathmargin
    \fi
    \halign\bgroup
        \hbox to\totwidth@{%
            \if@fleqn
                \hskip \@centering \relax
            \else
                \hfil
            \fi
            \strut@
            $\m@th\displaystyle{}##\endmultline@math
            \hfil
        }%
        \crcr
        \if@fleqn
            \hskip-\@mathmargin
            \def\multline@indent{\hskip\@mathmargin}% put it back
        \else
            \hfilneg
            \def\multline@indent{\hskip\multlinegap}%
        \fi
```

```
        \iftagsleft@
            \iftag@
                \begingroup
                    \ifshifttag@
                        \rlap{\vbox{%
                                \normalbaselines
                                \hbox{%
                                    \strut@
                                    \make@display@tag
                                }%
                                \vbox to\lineht@{}%
                                \raise@tag
                        }}%
                        \multline@indent
                    \else
                        \setbox\z@\hbox{\make@display@tag}%
                        \dimen@\@mathmargin \advance\dimen@-\wd\z@
                        \ifdim\dimen@<\multlinetaggap
                          \dimen@\multlinetaggap
                        \fi
                        \box\z@ \hskip\dimen@\relax
                    \fi
                \endgroup
            \else
                \multline@indent
            \fi
        \else
            \multline@indent
        \fi
    \ifmeasuring@ \else
      \UseTaggingSocket{math/luamml/mtable/aligncol} {left}
    \fi
    #1%
    \ifmeasuring@ \else
      \UseTaggingSocket{math/luamml/mtable/aligncol} {right}
    \fi
}
```

(*End of definition for* `\multline@`. *This function is documented on page* ??.)

Luckily, `multline` uses `\endmultline@math` in exactly the spot where we have to set the flag. Less luckily, `\endmultline@math` sometimes get overwritten for the last line. But that isn't much of a problem since we want special behavior there anyway.

`\endmultline@math`

```
\def\endmultline@math
 {
   \UseTaggingSocket{math/luamml/save/nNn}{ {} \displaystyle {mtd} }
   $
   \UseTaggingSocket{math/luamml/mtable/finalizecol}{last}
 }
```

(*End of definition for* `\endmultline@math`. *This function is documented on page* ??.)

\rendmultline@

```
\def\rendmultline@{%
    \iftag@
```

we need to surround the math token with tagging sockets.

```
      \UseTaggingSocket{math/luamml/save/nNn}{ {} \displaystyle {mtd} }
      $
      \UseTaggingSocket{math/luamml/mtable/finalizecol}{last}
        \let\endmultline@math\relax
            \ifshifttag@
                \hskip\multlinegap
                \llap{\vtop{%
                    \raise@tag
                    \normalbaselines
                    \setbox\@ne\null
                    \dp\@ne\lineht@
                    \box\@ne
                    \hbox{\strut@\make@display@tag}%
                }}%
            \else
                \hskip\multlinetaggap
                \make@display@tag
            \fi
```

Here we set the tag TODO: is that sensible with multline? Where is the tag saved?

```
    \UseTaggingSocket{math/luamml/mtable/tag/set}
    \else
        \hskip\multlinegap
    \fi
    \hfilneg
        \math@cr
```

Now we finalize the mtable.

```
       \UseExpandableTaggingSocket {math/luamml/mtable/finalize} {multline}
  \egroup
  \dollardollar@end
}
```

(*End of definition for* `\rendmultline@`. *This function is documented on page* **??**.)

And something similar for the left version

\lendmultline@

```
\def\lendmultline@{%
        \hfilneg
        \hskip\multlinegap
        \math@cr
        \UseExpandableTaggingSocket {math/luamml/mtable/finalize} {multline}
    \egroup
    \dollardollar@end
}
```

(*End of definition for* `\lendmultline@`. *This function is documented on page* **??**.)

2.5 Cases

env@cases

```
\def\env@cases{%
  \let\@ifnextchar\new@ifnextchar
  \left\lbrace
  \def\arraystretch{1.2}%
  \array{@{}l@{\quad}l@{\luamml_ignore:}}%
}
```

(*End of definition for* `env@cases`. *This function is documented on page* **??**.)

2.5.1 smallmatrix

`smallmatrix` (*env.*)

```
\renewenvironment {smallmatrix} {
  \UseTaggingSocket{ math/luamml/annotate/false } {} { \null\, }
  \vcenter \bgroup
    \Let@
    \restore@math@cr
    \default@tag
    \baselineskip 6 \ex@
    \lineskip 1.5 \ex@
    \lineskiplimit \lineskip
    \ialign \bgroup
        \hfil
        $
        \m@th
        \scriptstyle
        ##
        % No \scriptsize here since we want to add the mstyle nodes
        \UseTaggingSocket{math/luamml/save/nn}{ {} {mtd} }
        $
        \UseTaggingSocket{math/luamml/mtable/finalizecol}{last}
        \hfil
      &&
        \thickspace
        \hfil
        $
        \m@th
        \scriptstyle
        ##
        % No \scriptsize here since we want to add the mstyle nodes
        \UseTaggingSocket{math/luamml/save/nn}{ {} {mtd} }
        $
        \UseTaggingSocket{math/luamml/mtable/finalizecol}{last}
        \hfil
      \crcr
}{%
      \crcr
      \UseExpandableTaggingSocket{math/luamml/mtable/smallmatrix/save}
    \egroup
  \egroup
```

```
  \UseTaggingSocket{math/luamml/mtable/innertable/finalize}
  \UseTaggingSocket{ math/luamml/annotate/false } {}{ \, }
}
```

2.6 The split environment

The split environment is not trivial to handle as it has a rather convoluted implementation in amsmath: depending on in which display environment it is embedded it takes different branches, which makes it difficult to finalize the mtable.

The following patches work with leqno and reqno if the (default) `centertags` are used. The currently fail (the structure is wrong) if the option `tbtags` is used. The alignment of the cells is currently not handled. A simple debug command until everything is sorted.

__math_split_debug_typeout:n

```
\cs_new:Npn\__math_split_debug_typeout:n#1{\use_none:n{#1}}
```

(*End of definition for* `\__math_split_debug_typeout:n`.)

We need to detect if `\gather@split` has been used or not

\l__math_gathersplit_bool

```
\bool_new:N\l__math_gathersplit_bool
```

(*End of definition for* `\l__math_gathersplit_bool`.)

At first a redefinition of the main environment. Here we only have to add the saving command for the inner table:

`split` (*env.*)

```
\renewenvironment{split}{%
  \__math_split_debug_typeout:n {begin~split}
  \if@display
    \ifinner
       \@xp\@xp\@xp\split@aligned
    \else
      \ifst@rred \else \global\@eqnswtrue \fi
    \fi
  \else
    \let\endsplit\@empty \@xp\collect@body\@xp\split@err
  \fi
  \collect@body\gather@split
}{\__math_split_debug_typeout:n{end~split}%
     \crcr
    \UseExpandableTaggingSocket{math/luamml/mtable/innertable/save}%
    \egroup
  \egroup
  \iftagsleft@ \@xp\lendsplit@ \else \@xp\rendsplit@ \fi
}
```

In `\gather@split` we have to add the finalization socket. We also set the boolean to true so that we can detect if the finalization has already happened. Perhaps this could be done in the luamml code instead?

\gather@split

```
\def\gather@split#1#2#3{
  \__math_split_debug_typeout:n{gather@split}%
  \@xp\endgroup \reset@equation % math@cr will handle equation numbering
  \iftag@
     \toks@\@xp{\df@tag}%
     \edef\split@tag{%
       \gdef\@nx\df@tag{\the\toks@}%
       \global\@nx\tag@true \@nx\nonumber
     }%
  \else \let\split@tag\@empty
  \fi
  \bool_set_true:N\l__math_gathersplit_bool
  \spread@equation
  \vcenter\bgroup
    \gather@{\split@tag
    \begin{split}#1\end{split}}%
    \def\endmathdisplay@a{%
      \__math_split_debug_typeout:n{endmathdisplay@a}
      \__math_split_debug_typeout:n{finalize~innertable~endmathdisplay@a}
      \math@cr
      \black@ \totwidth@
      \egroup
      \egroup
      \UseExpandableTaggingSocket{math/luamml/mtable/innertable/finalize}%
    }%
}
```

(*End of definition for* `\gather@split`. *This function is documented on page* **??**.)

`\insplit@` In `\insplit@` we have to add the sockets which store the cell content.

```
\def\insplit@{\__math_split_debug_typeout:n{insplit@}%
  \global\setbox\z@\vbox\bgroup
    \Let@ \chardef\dspbrk@context\@ne \restore@math@cr
    \default@tag % disallow use of \tag here
    \ialign\bgroup
      \hfil
      \strut@
      $\m@th\displaystyle {##}%
      \UseTaggingSocket{math/luamml/save/nNn}{ {} \displaystyle {mtd} }%
      $%
      \UseTaggingSocket{math/luamml/mtable/finalizecol}{last}%
      &$\m@th\displaystyle {{}##}%
      \UseTaggingSocket{math/luamml/save/nNn}{ {} \displaystyle {mtd} }%
      $%
      \UseTaggingSocket{math/luamml/mtable/finalizecol}{last}%
      \hfill % Why not \hfil?---dmj, 1994/12/28
      \crcr
}
```

(*End of definition for* `\insplit@`. *This function is documented on page* **??**.)

And now the difficult part. Depending on the options leqno/reqno `\lendsplit@` or `\rendsplit@` are used for the typesetting and the inner table must be finalized here in case this hasn't happen yet. This must be tested with the boolean from `\gather@split`

```
\def\lendsplit@{%
    \global\setbox9\vtop{\unvcopy\z@}%
    \ifinalign@
        \setbox\@ne\vbox{%
            \unvcopy\z@
            \global\setbox8\lastbox
        }%
        \setbox\@ne\hbox{%
            \unhcopy8%
            \unskip
            \setbox\tw@\lastbox
            \unskip
            \global\setbox\thr@@\lastbox
        }%
        \__math_split_debug_typeout:n{lendsplit@/aligncase}
        \ifctagsplit@
         \__math_split_debug_typeout:n{lendsplit@/aligncase/centertags}
            \gdef\split@{%
                 \hbox to\wd\thr@@{}%
               &\vcenter{\vbox{\moveleft\wd\thr@@\box9}}%
               \__math_split_debug_typeout:n{finalize~innertable~aligncase}
               \UseTaggingSocket{math/luamml/mtable/innertable/finalize}%
            }%
        \else
        %TODO, not correct yet.
         \__math_split_debug_typeout:n{lendsplit@/aligncase/tbtags}
            \gdef\split@{%
                 \hbox to\wd\thr@@{}%
               &\vbox{\moveleft\wd\thr@@\box9}%
               \__math_split_debug_typeout:n{finalize~innertable~aligncase}
               \UseTaggingSocket{math/luamml/mtable/innertable/finalize}%
            }%
        \fi
    \else
```

If not in align we need to test for `\gather@split`

```
        \ifctagsplit@
           \bool_if:NTF \l__math_gathersplit_bool
            {
              \__math_split_debug_typeout:n{lendsplit/equationcase/centertags}
              \gdef\split@%
               {\UseTaggingSocket{math/luamml/annotate/false}{}{\vcenter{\box9}}}
            }
            {
              \__math_split_debug_typeout:n {lendsplit/gathercase/centertags}
              \gdef\split@{\vcenter{\box9}%
              \__math_split_debug_typeout:n {finalize~innertable~gathercase}
              \UseTaggingSocket{math/luamml/mtable/innertable/finalize}}
            }
        \else
```

```
          % TODO tbtags not correct yet.
           \bool_if:NTF \l__math_gathersplit_bool
            {
              \__math_split_debug_typeout:n {lendsplit/equationcase/tbtags}
              \gdef\split@%
               {\UseTaggingSocket{math/luamml/annotate/false}{}{\box9}}
            }
            {
              \__math_split_debug_typeout:n {lendsplit/gathercase/tbtags}
              \gdef\split@{
                \box9%
                \__math_split_debug_typeout:n {finalize~innertable~gathercase}
                \UseTaggingSocket{math/luamml/mtable/innertable/finalize}}
            }
        \fi
    \fi
    \aftergroup\split@
}
```

`\rendsplit@` And more or less the same for the `\rendsplit@` environment.

```
\def\rendsplit@{%
    \ifinalign@
        \global\setbox9 \vtop{%
            \unvcopy\z@
            \global\setbox8 \lastbox
            \unskip
        }%
        \setbox\@ne\hbox{%
            \unhcopy8
            \unskip
            \global\setbox\tw@\lastbox
            \unskip
            \global\setbox\thr@@\lastbox
        }%
        \ifctagsplit@
            \gdef\split@{%
                \hbox to\wd\thr@@{}
                &\vcenter{\vbox{\moveleft\wd\thr@@\boxz@}}%
            \__math_split_debug_typeout:n {rendsplit/aligncase/centertags}
            \__math_split_debug_typeout:n {finalize~innertable~aligncase}
            \UseTaggingSocket{math/luamml/mtable/innertable/finalize}
            }%
        \else
           \__math_split_debug_typeout:n{rendsplit@/aligncase/tbtags}
           %TODO tbtags is not correct yet
            \global\setbox7 \hbox{\unhbox\tw@\unskip}%
            \gdef\split@{%
                \global\@tempcnta\column@
               &\setboxz@h{}%
                \savetaglength@
                \global\advance\row@\@ne
                \vbox{\moveleft\wd\thr@@\box9}%
               \crcr
```

```
                \noalign{\global\lineht@\z@}%
                \add@amps\@tempcnta
                \UseTaggingSocket{math/luamml/annotate/false}{}{\box\thr@@}
               &\box7
                 \__math_split_debug_typeout:n {finalize~innertable~aligncase}
                 \UseTaggingSocket{math/luamml/mtable/innertable/finalize}%
            }%
        \fi
```

and again the if we are not in align we need to test for `\gathersplit`

```
    \else
        \ifctagsplit@
          \bool_if:NTF \l__math_gathersplit_bool
            {
              \__math_split_debug_typeout:n {rendsplit/equationcase/centertags}
              \gdef\split@%
               {\UseTaggingSocket{math/luamml/annotate/false}{}{\vcenter{\boxz@}}}
            }
            {
              \__math_split_debug_typeout:n  {rendsplit/gathercase/centertags}
              \gdef\split@{\vcenter{\boxz@}%
               \__math_split_debug_typeout:n {finalize~innertable~gathercase}
               \UseTaggingSocket{math/luamml/mtable/innertable/finalize}}
            }
        \else
           \bool_if:NTF \l__math_gathersplit_bool
            {
              \__math_split_debug_typeout:n {rendsplit/equationcase/tbtags}
              \gdef\split@{%
                \UseTaggingSocket{math/luamml/annotate/false}{}{\boxz@}}
            }
            {
              \__math_split_debug_typeout:n {rendsplit/gathercase/tbtags}
              \gdef\split@{%
                \boxz@
                \UseTaggingSocket{math/luamml/mtable/innertable/finalize}}
            }
        \fi
    \fi
    \aftergroup\split@
}
```

(*End of definition for* `\rendsplit@`*. This function is documented on page* **??**.)

2.7 \intertext

The `\intertext` command errors with active tagging as it is processed twice which leads to duplicated structures.

`\intertext@` TODO: review and add sockets!

```
\def\intertext@{%
  \def\intertext##1{%
```

```
    \ifvmode\else\\\@empty\fi
    \noalign{%
      \penalty\postdisplaypenalty\vskip\belowdisplayskip
      \vbox{
```

Stop tagging when measuring:

```
       \ifmeasuring@\tag_suspend:n{\measuring}\fi
       \normalbaselines
        \ifdim\linewidth=\columnwidth
        \else \parshape\@ne \@totalleftmargin \linewidth
        \fi
```

End the previous mc:

```
        \tag_mc_end_push:
```

We are already in a par so we change now to text:

```
        \tagpdfsetup{para/tag=P}%
```

TODO why `\tagpdfparaOn` needed?

```
        \tagpdfparaOn
        \noindent\ignorespaces##1\par
```

Restart the MC

```
        \tag_mc_begin_pop:n{}%
      \penalty\predisplaypenalty\vskip\abovedisplayskip%
    }%
}}
```

(*End of definition for* `\intertext@`. *This function is documented on page* **??**.)

2.8 \text

The `\text` command uses `\mathchoice` which "typesets" the argument four times. This makes it quite problematic for tagging. Without precautions structure objects would be created four times and would get MC-chunks as kids that don't really exist. amsmath contains a switch that allows to execute code only in the first (displaymath) branch, but that isn't usable here. At first because we don't know if the first branch creates the same structure as the one that is actually used. At second because the engines executes some commands like `\label` and `\pdfannot` only at shipout from the branch that really was used. So we would get structure data from one `\mathchoice`-branch and MC-labels and links from another one and that gets very messy.

We therefore have to avoid that tagging is active in unused branches. In pdflatex it is not possible to detect the mathstyle before, so we use a label. With lualatex is is possible to redefine `\text` not to use `\mathchoice`

`\text@`

```
\AddToHook{package/amstext/after}
 {
```

currently amsmath is loaded in a begindocument hook, so this test is fine. If amstext is loaded earlier (in the kernel), this needs perhaps a change.

```
  \tag_if_active:T
   {
     \sys_if_engine_luatex:TF
      {
        \def\text@#1{
         \tag_socket_use:nnn {math/luamml/hbox}{}
         {{%
          \ifcase\mathstyle
          \hbox{{#1}}\or
          \hbox{{#1}}\or
          \hbox{{#1}}\or
          \hbox{{#1}}\or
          \hbox{{\let\f@size\sf@size\selectfont#1}}\or
          \hbox{{\let\f@size\sf@size\selectfont#1}}\or
          \hbox{{\let\f@size\ssf@size\selectfont#1}}\or
          \hbox{{\let\f@size\ssf@size\selectfont#1}}\or
          \ERROR
          \fi
          \check@mathfonts
        }}}
      }
      {
        \def\text@#1
         {{
          \int_gincr:N\g__math_mathchoice_int
          \tag_suspend:n{\text@}
          \mathchoice
           {
            \__math_tag_if_mathstyle:en{mathchoice-\int_use:N\g__math_mathchoice_int}{0}
            \textdef@\displaystyle\f@size{#1}
           }
           {
            \__math_tag_if_mathstyle:en{mathchoice-\int_use:N\g__math_mathchoice_int}{2}
            \textdef@\textstyle\f@size{\firstchoice@false #1}
           }
           {
            \__math_tag_if_mathstyle:en{mathchoice-\int_use:N\g__math_mathchoice_int}{4}
            \textdef@\textstyle\sf@size{\firstchoice@false #1}
           }
           {
             \__math_tag_if_mathstyle:en{mathchoice-\int_use:N\g__math_mathchoice_int}{6}
             \textdef@\textstyle \ssf@size{\firstchoice@false #1}
           }
           \check@mathfonts
         }}
      }
   }
}
```

(*End of definition for* `\text@`. *This function is documented on page* ??.)

2.9 \pmb

\pmb prints its argument three times. For tagging we must mark two of occurrences as artifact. For luatex the attributes in the box must be reset, for this we switch to expl3-boxes.

\pbm@@
\pmb@

```
\AddToHook{package/amsbsy/after}
 {
   \def\pmb@@#1#2#3{\leavevmode\hbox_set:Nn\l__math_tmpa_box{xxx#3}
      \dimen@-\box_wd:N\l__math_tmpa_box
      \kern-.5\ex@\box_use:N\l__math_tmpa_box
      \tag_mc_end:\tag_mc_begin:n{artifact}
      \tag_mc_reset_box:N\l__math_tmpa_box
      \kern\dimen@\kern.25\ex@\raise.4\ex@\box_use:N\l__math_tmpa_box
      \kern\dimen@\kern.25\ex@\box_use_drop:N\l__math_tmpa_box
      \tag_mc_end:\tag_mc_begin:n{}
    }
   \def\pmb@#1#2{\hbox_set:Nn\l__math_tmpa_box{$\m@th#1{#2}$}
     \setboxz@h{$\m@th#1\mkern.5mu$}\pmbraise@\wdz@
     \binrel@{#2}
     \dimen@-\box_wd:N\l__math_tmpa_box
     \binrel@@{
       \mkern-.8mu\box_use:N\l__math_tmpa_box
       \tag_mc_end:\tag_mc_begin:n{artifact}
       \tag_mc_reset_box:N\l__math_tmpa_box
       \kern\dimen@\mkern.4mu\raise\pmbraise@\box_use:N\l__math_tmpa_box
       \kern\dimen@\mkern.4mu\box_use_drop:N\l__math_tmpa_box
       \tag_mc_end:\tag_mc_begin:n{}
       }
    }
  }
```

(*End of definition for* \pbm@@ *and* \pmb@. *These functions are documented on page* **??**.)

```
\ExplSyntaxOff
```

⟨/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.

S

T

U

V

W