# 将MathML公式转换为域代码公式

在上一篇文章中[将Html导出为Docx](http://shiba.hpe.sh.cn/jiaoyanzu/WULI/Article2317)介绍了如何包含MathML公式的网页转换为Docx文件，但我个人总是嫌Word公式不好看，在整理试卷的时候我用的都是Word自带的域代码公式，所以想在题库导出中添加生成域代码公式的功能。

比方说的域代码公式是**EQ \R(2)**，如何用编程的方式将这个公式插入到Word文件中去呢？可以用Open XML SDK 2.5 Productivity Tool打开包含这个公式的Word文件，找到对应的方法如下：

Paragraph paragraph1 = new Paragraph(){ RsidParagraphAddition = "002A4C03", RsidRunAdditionDefault = "003B4857" };

Run run1 = new Run();

FieldChar fieldChar1 = new FieldChar(){ FieldCharType = FieldCharValues.Begin };

run1.Append(fieldChar1);

Run run2 = new Run();

FieldCode fieldCode1 = new FieldCode(){ Space = SpaceProcessingModeValues.Preserve };

fieldCode1.Text = " EQ \\R(2)";

run2.Append(fieldCode1);

Run run3 = new Run();

FieldChar fieldChar2 = new FieldChar(){ FieldCharType = FieldCharValues.End };

run3.Append(fieldChar2);

paragraph1.Append(run1);

paragraph1.Append(run2);

paragraph1.Append(run3);

非常简单！因此我采用了如下方法实现了域代码公式的插入。

## 初步的尝试

在前端的Html页面中输入域代码公式，由于不想在页面中显示这个信息，我将它放置在**<i>**标签的**data-eq**属性中，即：

<i data-eq=”\R(2)”></i>

然后在提交数据到后台的时候，将这个信息包装在一个自定义的标签<eq>中，即：

<eq>\R(2)</eq>

在后端提取出“**\R(2)**”，由上面的C#代码将域代码公式插入到Word中，可以正常显示。

期间费了好大劲解决了这么一个问题：前端输入题目所用的富文本编辑器是CKEditor，它会自动将空的**<i></i>**标签过滤掉，在网上搜到了答案（具体网址忘了记录，好像是在CKEditor官网论坛上），即在配置文件config.js中加上如下代码：

CKEDITOR.editorConfig = function( config ) {

 ……

 // 允许输入所有元素

 config.allowedContent = {

 $1: {

 elements: CKEDITOR.dtd,

 attributes: true,

 styles: true,

 classes: false

 }

};

 // 过滤掉不允许输入的元素

 config.disallowedContent = 'script; \*[on\*];a;blockquote;body;dd;dl;div;dt;fieldset;form;frame;frameset;h1;h2;h3;h4;h5;h6;noframes;ol;ul;center;dir;hr;menu;pre;li;head';

 // 只允许输入 <i data-eq=\*></i>的形式

 config.protectedSource.push(/<i data-eq=[\s\S]\*?\><\/i>/g);

};

但这种做法有两个很大的问题：

1．在题库中已经输入了几千个以Latex格式的保存的公式，用这个方法意味着还需要输入几千个域代码公式，工作量不小。

2．不能处理包含上，下标以及斜体的公式。

## 目前采用的前端方法

要解决上述第1个问题，思路很简单，因为Html网页中可以提取公式的Latex表达，只要将它转变为域代码公式即可。以为例，就是将“**\(\sqrt 2 \)**”转换成“**\R(2)**”，但自身水平不够不知如何处理。

万幸在<https://stackoverflow.com/questions/26357109/parsing-mathml-to-plain-math-expression>找到了另一个方法，即将MathML公式在前端转换为普通的文本格式，最关键的一个方法为**remove\_tags**，代码如下：

function remove\_tags(node) {

 var result = "";

 var nodes = node.childNodes;

 var tagName = node.tagName;

 if (!nodes.length) {

 if (node.nodeValue === " ")

 result = "";

 else result = node.nodeValue;

 }

 else if (tagName === "mroot") {

 result = "\\R(" + remove\_tags(nodes[1]) + "," + remove\_tags(nodes[0]) + ")";

 }

 else if (tagName === "mfrac") {

 result = "\\F("+remove\_tags(nodes[0])+","+remove\_tags(nodes[1])+")";

 }

 else if (tagName === "msup") {

 result = remove\_tags(nodes[0]) + "^(" + remove\_tags(nodes[1])+")";

 }

 else if (tagName === "msub") {

 result = remove\_tags(nodes[0]) + "\_(" + remove\_tags(nodes[1])+")";

 }

 else if (tagName === "msubsup") {

 result = remove\_tags(nodes[0]) + "\_(" + remove\_tags(nodes[1]) + ")^(" + remove\_tags(nodes[2])+")";

 }

 else for (var i = 0; i < nodes.length; ++i) {

 result += remove\_tags(nodes[i]);

 }

 if (tagName === "msqrt")

 result = "\\R(" + result + ")";

 return result;

}

上述方法利用递归将一段MathML转换为文本格式。不过没有支持所有MathML标签，不支持的原因主要是某些公式在高中物理题库中用不到（比方说矩阵），**mover**标签主要用来表示平均值，例如$\overbar{v}$，但域代码公式太难看，也不支持，遇到这种公式，会以Word公式的形式导出。下表是支持情况：

|  |  |
| --- | --- |
| 元素 | 支持情况 |
| [mrow](https://www.w3.org/TR/MathML3/chapter3.html#presm.mrow) | √ |
| [mfrac](https://www.w3.org/TR/MathML3/chapter3.html#presm.mfrac) | 分数 |
| [msqrt](https://www.w3.org/TR/MathML3/chapter3.html#presm.mroot) | 平方根 |
| [mroot](https://www.w3.org/TR/MathML3/chapter3.html#presm.mroot) | 高次平方根 |
| [mstyle](https://www.w3.org/TR/MathML3/chapter3.html#presm.mstyle) | × |
| [merror](https://www.w3.org/TR/MathML3/chapter3.html#presm.merror) | × |
| [mpadded](https://www.w3.org/TR/MathML3/chapter3.html#presm.mpadded) | × |
| [mphantom](https://www.w3.org/TR/MathML3/chapter3.html#presm.mphantom) | × |
| [mfenced](https://www.w3.org/TR/MathML3/chapter3.html#presm.mfenced) | × |
| [menclose](https://www.w3.org/TR/MathML3/chapter3.html#presm.menclose) | × |
| [msub](https://www.w3.org/TR/MathML3/chapter3.html#presm.msub) | 下标 |
| [msup](https://www.w3.org/TR/MathML3/chapter3.html#presm.msup) | 上标 |
| [msubsup](https://www.w3.org/TR/MathML3/chapter3.html#presm.msubsup) | 上下标 |
| [munder](https://www.w3.org/TR/MathML3/chapter3.html#presm.munder) | × |
| [mover](https://www.w3.org/TR/MathML3/chapter3.html#presm.mover) | × |
| [munderover](https://www.w3.org/TR/MathML3/chapter3.html#presm.munderover) | × |
| [mmultiscripts](https://www.w3.org/TR/MathML3/chapter3.html#presm.mmultiscripts) | × |
| [mtable](https://www.w3.org/TR/MathML3/chapter3.html#presm.mtable) | × |
| [mlabeledtr](https://www.w3.org/TR/MathML3/chapter3.html#presm.mlabeledtr) | × |
| [mtr](https://www.w3.org/TR/MathML3/chapter3.html#presm.mtr) | × |
| [mtd](https://www.w3.org/TR/MathML3/chapter3.html#presm.mtd) | × |
| [mstack](https://www.w3.org/TR/MathML3/chapter3.html#presm.mstack) | × |
| [mlongdiv](https://www.w3.org/TR/MathML3/chapter3.html#presm.mlongdiv) | × |
| [msgroup](https://www.w3.org/TR/MathML3/chapter3.html#presm.msgroup) | × |
| [msrow](https://www.w3.org/TR/MathML3/chapter3.html#presm.msrow) | × |
| [mscarries](https://www.w3.org/TR/MathML3/chapter3.html#presm.mscarries) | × |
| [mscarry](https://www.w3.org/TR/MathML3/chapter3.html#presm.mscarry) | × |
| [maction](https://www.w3.org/TR/MathML3/chapter3.html#presm.maction) | × |
| [math](https://www.w3.org/TR/MathML3/chapter2.html#interf.toplevel) | √ |

最后在stringifyMathML方法中调用**remove\_tags**即可，完整代码如下：

function stringifyMathML(input)

{

 // 只支持<msub><msup><mfrac><msqrt><mroot>等标签

 var reg = /mover|mstyle|merror|mpadded|mphantom|mfenced|menclose|munder|mover|munderover|mmultiscripts|mtable|mlabeledtr|mtr|mtd|mstack|mlongdiv|msgroup|mscarries|mscarry|maction/i;

if (!reg.test(input)) {

 // 删除空字符<mo>&#x2061;<\/mo>

 var strMath = input.replace(/<mo>&#x2061;<\/mo>/g, "");

 parser = new DOMParser();

 xmlDoc = parser.parseFromString(strMath, "text/xml");

 var plainText = remove\_tags(xmlDoc.documentElement);

 // 替换某些字符

 plainText = plainText.replace(/\^\(∘\)/g, "°");

 plainText = plainText.replace(/\^\(′\)/g, "′");

 plainText = plainText.replace(/\^\(″\)/g, "″");

 plainText = plainText.replace(/\^\(‴\)/g, "‴");

 // 拆分字符串，并用#连接为一个字符串

 var out = plainText.match(/\\R\(|\\F\(|\^\(.\*?\)|\_\(.\*?\)|sin|cos|tan|tg|cot|ctg|ln|log|\S/gi);

 var finalOut = out.join("#");

 // 将此字符串包含在<eq>标签中

 $(this).replaceWith("<eq>" + finalOut + "</eq>");

 }

}

例如公式的MathML公式如下：

<math xmlns=”http://www.w3.org/1998/Math/MathML”>

<mroot>

<mrow>

<mfrac>

<mrow>

<msup>

<mi>G</mi>

<mo>&amp;#x2032;</mo>

</msup>

<mrow>

<msub>

<mi>M</mi>

<mn>1</mn>

</msub>

</mrow>

<msubsup>

<mi>T</mi>

<mn>1</mn>

<mn>2</mn>

</msubsup>

<mi>sin</mi>

<mo>&amp;#x2061;</mo>

<msup>

<mn>30</mn>

<mo>&amp;#x2218;</mo>

</msup>

</mrow>

<mrow>

<mn>4</mn>

<mrow>

<msup>

<mi>&amp;#x03C0;</mi>

<mn>2</mn>

</msup>

</mrow>

</mrow>

</mfrac>

</mrow>

<mn>3</mn>

</mroot>

</math>

经过上面的处理输出的代码为：

\R(#3#,#\F(#G#′#M#\_(1)#T#\_(1)#^(2)#sin#3#0#°#,#4#π#^(2)#)#)

其中^表示上标，\_表示下标。

### 代码的解释

* 为什么要删除空字符**<mo>&#x2061;<\/mo>**？

var strMath = input.replace(/<mo>&#x2061;<\/mo>/g, "");

因为在sin、cos、cot、tan之后会出现一个诡异的空字符**&#x2061;**，原因不明，只能简单粗暴地将它删除。

* 为什么要替换某些字符？

plainText = plainText.replace(/\^\(∘\)/g, "°");

plainText = plainText.replace(/\^\(′\)/g, "′");

plainText = plainText.replace(/\^\(″\)/g, "″");

plainText = plainText.replace(/\^\(‴\)/g, "‴");

非常奇怪，度符号“°”、撇号“′”、“″”、“‴”在MathML中竟然是用上标表示的，所以只能手动将它们替换为正常的字符。

* 为什么要拆分字符串，并用#连接为一个字符串？

var out = plainText.match(/\\R\(|\\F\(|\^\(.\*?\)|\_\(.\*?\)|sin|cos|tan|tg|cot|ctg|ln|log|\S/gi);

var finalOut = out.join("#");

这是为了后台代码设置斜体用的。

按照科学论文的规范，通常这样规定：

* 在数学表达式中，按规定**函数、数学符号应用正体表示**。例如，、sin、cos、ln、e、exp、等均为数学符号，因此使用正体。
* 数学式中的**变量、物理量用斜体**。其中一些普适物理常数，例如Boltzmann常数*k*、通用气体常数*R*、Planck常数*h*、Faraday常数*F*等，均用斜体字母表示。
* 用英文词字头表示上下标应用正体字母，用**字母表示条件或编号代号时用斜体字母**，用阿拉伯数字表示编号时，一律为正体。
* 物理量单位用正体。
* ……

挺复杂的，我做了简化，规定如下：

* 英文字母和小写希腊字母用斜体；
* 大写希腊字母和上下标一律用正体。

依据以上原则，要用正则表达式将一个公式拆分成英文字母、数字、上下标等，由于我的正则表达式水平不够，还没达到最完美的拆分方式，但也够用了。

最后用“#”将拆分后的字符拼成一个字符串用于发送到后台，之所以不用常用的“，”作为连接符是因为在公式中“，”有特殊用途。

## 后台代码

后台接收到字符串之后的处理代码如下：

// 找到上传的html代码中的<eq></eq>标签中的内容

HtmlNodeCollection eqs = hdoc.DocumentNode.SelectNodes("//eq");

if (eqs != null)

{

 // 将<eq></eq>替换为"<b>/////<b>"

 HtmlNode newNode4Eq = HtmlNode.CreateNode("<b>/////</b>");

 foreach (HtmlNode eq in eqs)

 {

 eq.ParentNode.ReplaceChild(newNode4Eq, eq);

 }

}

using (MemoryStream memoryStream = new MemoryStream())

{

 doc.WriteByteArray(memoryStream);

 using (WordprocessingDocument wordDoc = WordprocessingDocument.Open(memoryStream, true))

 {

 List<Text> replaceTexts = new List<Text>();

 List<Text> replaceEqs = new List<Text>();

 // 首先找到要插入公式的位置

foreach (var paragraph in wordDoc.MainDocumentPart.Document.Body.Descendants<Paragraph>())

 {

 foreach (var run in paragraph.Descendants<Run>())

 {

 Text text = run.Elements<Text>().Where(p => p.Text == "////").FirstOrDefault();

 Text text4Eq = run.Elements<Text>().Where(p => p.Text == "/////").FirstOrDefault();

 if (text != null)

 replaceTexts.Add(text);

 if (text4Eq != null)

 replaceEqs.Add(text4Eq);

 }

 };

 // 插入普通公式

 if (replaceTexts.Count() > 0)

 {

 int i = 0;

 foreach (var replaceText in replaceTexts)

 {

 foreach (var currentRun in ommls[i].Descendants<DocumentFormat.OpenXml.Math.Run>())

 {

 // Add font information to every run.

 DocumentFormat.OpenXml.Wordprocessing.RunProperties runProperties2 =

 new DocumentFormat.OpenXml.Wordprocessing.RunProperties();

 RunFonts runFonts2 = new RunFonts() { Ascii = "Cambria Math", HighAnsi = "Cambria Math" };

 runProperties2.Append(runFonts2);

 currentRun.InsertAt(runProperties2, 0);

 }

 replaceText.Parent.ReplaceChild(ommls[i], replaceText);

 i += 1;

 }

 }

 // 插入域代码公式

 if (replaceEqs.Count() > 0)

 {

 int j = 0;

 Regex reTriFun = new Regex("sin|cos|tan|cot|ctg|tg");

 Regex reVar = new Regex("^[a-zA-Z]+|^[\u03B1-\u03C9]+"); // 判断是否英文或小写希腊

 Regex reSup = new Regex("^\\^\\(", RegexOptions.IgnoreCase); // 判断开头是否是^((上标)

 Regex reSub = new Regex("^\_\\(", RegexOptions.IgnoreCase); // 判断开头是否是\_((下标)

 foreach (var replaceEq in replaceEqs)

 {

 // 添加域代码公式

 // Begin

 Run run1 = new Run();

 FieldChar fieldChar1 = new FieldChar() { FieldCharType = FieldCharValues.Begin };

 run1.Append(fieldChar1);

 replaceEq.Parent.InsertAfterSelf(run1);

 Run run2 = new Run();

 FieldCode fieldCode2 = new FieldCode() { Space = SpaceProcessingModeValues.Preserve };

 fieldCode2.Text = "EQ ";

 run2.Append(fieldCode2);

 run1.InsertAfterSelf(run2);

 // End

 Run run3 = new Run();

 FieldChar fieldChar3 = new FieldChar() { FieldCharType = FieldCharValues.End };

 run3.Append(fieldChar3);

 run2.InsertAfterSelf(run3);

 string eq = eqs[j].InnerText;

 string[] eqElements = eq.Split('#');

 // Begin和End之间的内容

 for (int i = 0; i < eqElements.Length; i++)

 {

 Run run = new Run();

 // 上标

 if (reSup.IsMatch(eqElements[i]))

 {

 RunProperties runProperties = new RunProperties();

 VerticalTextAlignment verticalTextAlignment = new VerticalTextAlignment() { Val = VerticalPositionValues.Superscript };

 runProperties.Append(verticalTextAlignment);

 run.Append(runProperties);

 // 去掉前两个字符^(，最后一个字符)

 eqElements[i] = eqElements[i].Substring(2, eqElements[i].Length-3);

 }

 // 下标

 else if (reSub.IsMatch(eqElements[i]))

 {

 RunProperties runProperties = new RunProperties();

 VerticalTextAlignment verticalTextAlignment = new VerticalTextAlignment() { Val = VerticalPositionValues.Subscript };

 runProperties.Append(verticalTextAlignment);

 run.Append(runProperties);

 // 去掉前两个字符\_(，最后一个字符)

 eqElements[i] = eqElements[i].Substring(2, eqElements[i].Length - 3);

 }

 // 英文和小写希腊字母，但不包含(sin、cos、……)

 else if (!reTriFun.IsMatch(eqElements[i]) && reVar.IsMatch(eqElements[i]))

 {

 RunProperties runProperties = new RunProperties();

 Italic italic = new Italic();

 runProperties.Append(italic);

 run.Append(runProperties);

 }

 FieldCode fieldCode = new FieldCode() { Space = SpaceProcessingModeValues.Preserve };

 fieldCode.Text = eqElements[i];

 run.Append(fieldCode);

 run3.InsertBeforeSelf(run);

 }

 replaceEq.Parent.Remove();

 j += 1;

 }

 }

}

File.WriteAllBytes("../../finalWord.docx", memoryStream.ToArray());

}

至此大功告成。**还有一个bug无法解决**：“<”“>”会被转义为“&lt;”“&gt;”，不知道在哪个环节出了问题，无法解决。

源代码不包含前端代码。