

# A Brief Introduction to Regular GELLMU

*William F. Hammond*

Department of Mathematics & Statistics

The University at Albany

Albany, NY 12222, USA

September 2000

Last revision: January 10, 2010

(still needs revision as well as a better proof reading)

D R A F T

**Note:** Since this document was written, there have been some enhancements to the system it describes. For more information see the GELLMU *Manual*<sup>1</sup>.

## Table of Contents

1	Another Mark-Up Language? .....	1
2	How did GELLMU arise?.....	3
3	Translation of GELLMU Markup.....	4
4	Advanced GELLMU vs. Basic GELLMU .....	4
5	Some Fundamentals on Regular GELLMU .....	6
5.1	The Markup Commands .....	6
5.2	Multiple Argument/Option Syntax.....	8
5.3	Mathematics.....	9
5.4	Examples.....	10
6	Notes .....	11

## 1 Another Mark-Up Language?

The purpose of *regular* GELLMU<sup>2</sup> is to provide a markup language with provision for mathematics under a system design that is *structured enough* to admit rigorous, modularly-staged, automatic processing into many other formats while at the same time being a comfortable input format for those accustomed to using L<sup>A</sup>T<sub>E</sub>X. There is not a set list of translation target formats. Examples of possible translation targets include:

<sup>1</sup>URI: <http://www.albany.edu/~hammond/gellmu/glman/glman.html>

<sup>2</sup>GELLMU stands for “Generalized Extensible L<sup>A</sup>T<sub>E</sub>X-like Markup”

- L<sup>A</sup>T<sub>E</sub>X .
- HTML.
- HTML as enhanced by MATHML.
- TEXINFO.
- ME (a \*-roff package).
- `text/plain` (for optimal “de-TeXing”).
- audio stream.
- search engine.
- index engine.

GELLMU is not L<sup>A</sup>T<sub>E</sub>X although its style of markup resembles that of L<sup>A</sup>T<sub>E</sub>X. An example of how there is provision for “enrichment of information” lies in the markup used for a quoted phrase in the source<sup>3</sup> for this document.<sup>4</sup> Whereas the usual L<sup>A</sup>T<sub>E</sub>X way of writing the phrase in the last sentence would be

```
‘‘enrichment of information’’,
```

and that markup would be meaningful in regular GELLMU, the markup

```
\quophrase{enrichment of information}
```

can be detected more easily by a processor looking for *quophrase* contents, while the separate double-quoting indicators may or may not be adequate for such a processor seeking to glean similar information. Furthermore, the effort by a processor to glean such information might be spoiled by the occurrence of one or more unbalanced double-quoting indicators. Possibly even more useful ways of marking the same phrase that still would yield the same printed appearance, depending on the overall context, might be:

```
\jargon{enrichment of information}
```

or

```
\topic{enrichment of information}.
```

An important characteristic of this markup is that the processing of a document always begins with a faithful translation to an instance under the umbrella of “Standard Generalized Markup Language (SGML)”. It is widely known in the text processing community (though not widely observed in the mathematical community) that sufficiently structured markup languages under SGML can be robustly converted to many other formats. GELLMU has been designed to provide a bridge from LaTeX-like markup to the world of SGML.

<sup>3</sup>URI: <http://www.albany.edu/~hammond/gellmu/igl/iglm.glm>

<sup>4</sup>The base for relative URI references given here may be found at the end of the printed form of this document.

The translation from source markup to SGML markup is performed by my program called the GELLMU Syntactic Translator. Once that translation is performed the author's document exists in the world of SGML, and in that world many programs, almost all of which are very highly configurable, are available for automatic processing.

Beyond the syntactic translation from source markup to SGML the first release of materials from this project will contain a didactic *article* SGML document type, and some of the statements herein are, for illustration, non-explicit allusions to the article document type, which is just one example of an SGML document type in the *advanced* GELLMU category.

Realistically today, SGML is a document standard for in-house work, and its "subset" XML<sup>5</sup> has become the only shareable form of SGML. Major SGML document types such as *DocBook* and *TEI* have canonical translations to XML. Still one does not abandon SGML in an authoring environment, because some of the features of full SGML provide substantial convenience for authors.

What added convenience there is in using L<sup>A</sup>T<sub>E</sub>X-like markup as a writing interface for XML is greatly aided by taking advantage of a few of the features in full SGML that are not available in XML. On the other hand, *basic* GELLMU is available for the direct preparation of an XML document as the direct output of the syntactic translator.

## 2 How did GELLMU arise?

Aside from markup languages under SGML another example of a markup language that is structured enough to admit robust conversion is TEXINFO, which is the markup language of The GNU<sup>6</sup> Documentation System. The "Info" side of TEXINFO is the *Info* hypertext self-documentation system within the popular fully programmable editing system *GNU Emacs*. Documents authored in TEXINFO can be automatically converted into both *Info* (for viewing in an *GNU Emacs* buffer) and T<sub>E</sub>X (for printing).

The *Info* system is a hypertext system that pre-dates the World Wide Web (WWW) and the latter's default presentation markup language known as HTML, which falls under the SGML regime. One should, therefore, not be surprised to learn that with the dawn of HTML Lionel Cons of CERN (the birthplace of the World Wide Web) furnished the world with a PERL program for converting a TEXINFO document into HTML with (internal) hypertext references.

Although TEXINFO is a T<sub>E</sub>X-like markup language, it makes wide use of the character @ as command-introducer where in T<sub>E</sub>X one would (normally) use \. It seems that many authors find this superficial difference uncomfortable. Ulrik Vieth of The T<sub>E</sub>X Users Group (TUG)<sup>7</sup> wrote an *Emacs Lisp* program that served to convert into TEXINFO a L<sup>A</sup>T<sub>E</sub>X file written in January 1998 as the draft report<sup>8</sup> of the TUG Working Group on a directory structure for T<sub>E</sub>X systems.

This example, which was an *ad hoc* conversion, played an important role in my arrival at this design for a bridge from L<sup>A</sup>T<sub>E</sub>X to SGML.

---

<sup>5</sup> *Subcategory* might be a better term here than *subset*.

<sup>6</sup>URI: <http://www.gnu.org/>

<sup>7</sup>URI: <http://www.tug.org/>

<sup>8</sup>URI: <ftp://ctan.tug.org/tex-archive/tds/draft-standard/tds-0.9995.tar.gz>

### 3 Translation of GELLMU Markup

GELLMU is essentially, but for appearance, a markup language under SGML. This means that after a document is translated by the syntactic translator into the corresponding markup language falling formally under SGML, any general-purpose SGML processing tool can be used.

The project’s prototype production system is built around *Regular* GELLMU, which uses features for enhanced authoring convenience in the syntactic translator that require some assumptions about the SGML document type under which the author is writing. *Basic* GELLMU, on the other hand, may be used as a simple L<sup>A</sup>T<sub>E</sub>X-like markup interface with a large number of document types, including XML document types.

Two SGML processing tools that are easily available form the core of the prototype production system.

1. James Clark’s `nsgmls`, an SGML parser that is part of his `SP` library package<sup>9</sup>.
2. David Megginson’s PERL program `sgmlspl` that is an interface for his Perl library `SGML-SPM`. It may be found at CPAN<sup>10</sup> or at his office<sup>11</sup>. `sgmlspl` is a programmer-friendly interface for writing simple or complex “event-driven” translators (to other formats) that operate on `nsgmls` output.

In the prototype production system a document under the didactic *article* SGML document type made with the GELLMU Syntactic Translator from L<sup>A</sup>T<sub>E</sub>X-like source markup is parsed by `nsgmls` and then passed through a simple `sgmlspl` translation to yield an instance under the corresponding XML document type.

In the prototype production system the XML version of an article may be translated either to HTML or to L<sup>A</sup>T<sub>E</sub>X. Other translations are possible. The two existing translations are done with `sgmlspl`.

But a number of other freely available translation tools may be considered for the XML version of a GELLMU article including:

1. James Clark’s `jade`, an engine for DSSSL with a wide variety of possible translation targets.
2. James Clark’s `xt`, an XSLT engine also with a wide variety of possible translation targets.
3. David Carlisle’s `xmltex`, a package for *TeX*, the Program, that enables its user to write what amounts to a “style sheet” in T<sub>E</sub>X, rather than, say, XSL, for formatting (with its own parse) an instance of an XML document type in T<sub>E</sub>X (actually DVI since one does not get a chance to “see” a T<sub>E</sub>X source image).

### 4 Advanced GELLMU vs. Basic GELLMU

This document is about the *regular* GELLMU. It is important first to know a bit about *basic*

---

<sup>9</sup>URI: <ftp://ftp.jclark.com/pub/sp/>

<sup>10</sup>URI: [http://www.cpan.org/modules/by-authors/David\\_Megginson/](http://www.cpan.org/modules/by-authors/David_Megginson/)

<sup>11</sup>URI: <http://www.megginson.com/>

mode and *advanced* mode. What is called *regular* in this document is an instance of *advanced* mode. The term *regular* is being used here to reference the didactic *article* document type, which is an SGML document type that admits the more elaborate and succinct handling of *advanced* mode while not being in any way incompatible with *basic* mode.

Neither the basic nor the advanced mode involves in any way adoption of the language of L<sup>A</sup>T<sub>E</sub>X. (But many command names under the didactic document type, mimic L<sup>A</sup>T<sub>E</sub>X command names.)

To use either *basic* or *advanced* mode one must be familiar with the SGML document type for which one is writing. Ordinary HTML is an example where the use of *basic* mode is indicated. Only a very few of the features in advanced markup not part of basic markup<sup>12</sup> make any sense for use in the direct preparation of HTML with L<sup>A</sup>T<sub>E</sub>X-like input.

Basic GELLMU uses the L<sup>A</sup>T<sub>E</sub>X special characters ‘\’, ‘{’, and ‘}’ along with L<sup>A</sup>T<sub>E</sub>X-like argument/option syntax, where braces immediately following a command name indicate command arguments and square brackets, i.e., ‘[’ and ‘]’, indicate command options. A command corresponds to an SGML element, and in basic mode a command may have at most one argument, the content of which corresponds to SGML element content, and at most one option, the content of which corresponds to a list of SGML attribute specifications. Thus for example, in basic mode for HTML one may use the markup

```
\a[href="http://www.w3.org/"]{The World Wide Web Consortium}
```

to form the HTML anchor:

```
<a href="http://www.w3.org/">The World Wide Web Consortium</a> .
```

(The formation of anchors with the didactic document type in advanced mode is slightly more complicated because the characters ‘=’ and ‘/’, which may acquire special (and “overloaded”) semantic significance in mathematical contexts, are held for delayed evaluation as empty elements and because the syntactic translator, which does not recognize command names, regards this usage in advanced mode as *multiple* argument/option syntax, which is not part of basic mode.)

An example of the distinction between basic and advanced GELLMU is that in advanced mode it is possible and easy to arrange to have a blank line, as in L<sup>A</sup>T<sub>E</sub>X, represent the beginning of a paragraph. In basic mode for classical HTML one must<sup>13</sup> use “\p” to begin a paragraph, and for the XML version of HTML one must also provide markup for the end of every paragraph, which may be done in several ways.

For some of the details on using the basic markup with HTML see *Using the GELLMU Syntactic Translator to Write HTML*<sup>14</sup>.

---

<sup>12</sup>With several minor exceptions, one related to the direct writing of SGML attributes (which cannot contain markup and which do not have many parallels in L<sup>A</sup>T<sub>E</sub>X) and another related to the way of escaping the character ‘\’, everything about basic mode also applies to advanced mode.

<sup>13</sup>There is a way, with the setting of several variables for the syntactic translator, to have blank lines begin new paragraphs in basic input for HTML

<sup>14</sup>URI: <http://www.albany.edu/~hammond/gellmu/ghtml.html>

## 5 Some Fundamentals on Regular GELLMU

### 5.1 The Markup Commands

There are several kinds of commands:

- **A maximal string introduced by `\` followed by a letter<sup>15</sup> and otherwise consisting only of letters and numbers. The first number, if any, must not be “0” since such names are reserved for use by the syntactic translator.**

Command names are case sensitive. Such a command is a *container*, corresponding to an SGML *element*, if it is immediately followed, without intervening white space, by the character `{`. In that case the delimited zone of containment normally ends with the subsequent “balancing character” `}`. (L<sup>A</sup>T<sub>E</sub>X-like multiple argument/option chains deserve more discussion; for now it will suffice to point out that the use of the `\anch` command in this document for making “anchors” is an example, and, of course, L<sup>A</sup>T<sub>E</sub>X’s `\frac` command is another example. For the present discussion these commands are considered to be containers.) Such a command corresponds to an SGML *defined-empty element* if it is followed immediately without intervening white space by the character `;`. In any other case there is syntactical ambiguity. However, the syntactic translator will produce a corresponding SGML open tag unless the logical variable *gellmu-xml-strict* has been set. (This variable is normally not set for advanced GELLMU.)

- **Certain single characters.**

`\`, `{`, `}`, `^`, `_`, `$`, `%`, and `~` have command meanings that are similar to their meanings in L<sup>A</sup>T<sub>E</sub>X. The characters `;` and `:` are ordinary characters that have special meaning at the end of a command name. The character `#` is also a special character used, as with L<sup>A</sup>T<sub>E</sub>X, in *newcommand* templates. In source for the didactic *article* document type any non-alpha-numeric ASCII character may be escaped (referenced for itself) with a named command, e.g. `%` may be referenced for itself as `\pct;`. This is **necessary** in order to provide delayed evaluation for ultimate translation to one of many possible ultimate formats.

The following language meanings apply to both basic and regular markup:

1. **"\`\`": Command introducer.** Escape in basic mode: `\\`. This escape is incorrect in regular mode. For the didactic *article* document type the escape is `\bs1;`. For other document types one may resort to an entity reference if there is an aversion to the provision of a corresponding defined-empty element or if one lacks control of the document type.
2. **"{": Command argument opener.** Escape: `\{`.

---

<sup>15</sup>For the first alpha release “letter” means something matching the regular expression “[A-Za-z]”. This will not be a limitation of the syntactic translator after the alpha releases. I believe that *GNU Emacs*, from version 20, is fully capable of meeting the needs of GELLMU for supporting international character sets. At present, it is possible to use *Emacs* “word constituent characters” in names defined with *newcommand*, and there is no reason why the definition strings cannot involve standard SGML entity notation. It is also possible to define symbolic SGML entities in the SGML internal declaration subset, which is the content of an option (at most one) that follows the unique argument to the meta-command `\documenttype`. Additionally, the SGML version of the didactic *article* document type provides commands for the document preamble that may be used to construct an internal declaration subset for the XML version of the document. (Of course, someone using this markup to edit directly for a document type under XML would need to use the `\documenttype` option provided for the SGML internal declaration subset.)

3. "}" : **Command argument closer**. Escape: "\}".
4. "[" : **Command option opener**.
5. "]" : **Command option closer**.
6. ";" : **Command terminator** for commands corresponding to defined-empty SGML elements; otherwise an ordinary character. Its use as a command terminator is invisible and may be omitted optionally in some contexts. This syntax is analogous to the use of ‘;’ as an entity reference terminator in HTML.
7. ":" : **Special purpose command terminator** used for indicating the close of a command-zone; otherwise an ordinary character. Its use as a command terminator is invisible.
8. "%" : **Comment introducer** (in force to end of line).

Any command is terminated by a non-alpha-numeric character. There is syntactic ambiguity unless the terminator is one of ‘{’, ‘[’, ‘;’, or ‘:’. This kind of syntactic ambiguity is not permitted in the direct editing for an XML document type with GELLMU input. The terminator can be a blank space, but, if so, the blank space becomes invisible.

The following language meanings apply to regular markup with allusion to the didactic *article* document type.

1. "~" : **Non-breaking interword space**. Equivalent: \nbs; , cf. &nbsp; in HTML.
2. "^" : **Superscript command**. Equivalent: \sup.
3. "\_" : **Subscript command**. Equivalent: \sub.
4. "&" : **Dual use: tabular cells and entity introducer**. ‘&’ introduces an entity reference if it is followed by anything other than white space. It is used, as in L<sup>A</sup>T<sub>E</sub>X, as a *tabular* cell delineator when it is followed by white space.
5. "\$" : **Toggle inline math mode**.

Equivalent:

"\tmath{ . . . }"

Nearly equivalent:

"\( . . . \)" or  
"\math{ . . . }".<sup>16</sup>

- **Certain strings of plain text in regular GELLMU that are part of legacy practice under L<sup>A</sup>T<sub>E</sub>X.**

1. "--"
  - A short dash** as used with a range of numbers, e.g., 1–2. Equivalent: \rdash.
2. "---"
  - A long dash** as used for punctuation, e.g., a dash — like this. Equivalent: \pdash.
3. "\ "
  - Blank interword space**. Equivalent: \spc;
4. "\,"
  - Small horizontal space**. Equivalent: \hsp;

---

<sup>16</sup>It is possible to merge the inline *math* and *tmath* zones at any level of processing beyond the syntactic translator. (These are indeed the same in L<sup>A</sup>T<sub>E</sub>X.) The syntactic translator resists the temptation here to go beyond syntax, and with the didactic *article* document type the formatting to L<sup>A</sup>T<sub>E</sub>X inserts the L<sup>A</sup>T<sub>E</sub>X markup "\," for a small horizontal space before and after *math*, but not before and after *tmath*.

5. "\\\"
 

**A forced line break.** "\\\" may be used at the end of a line of input for a forced line break. In a *tabular* environment (with the didactic *article* document type, as in L<sup>A</sup>T<sub>E</sub>X) it begins a new *tabular* row. Any other use is deprecated, and will result in translation to the defined-empty element *bsl* corresponding to the ASCII character ‘\’ with a warning from the syntactic translator. The use of *tabular* is deprecated and is not supported beyond “lrc”. Use *table*, which is not similar to L<sup>A</sup>T<sub>E</sub>X’s *table*, instead of *tabular*. Equivalents: `\brk`; for a line break outside of a *tabular* environment.<sup>17</sup>
  6. A blank line.
 

**Begin new paragraph command.** Equivalent: `\parb`. Nearly equivalent: `\par`.
  7. "““"
 

**Left (double) quotation mark.** Equivalent: `\ldq`.
  8. "’’"
 

**Right (double) quotation mark.** Equivalent: `\rdq`.
  9. "\("
 

**Begin math mode command.** Equivalent: `\begin{math}`.
  10. "\)"
 

**End math mode command.** Equivalent: `\end{math}`.
  11. "\["
 

**Begin displaymath mode command.** Equivalent: `\begin{displaymath}`.
  12. "\]"
 

**End displaymath mode command.** Equivalent: `\end{displaymath}`.
- `\begin{name} . . . \end{name}`

This usage is equivalent to `"\name . . . \name:"`, which, in turn, is equivalent to `"\name{ . . . }"`.

## 5.2 Multiple Argument/Option Syntax

An essential point in the present design is that the whole system is built from components, each of which has its own function<sup>18</sup>. Consistent with this design the syntactic translator operates with knowledge of syntax but little or no knowledge of language.

Multiple argument/option syntax has been built into advanced mode as part of the overall idea of providing, where sensible, L<sup>A</sup>T<sub>E</sub>X-like features in a precise user markup interface for writing in document types under SGML and XML.

What are the rules for converting the multiple argument/option syntax in source markup into SGML? Direct conversion by the syntactic translator of this type of usage into XML is not

<sup>17</sup>The syntactic translator simply outputs the SGML defined-empty element *brk0*, which belongs to its reserved name space. The dual use of *brk0* involves some SGML chicanery that is resolved during translation to the XML version of the *article* document type, where *tabular* is converted to *table* and non-tabular use of *brk0* is converted to *brk*.

<sup>18</sup>In the prototype production system based on the didactic *article* document type the output from each stage is available for examination and, where necessary, intervention. However, such use of intervention is intended only for temporary expedient use while a GELLMU system is being designed or enhanced. As with L<sup>A</sup>T<sub>E</sub>X, enhancement is an ongoing process.

available because such conversion requires some language knowledge and the program does not operate with knowledge of language at that level. One obtains an XML version of a document in the prototype production system by using a translator with minimal knowledge of the command vocabulary to create the XML version from an SGML version that is the immediate output of the syntactic translator.

In multiple argument/option syntax, which is much like that of L<sup>A</sup>T<sub>E</sub>X, arguments and options follow command names. Arguments are delimited by braces, i.e., ‘{’ and ‘}’ and options by square brackets, i.e., ‘[’ and ‘]’. There must be no white space between the arguments and options nor between the command name and the first member of an argument/option sequence.

Each command with a multiple argument/option sequence is translated to an open tag whose name is the name of the command. Each argument is translated to an *agθ* element and each option to an *opθ* element lying in GELLMU’s reserved name space. There are two exceptional cases.

1. The first argument or option is an option inside which the very first character is a colon, i.e., ‘:’. This is the method provided in advanced mode for the direct entry of an SGML attribute sequence. The entire contents of the option string, apart from the leading ‘:’, which is discarded, are understood to be a sequence of SGML attributes for the SGML element whose name is the name of the command. There is no syntax check of the attribute contents by the syntactic translator. Such an *attribute option* is not treated as an *opθ* element. In particular, an attribute option is correctly followed immediately by a semi-colon, i.e., the character ‘;’, if and only if the corresponding SGML element is a defined-empty element under the SGML document type. Since SGML attributes correspond to very little of classical L<sup>A</sup>T<sub>E</sub>X, attribute options may be entirely ignored.
2. The first argument is the only argument and there are no options apart from a possible attribute option. This case, which is extremely common, is exceptional relative to argument/option handling since the sole argument simply becomes element content without an *agθ* wrapper.

### 5.3 Mathematics

The Greek letter Γ is marked, up as in L<sup>A</sup>T<sub>E</sub>X, with

`\Gamma`.

The didactic *article* document type provides mathematical markup that is similar to that of L<sup>A</sup>T<sub>E</sub>X. For example, one may use the markup

`\[ 2^{p-1} \equiv 1 \pmod{p^2} \eos\]`

to speak of the congruence

$$2^{p-1} \equiv 1 \pmod{p^2} .$$

In this example the markup "`\eos`" is a formal end-of-sentence<sup>19</sup>.

---

<sup>19</sup>Regular GELLMU recognizes ‘.’ followed by two blank spaces or by a newline as ends-of-sentence markup when these occur outside of mathematical contexts.

In a  $\LaTeX$  math environment, input text “`abc`” is set the same as the the input text “`a b c`” following the presumption that each glyph is a separate symbol. At one point I had planned to provide for a distinction between these two forms of input for *article*. However, I am concerned about author confusion; so there will be provision for having “`abc`” be a math symbol, but an instance will be invoked as “`\abc`”.

In math mode all symbols are assumed to be single-characters as in  $\LaTeX$ . A `\mathsym` command (part of language design that, for the moment, has no extant implementation) in the document preamble may be used to specify that a string is to be regarded as a single mathematical entity whenever it occurs in math mode. This contrasts with  $\LaTeX$  in that there is no formal declaration of math symbols.

*mathsym* will be a variant of *newcommand*, which is a meta-command that is handled entirely by the syntactic translator with no trace in the output. However, the design of *mathsym*, which will also be a meta-command but which will leave marks in the output, is not yet decided. The design issues surround how to provide the author who is its user with flexible open means of planting semantic information in the output without undue verbosity. The syntax might be simply

```
\mathsym{symbol-name} {presentation-definition} [semantic-information-without-markup]
```

Omitting the option could make author usage equivalent to that of *newcommand*, but still trace information planted in the output of the syntactic translator could be sufficient that in the future a derived MATHML object or a derived XML formatted object produced by translation and viewed in *Mozilla*<sup>20</sup> might enable the reader to ascertain the *symbol-name* and then launch a search for other occurrences. The option would serve as a mechanism for the author to pass semantic enrichment information to downstream formatters.

## 5.4 Examples

The following table refers to the didactic *article* document type. It demonstrates how source is translated by the GELLMU Syntactic Translator to the SGML version of *article* and from there is translated using *sgmlspl* to the XML version of *article*.

<b>Source</b>	<code>\emph{this}</code>
<b>SGML</b>	<code>&lt;emph&gt;this&lt;/emph&gt;</code>
<b>XML</b>	<code>&lt;emph&gt;this&lt;/emph&gt;</code>
<b>Source</b>	<code>\tex;</code>
<b>SGML</b>	<code>&lt;tex/&gt;</code>
<b>XML</b>	<code>&lt;tex/&gt;</code>
<b>Source</b>	<code>\$ \frac{2}{3} \$</code>
<b>SGML</b>	<code>&lt;tmath&gt; &lt;frac&gt;&lt;ag0&gt;2&lt;/ag0&gt;&lt;ag0&gt;3&lt;/ag0&gt; &lt;/tmath&gt;</code>
<b>XML</b>	<code>&lt;tmath&gt; &lt;frac&gt;&lt;numr&gt;2&lt;/numr&gt;&lt;denm&gt;3&lt;/denm&gt;&lt;/frac&gt;&lt;/tmath&gt;</code>
<b>Source</b>	<code>\anch[href="nil"]{Null}\anch:</code>
<b>SGML</b>	<code>&lt;anch&gt;&lt;op0&gt;href&lt;eqs/&gt;"nil"&lt;/op0&gt;&lt;ag0&gt;Null&lt;/ag0&gt;&lt;/anch&gt;</code>
<b>XML</b>	<code>&lt;anch&gt;&lt;anchref&gt;href="nil"&lt;/anchref&gt;&lt;anchv&gt;Null&lt;/anchv&gt;&lt;/anch&gt;</code>

---

<sup>20</sup>URI: <http://www.mozilla.org/>

One may find many other examples of GELLMU markup in the project archive<sup>21</sup>.

## 6 Notes

Although the meta-command *newcommand*, which is handled internally by the syntactic translator, is now available, the variant meta-command *mathsym* is not yet available.

This document is still at draft stage.

It is itself a GELLMU document. The following versions are available:

- source<sup>22</sup>.
- SGML<sup>23</sup>.
- XML<sup>24</sup>.
- XHTML with MathML<sup>25</sup>.
- classic HTML<sup>26</sup>.
- L<sup>A</sup>T<sub>E</sub>X<sup>27</sup>.
- DVI<sup>28</sup>.

---

<sup>21</sup>URI: <http://www.albany.edu/~hammond/gellmu/>

<sup>22</sup>URI: <http://www.albany.edu/~hammond/gellmu/igl/iglm.glm>

<sup>23</sup>URI: <http://www.albany.edu/~hammond/gellmu/igl/iglm.sgml>

<sup>24</sup>URI: <http://www.albany.edu/~hammond/gellmu/igl/iglm.xml>

<sup>25</sup>URI: <http://www.albany.edu/~hammond/gellmu/igl/iglm.xhtml>

<sup>26</sup>URI: <http://www.albany.edu/~hammond/gellmu/igl/iglm.html>

<sup>27</sup>URI: <http://www.albany.edu/~hammond/gellmu/igl/iglm.ltx>

<sup>28</sup>URI: <http://www.albany.edu/~hammond/gellmu/igl/iglm.dvi>