

SPEEDING UP WITH MATHML*

Sunil Kumar Singh

This work is produced by The Connexions Project and licensed under the Creative Commons Attribution License †

Abstract

Code generation can be speeded up a bit, using macros and proper coding scheme.

Surprisingly, we are not always aware of things available to us. MS Word, for example, has the capability to reproduce a chunk of text with a stroke of key combination. This appears a great facility, particularly for speeding up writing any mark up codes, including MathML. This facility, though available all the time, is hardly ever used in normal text editing session with MS Word.

Using the facility to create macros in MS Word is simple. Matter of fact, this is one of the reasons for recommending an advanced text editing software like MS Word as against bare editors like Notepad for writing plain text MathML code lines.

We need to create macros for writing a set of code lines for the basic coding forms for most of the frequently used element and assign the same to a keyboard combination like ALT a, ALT b etc. Subsequently, when we need to write the code lines for MathML element in the basic form, we shall require to simply use the key combination and the code lines will be reproduced. It is warned that we must provide sufficient space between existing code or text lines so that the reproduced text by macros do not mix up. This technique, involving macros, has the advantage that once a correct format is recorded, it is less likely that there would be any syntax error.

1 Creating macros for MathML basic forms of elements

In this section, the exact steps to create a macro with few code lines involving “mfrac” element are given. Before we proceed, we must realize that choosing a particular size and font to write the codes in macros is extremely important. The renderer accepts MathML in mono font for the code lines. What it means that we should encode our markups in one font only. A mix of many fonts results from “copy” and “paste” operations from different sources. We must avoid font mixing so that the renderer does not report error, while displaying the content.

Now, follow the steps to create a macro for the basic form of “mfrac” element :

Step 1: Open MS Word. Create a new document. Save the document as “test.doc”, ensuring that “save as type” is a word document. Note that we can **not** record and store macros in plain format, which is used for encoding MathML.

Step 2: Select “Courier New” from the font drop down as shown here :

*Version 1.2: Apr 20, 2006 3:27 am GMT-5

†<http://creativecommons.org/licenses/by/2.0/>

Font selection

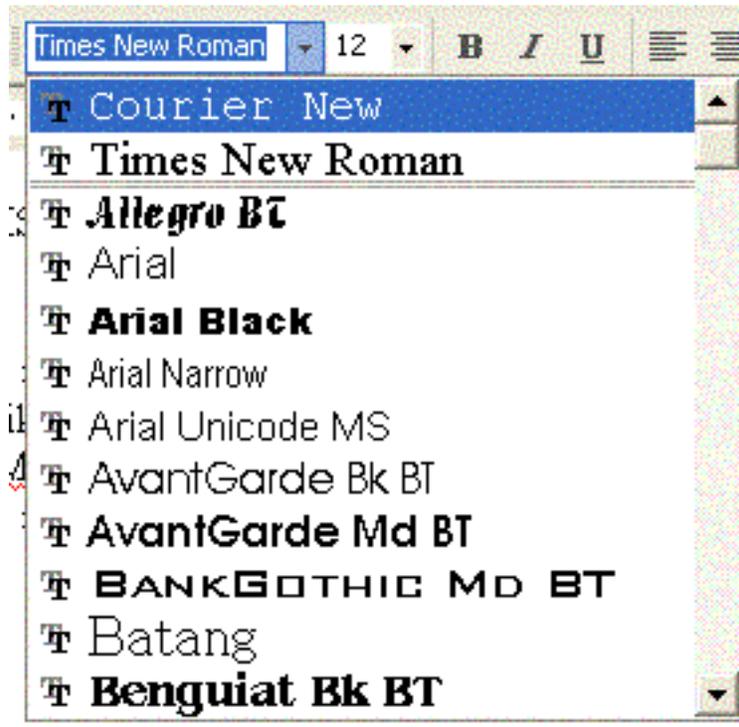


Figure 1

Select font size 12 from the adjacent drop down.

Step 3 : Write code lines for the basic form of the “mfrac” element as shown here :

```
<m:frac>
  <m:mi> x </m:mi>
  <m:mi> y </m:mi>
</m:frac>
```

Step 4 : Select “Tools” from the top tool bar of MS Word and choose “Record New Macro” as shown below :

Recording macros

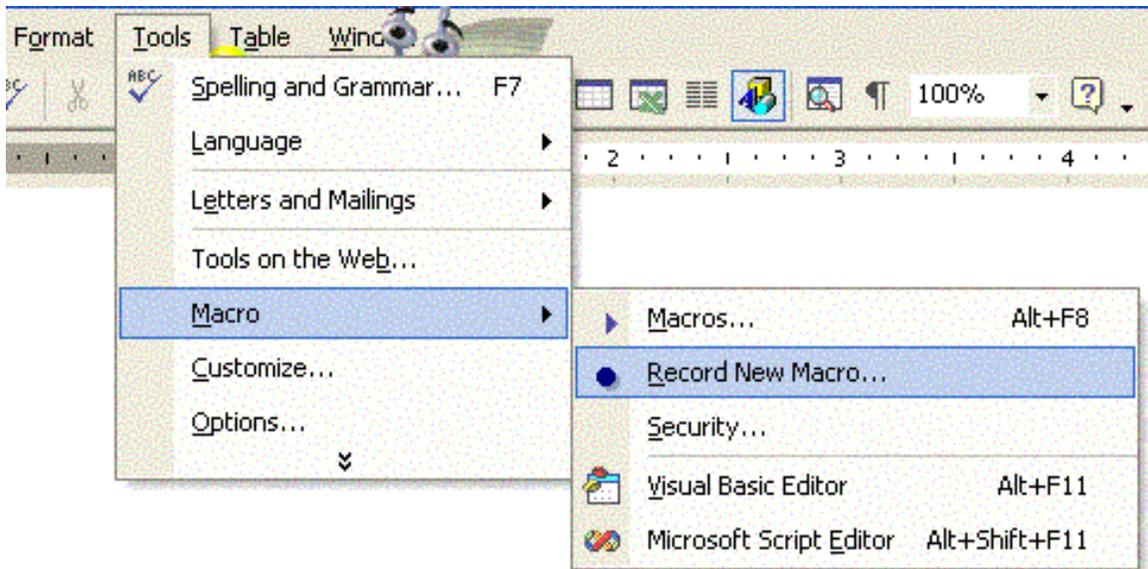


Figure 2

Step 5 : “Record macro” dialog box appears. In the “Macro name” give the name of macro, say “mfrac”. In the “Store macro in” select the name of document “test document” or the name of the document in which you are recording macro.

Setting macros

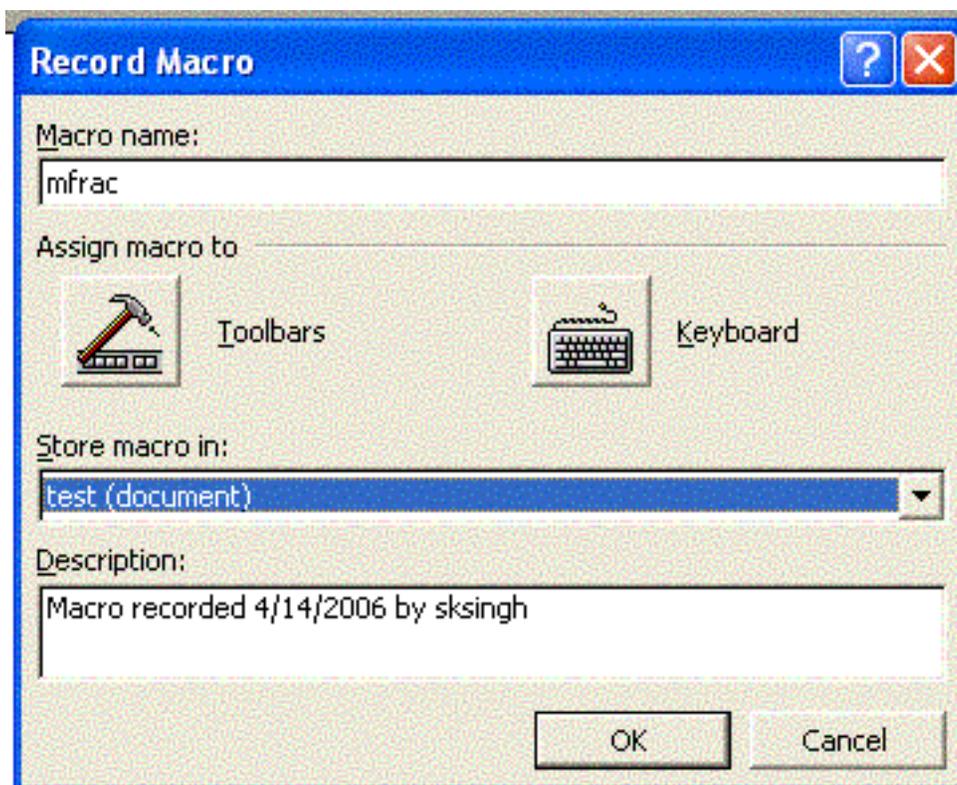


Figure 3

Step 6 : Press “keyboard” button. The “Customize Keyboard” dialog box appears. Put the cursor in the “Press new shortcut key” box (if not placed automatically). Press “Alt and f” to record the desired key combination to run the macro for “mfrac” code lines. Note that “Alt + F” text with capital “F” is automatically placed inside the box. In the “unassigned” box, select “test”.

Assigning key combination

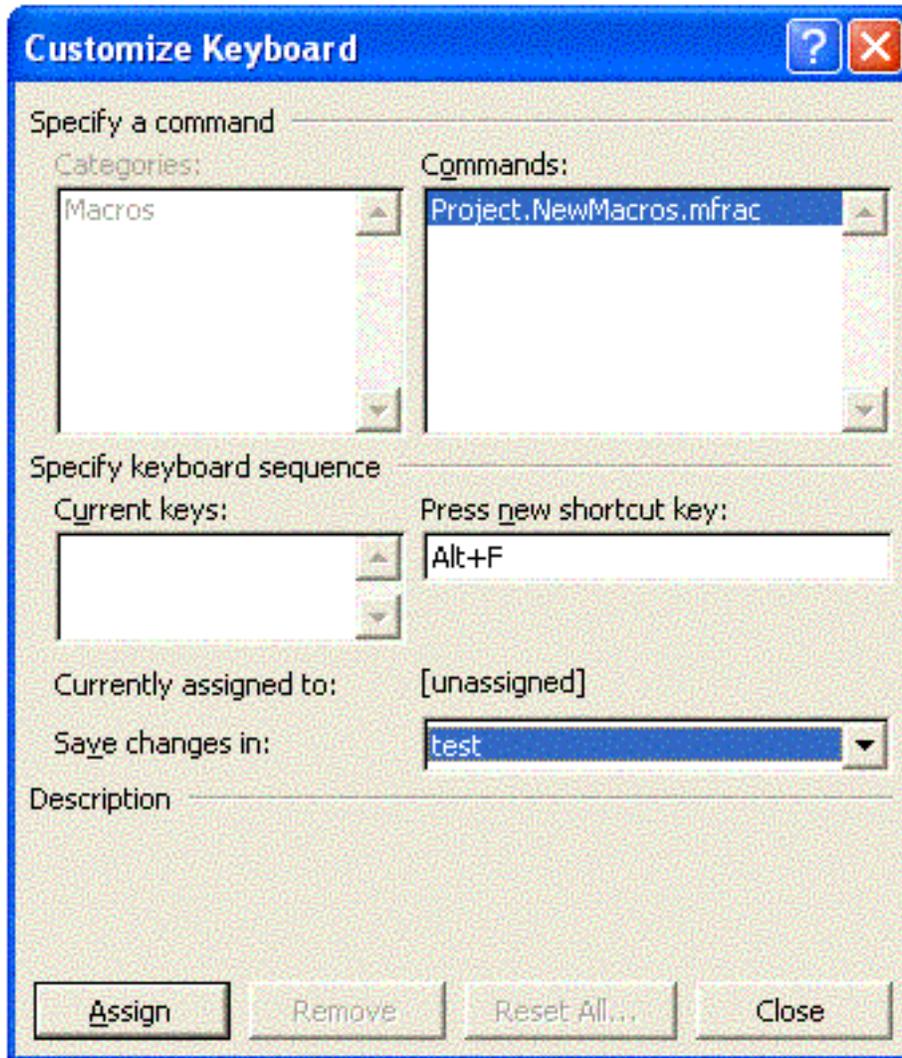


Figure 4

Step 7 : Press “Assign” button and then “Close” button. Macro recording has started. A pair of buttons are displayed on the margin of the document as shown in the figure below. One of the button is “Stop” button, while the other is “Pause” button for macro recording.

Recording code lines

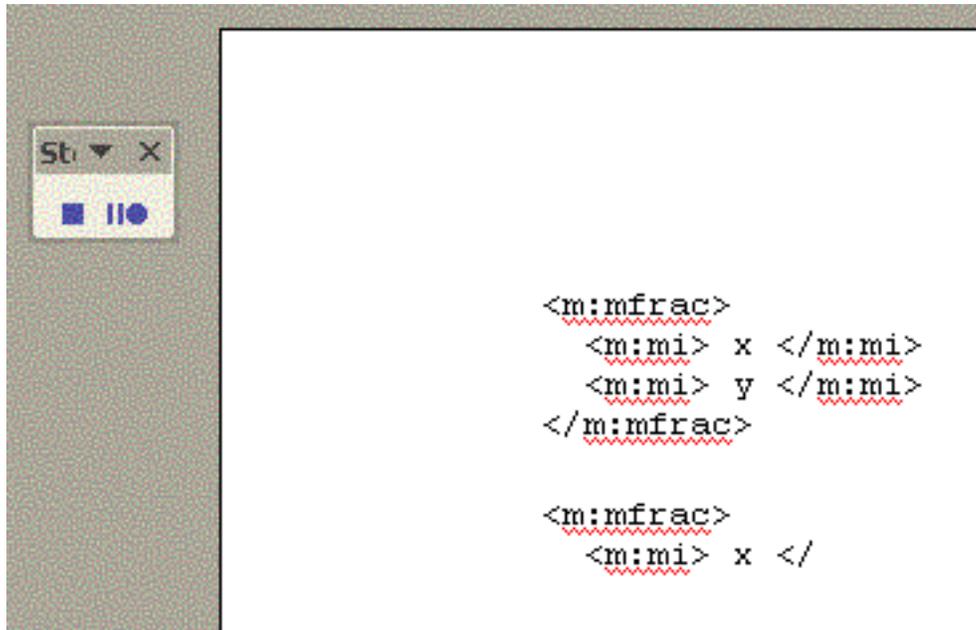


Figure 5

Now, looking at the code lines written in step 3, type exactly the same after one or two lines. Do not bother for key and cursor movements. It is advisable that we maintain the required indenting say between “mfrac” tag and “mi” tag. When the code lines have been reproduced, press “Stop” button to stop the recording.

Step 8 : You are now ready to use the macro. Hold “Alt” key and simultaneously press “F” key. The code lines as recorded are reproduced instantly as shown here :

```

<m:mfrac>
  <m:mi> x </m:mi>
  <m:mi> y </m:mi>
</m:mfrac>

```

Macro recording is a powerful tool to reproduce a set of code lines. However, this facility is available in the normal document mode and not in the plane text mode. As such, codes are required to be first written in a document file, say “test.doc”. Once coding is complete in a major way, we need to save the file in plane text with Unicode character format (UTF -8) for renderer to display the content. It is implied here that a repository of macros corresponding to the basic forms of element encoding be prepared in advance in order to use the same, when required.

In the following section, we shall illustrate the use of macros and few improvisations to encode “Cauchy-Schwarz” inequality.

2 “Cauchy-Schwarz” inequality

Understanding of the underlying presentation framework of MathML is just one aspect of learning process. Equally important aspect is to develop the ability to convert presentation requirement in terms of codes. There is no universal set of algorithms that would help to successfully translate visual presentation requirement into MathML code lines in a mechanical manner - unless aided by specialized program or software. Unaided, this may be achieved by developing ability to visualize the presentation requirements in some logical modular work pieces like :

- Break the presentation requirement in smaller components
- Write building blocks of code with the help of macros
- Copy and paste identical components and modify
- Combine the code lines to make a “whole”

“Cauchy-Schwarz” inequality is shown here :

$$\left(\sum_{k=1}^n a_k b_k \right)^2 \leq \left(\sum_{k=1}^n a_k^2 \right) \left(\sum_{k=1}^n b_k^2 \right)$$

A visual inspection of the formula suggests that this inequality is built of three large blocks of similar composition. The first block is raised to a power of “2”. This can be implemented with “msup” element. Thus, we start with using macro for “msup” element :

```
<m:msup>
  <m:mi> x </m:mi>
  <m:mi> y </m:mi>
</m:msup>
```

The superscript is changed to “2” as required. A code line for element “mo” is written using macro for “mo” element and its content is changed to parenthesis :

```
<m:math display="block">
  <m:mo> ( </m:mo>
  <m:msup>
    <m:mi> x </m:mi>
    <m:mn> 2 </m:mn>
  </m:msup>
</m:math>
```

The base of the superscript is the large block, involving summation sign. The base is built upon summation sign with “over” and “under” limits implemented by “munderover” element. We write “munderover” basic code lines, using macro to replace the “base” of superscript i.e. “x” in the basic format reproduced by macro in the above step.

```

<m:math display="block">
  <m:mo> ( </m:mo>
  <m:msup>
  <m:munderover>
    <m:mo> &int; </m:mo>
    <m:mi>x</m:mi>
    <m:mi>y</m:mi>
  </m:munderover>
  <m:mn> 2 </m:mn>
</m:msup>

</m:math>

```

We replace integer symbol with summation symbol “ \sum ” and change the limits as required. Note that we need to bunch the underscript with “mrow” tags to work as one of the argument of the “msup” element.

```

<m:math display="block">
  <m:mo> ( </m:mo>
  <m:msup>
  <m:munderover>
    <m:mo>&sum;</m:mo>
    <m:mrow>
    <m:mi> k </m:mi>
    <m:mo> = </m:mo>
    <m:mn> 1</m:mn>
    </m:mrow>
    <m:mn>n</m:mn>
  </m:munderover>
  <m:mn> 2 </m:mn>

  </m:msup>

</m:math>

```

Now, we add codes for a_k and b_k , which can be implemented with “msub” element. We produce two sets of “msub” codes with macro and place them just after </m:munderover>. Finally, we also insert a “mo” line to encode for closing parenthesis. Putting these code lines ahead of superscript element line <m:mi> 2 </m:mi> is important as everything in totality is raised to the power “2”. For this reason, everything other than superscript element line is required to be bunched together with a pair of “mrow” tags as shown here :

```

<m:math display="block">
  <m:mo> ( </m:mo>
  <m:msup>

  <m:mrow>

```

```

<m:munderover>
  <m:mo>&sum;</m:mo>
  <m:mrow>
    <m:mi> k </m:mi>
    <m:mo> = </m:mo>
    <m:mn> 1</m:mn>
  </m:mrow>
  <m:mn>n</m:mn>
</m:munderover>
<m:msub>
  <m:mi> a </m:mi>
  <m:mi> k </m:mi>
</m:msub>
<m:msub>
  <m:mi> b </m:mi>
  <m:mi> k </m:mi>
</m:msub>
<m:mo> ) </m:mo>

</m:mrow>

<m:mn> 2 </m:mn>
</m:msup>

</m:math>

```

This completes the coding for the first left hand side block of the “Cauchy-Schwarz” inequality. At this stage, the display in the browser looks like :

$$\left(\sum_{k=1}^n a_k b_k \right)^2$$

Now coding for two other large block is easy. We need to replicate the above code block twice and do the modification as required. Finally three blocks are joined together with the help of an “inequality” operator “≤”. The display, now, looks like :

$$\left(\sum_{k=1}^n a_k b_k \right)^2 \leq \left(\sum_{k=1}^n a_k^2 \right) \left(\sum_{k=1}^n b_k^2 \right)$$

We observe that the parenthesis of the first term has not grown to the expected height to cover the highest term. Moreover, the parentheses are unequal in size. To achieve the equality in size and proper coverage, we enclose all expressions involving parentheses within a pair of “mrow” tags and a “mstyle” pairs of tags to display “true”. The complete code and display is illustrated in the example below :

Example 1: “Cauchy-Schwarz” inequality

```

<m:math display="block">
  <m:mstyle displaystyle='true'>

```

```

    <m:mrow>
    <m:msup>
    <m:mrow>
    <m:mo> ( </m:mo>
<m:munderover>
    <m:mo>&sum;</m:mo>
    <m:mrow>
    <m:mi> k </m:mi>
    <m:mo> = </m:mo>
    <m:mn> 1</m:mn>
    </m:mrow>
    <m:mn>n</m:mn>
</m:munderover>
<m:msub>
<m:mi> a </m:mi>
<m:mi> k </m:mi>
</m:msub>
<m:msub>
<m:mi> b </m:mi>
<m:mi> k </m:mi>
</m:msub>
<m:mo> ) </m:mo>
</m:mrow>
<m:mn> 2 </m:mn>
</m:msup>

<m:mo> &le; </m:mo>

<m:mrow>
<m:mo> ( </m:mo>
<m:munderover>
    <m:mo>&sum;</m:mo>
    <m:mrow>
    <m:mi> k </m:mi>
    <m:mo> = </m:mo>
    <m:mn> 1</m:mn>
    </m:mrow>
    <m:mn>n</m:mn>
</m:munderover>
<m:msup>
<m:mrow>
<m:msub>
<m:mi> a </m:mi>
<m:mi> k </m:mi>
</m:msub>
</m:mrow>
<m:mn> 2 </m:mn>
</m:msup>
<m:mo> ) </m:mo>
</m:mrow>

```

```

    <m:mrow>
    <m:mo> ( </m:mo>
<m:munderover>
    <m:mo>&sum;</m:mo>
    <m:mrow>
    <m:mi> k </m:mi>
    <m:mo> = </m:mo>
    <m:mn> 1</m:mn>
    </m:mrow>
    <m:mn>n</m:mn>
</m:munderover>
<m:msup>
<m:mrow>
<m:msub>
<m:mi> b </m:mi>
<m:mi> k </m:mi>
</m:msub>
</m:mrow>
<m:mn> 2 </m:mn>
</m:msup>
<m:mo> ) </m:mo>
</m:mrow>
</m:mrow>
</m:mstyle>
</m:math>

```

Save the file after editing as “test.xml”. The display looks like :

$$\left(\sum_{k=1}^n a_k b_k \right)^2 \leq \left(\sum_{k=1}^n a_k^2 \right) \left(\sum_{k=1}^n b_k^2 \right)$$