MathML Formatting (with $\TeX$ Rules and $\TeX$ Fonts)

Luca Padovani

Department of Computer Science

University of Bologna

Member of

HELM Project http://helm.cs.unibo.it

MoWGLI Project IST-2001-33562 http://mowgli.cs.unibo.it

W3C Math Working Group http://www.w3.org/Math

Summary

- MathML
- Definition of the problem
- Architecture of the formatting engine
- Area model for MathML
- MathML formatting rules
- Conclusion

MathML Presentation: example

MathML Presentation: example

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

$$\lim_{x \to 0} \frac{\sin x}{x} = 25$$

```
</math>
```

MathML Presentation: example

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mrow>
```

$$\lim_{x \to 0} \frac{\sin x}{x} = 25$$

```
  </mrow>
</math>
```

MathML Presentation: example

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mrow>
    <mrow>









    </mrow>
    <mo> = </mo>
    <mn> 25 </mn>
  </mrow>
</math>
```

$$\lim_{x \to 0} \frac{\sin x}{x} = 25$$

MathML Presentation: example

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mrow>
    <mrow>
      <munder>
        <mo> lim </mo>
        <mrow>



        </mrow>
      </munder>
      <mfrac>
        <mrow>



        </mrow>
        <mi> x </mi>
      </mfrac>
    </mrow>
    <mo> = </mo>
    <mn> 25 </mn>
  </mrow>
</math>
```

$$\lim_{x \longrightarrow 0} \frac{\sin x}{x} = 25$$

MathML Presentation: example

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mrow>
    <mrow>
      <munder>
        <mo> lim </mo>
        <mrow>
          <mi> x </mi>
          <mo> &RightArrow; </mo>
          <mn> 0 </mn>
        </mrow>
      </munder>
      <mfrac>
        <mrow>
          <mi> sin </mi>
          <mo> &ApplyFunction; </mo>
          <mi> x </mi>
        </mrow>
        <mi> x </mi>
      </mfrac>
    </mrow>
    <mo> = </mo>
    <mn> 25 </mn>
  </mrow>
</math>
```

$$\lim_{x \to 0} \frac{\sin x}{x} = 25$$

MathML Presentation: example

```
<math xmlns="http://www.w3.org/1998/Math/MathML">
  <mrow>
    <mrow>
      <munder>
        <mo> lim </mo>
        <mrow>
          <mi> x </mi>
          <mo> &RightArrow; </mo>
          <mn> 0 </mn>
        </mrow>
      </munder>
      <mfrac>
        <mrow>
          <mi> sin </mi>
          <mo> &ApplyFunction; </mo>
          <mi> x </mi>
        </mrow>
        <mi> x </mi>
      </mfrac>
    </mrow>
    <mo> = </mo>
    <mn> 25 </mn>
  </mrow>
</math>
```

$$\lim_{x \to 0} \frac{\sin x}{x} = 25$$

MathML Presentation

- About 30 MathML presentation elements which accept about 50 attributes

MathML Presentation

- About 30 MathML presentation elements which accept about 50 attributes

  Most elements represent templates or patterns for laying out subexpressions. For example, there is an `mfrac` element for fractions, and an `msqrt` element for square roots

MathML Presentation

- About 30 MathML presentation elements which accept about 50 attributes

  Most elements represent templates or patterns for laying out subexpressions. For example, there is an `mfrac` element for fractions, and an `msqrt` element for square roots

  Attributes generally specify additional optional information about the element layout. For example, the `mfrac` element has an attribute called `linethickness`

MathML Presentation

- About 30 MathML presentation elements which accept about 50 attributes

  Most elements represent templates or patterns for laying out subexpressions. For example, there is an `mfrac` element for fractions, and an `msqrt` element for square roots

  Attributes generally specify additional optional information about the element layout. For example, the `mfrac` element has an attribute called `linethickness`

- Using presentation markup, it's possible to specify how an expression will look when displayed

MathML Presentation Summary

- tokens (`mi`, `mo`, `mn`)
- general layout schemata (`mfrac`, `msqrt`)
- scripts and limits (`msub`, `msup`, `munder`, `mover`)
- tables and alignment (`mtable`, `mtr`, `mtd`)
- style and attribute inheritance (`mstyle`)
- "live" expressions (`maction`)

There is a fair amount of semantics even in presentation elements:

- refine formatting, higher quality
- "meaningful" presentation (conversions)

$\TeX$ Formatting Rules: Appendix G

$\TeX$ Formatting Rules: Appendix G

About fractions:

If $C > T$, set $u \leftarrow \sigma_8$ and $v \leftarrow \sigma_{11}$. Otherwise set $u \leftarrow \sigma_9$ or σ_{10}, according as $\theta \neq 0$ or $\theta = 0$, and set $v \leftarrow \sigma_{12}$. . .

$\TeX$ Formatting Rules: Appendix G

About fractions:

If $C > T$, set $u \leftarrow \sigma_8$ and $v \leftarrow \sigma_{11}$. Otherwise set $u \leftarrow \sigma_9$ or σ_{10}, according as $\theta \neq 0$ or $\theta = 0$, and set $v \leftarrow \sigma_{12} \ldots$

About scripts:

If the translation of the nucleus is a character box, possibly followed by a kern, set u and v equal to zero; otherwise set $u \leftarrow h - q$ and $v \leftarrow d + r$, where h and d are the height and depth of the translated nucleus, and where q and r are the values of σ_{18} and σ_{19} in the font corresponding to styles $C\uparrow$ and $C\downarrow \ldots$

MathML Formatting

Two contrasting objectives:

- we want to design a MathML formatting engine which is capable of using TeX fonts (hence TeX formatting rules, see Appendix G)
- we don't want to tie the engine to TeX fonts, as we recognize that a significant part of the formatting process is independent of the environment

Plan

Deeper analysis of the formatting process:

- understand and separate the modules of the architecture that depend on the environment from those that don't
- define a common interface for the loss-less communication of information among the modules
- exploit the semantically rich MathML markup for refining the outcome automatically

Formatting functions

We can express the formatting function for MathML recursively on the structure of the tree

Formatting functions

We can express the formatting function for MathML recursively on the structure of the tree

$$[\![\texttt{<}t\texttt{>}\ c_1 \cdots c_n\ \texttt{</}t\texttt{>}]\!] = \mathbf{f}_t(c_1 \cdots c_n)$$

$$[\![\texttt{<}t\texttt{>}\ X_1 \cdots X_m\ \texttt{</}t\texttt{>}]\!] = \mathbf{f}_t([\![X_1]\!], \ldots, [\![X_m]\!])$$

Formatting functions

We can express the formatting function for MathML recursively on the structure of the tree

$$[\![\texttt{<}t\texttt{>}\ c_1 \cdots c_n\ \texttt{</}t\texttt{>}]\!]_C = \mathbf{f}_t(C', c_1 \cdots c_n)$$

$$[\![\texttt{<}t\texttt{>}\ X_1 \cdots X_m\ \texttt{</}t\texttt{>}]\!]_C = \mathbf{f}_t(C', [\![X_1]\!]_{C_1}, \ldots, [\![X_m]\!]_{C_m})$$

Formatting functions

We can express the formatting function for MathML recursively on the structure of the tree

$$
\begin{aligned}
[\![\texttt{<}t\texttt{>}\; c_1 \cdots c_n \;\texttt{</}t\texttt{>}]\!]_C &= \mathbf{f}_t(C', c_1 \cdots c_n) \\
[\![\texttt{<}t\texttt{>}\; X_1 \cdots X_m \;\texttt{</}t\texttt{>}]\!]_C &= \mathbf{f}_t(C', [\![X_1]\!]_{C_1}, \ldots, [\![X_m]\!]_{C_m})
\end{aligned}
$$

We seek for a decomposition of the $\mathbf{f}_t$ functions such that

- $\mathbf{f}_t \approx \mathbf{g}_t \circ \mathbf{h}_t$
- the $\mathbf{g}_t$ are independent of the environment
- neither of $\{ \mathbf{g}_t \}$ and $\{ \mathbf{h}_t \}$ is trivial

Criteria for decomposition

We have to format an element of type t along with its components:

- any operation that depends only on the "size" (the bounding box) of the components is part of the $\mathbf{g}_t$

Criteria for decomposition

We have to format an element of type t along with its components:

- any operation that depends only on the "size" (the bounding box) of the components is part of the $\mathbf{g}_t$

  Example: context update, table layout, line breaking, alignment...

Criteria for decomposition

We have to format an element of type t along with its components:

- any operation that depends only on the “size” (the bounding box) of the components is part of the $\mathbf{g}_t$

  Example: context update, table layout, line breaking, alignment...

- any operation thay may depend on the “shape” (the content) of the components is part of the $\mathbf{h}_t$

Criteria for decomposition

We have to format an element of type t along with its components:

- any operation that depends only on the "size" (the bounding box) of the components is part of the $\mathbf{g}_t$

  Example: context update, table layout, line breaking, alignment...

- any operation thay may depend on the "shape" (the content) of the components is part of the $\mathbf{h}_t$

  Example: Unicode strings, scripts, accents, fractions...

Areas

Formatting functions return areas that describe glyphs and their geometrical relationship

$$[\![\cdot]\!] : \text{MathML Element} \times \text{Formatting Context} \longrightarrow \text{Area}$$

Areas

Formatting functions return areas that describe glyphs and their geometrical relationship

$$[\![\cdot]\!] : \text{MathML Element} \times \text{Formatting Context} \longrightarrow \text{Area}$$

Areas are passed back and forth between the formatting engine and the graphic device for mathematics. They are complex objects that implement a loss-less communication mechanism

Areas

Formatting functions return areas that describe glyphs and their geometrical relationship

$$[\![\cdot]\!] : \text{MathML Element} \times \text{Formatting Context} \longrightarrow \text{Area}$$

Areas are passed back and forth between the formatting engine and the graphic device for mathematics. They are complex objects that implement a loss-less communication mechanism

Areas are opaque to the formatting engine: the formatting engine can ask areas for a (very limited) set of properties, but doesn't know anything about their type

Areas

Formatting functions return areas that describe glyphs and their geometrical relationship

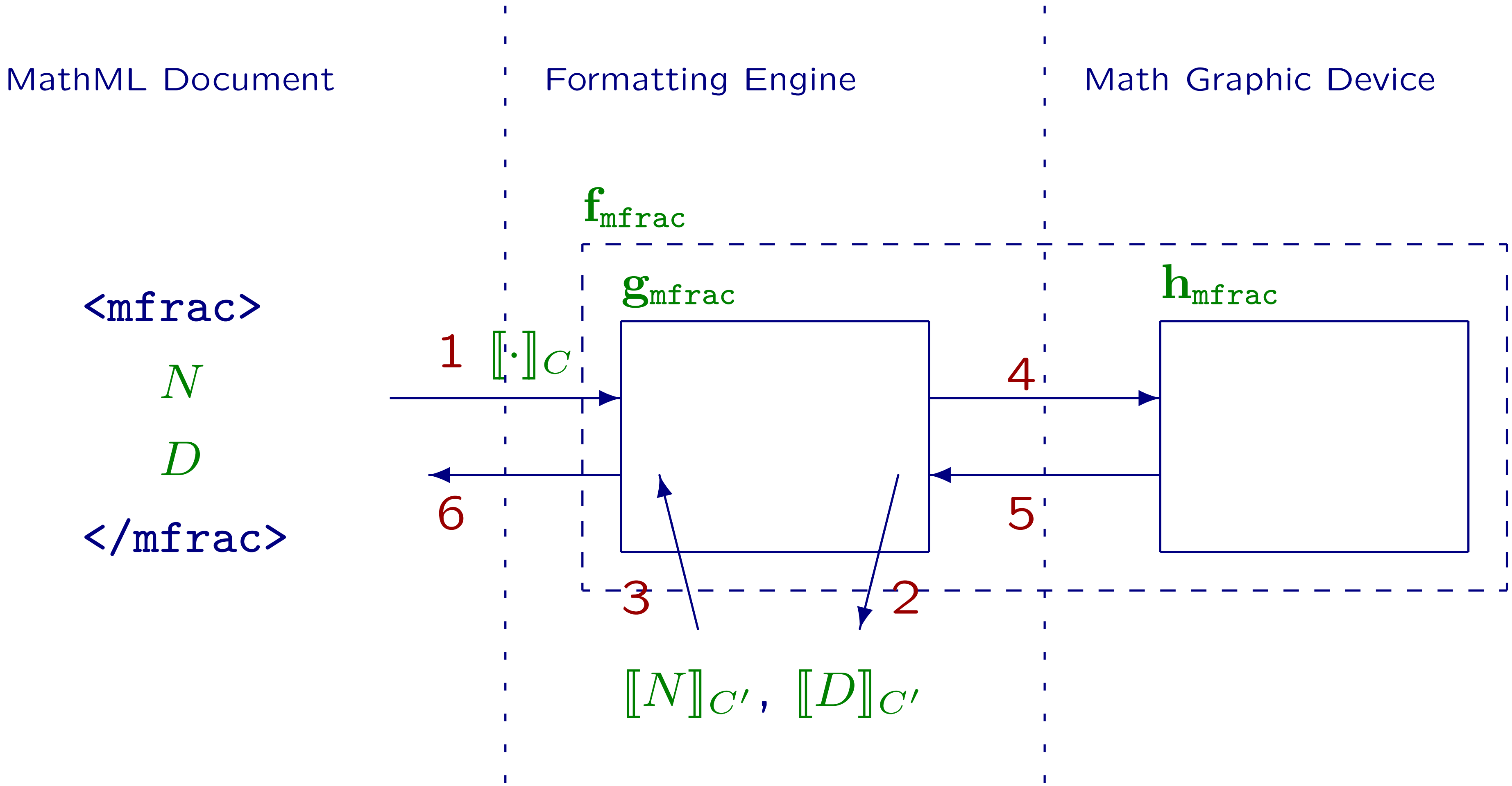
$$[\![\cdot]\!] : \text{MathML Element} \times \text{Formatting Context} \longrightarrow \text{Area}$$

Areas are passed back and forth between the formatting engine and the graphic device for mathematics. They are complex objects that implement a loss-less communication mechanism

Areas are opaque to the formatting engine: the formatting engine can ask areas for a (very limited) set of properties, but doesn't know anything about their type

Areas are transparent to the graphic device for math: the device can do different things depending on the type and the content of areas

Example



MathML Document
Formatting Engine
Math Graphic Device
$\mathbf{f}_{\texttt{mfrac}}$
$\mathbf{g}_{\texttt{mfrac}}$
$\mathbf{h}_{\texttt{mfrac}}$
<mfrac>
N
D
</mfrac>
1 $[\![\cdot]\!]_C$
4
5
6
3
2
$[\![N]\!]_{C'}$, $[\![D]\!]_{C'}$

Areas

Requirements of the area model:

- language that we can use to express the layout of a mathematical formula in a concise and unambiguous way
- language that can be extended such that new types of entities can be introduced by an implementation (e.g. selection, back-pointers,...)

Areas

Requirements of the area model:

- language that we can use to express the layout of a mathematical formula in a concise and unambiguous way
- language that can be extended such that new types of entities can be introduced by an implementation (e.g. selection, back-pointers,. . .)

An area describes a rectangular portion of the output medium as a tree structure. Each node in the tree (an area node) is identified by:

- a type
- a possibly empty set of properties
- a possibly empty, ordered list of child areas

Areas as TEX boxes

Area	TEX	Description
$G[\cdot]$		Glyph
K_n	`\kern`	Kern with value n. The kern is horizontal or vertical depending on the container it is in
F	`\hfill \vfill`	Filler area
R_n	`\hrule \vrule`	Filler rule of thickness n
$S_n[\alpha]$	`\raisebox` `\lowerbox`	Shift α's baseline by n
$H[\alpha_1, \ldots, \alpha_n]$	`\hbox`	Horizontal group of areas α_1,. . . ,α_n
$V_k[\alpha_1, \ldots, \alpha_n]$	`\vbox`	Vertical group of areas α_1,. . . ,α_n, where α_k is the reference area that determines the baseline

$[\![$`<mfrac>` N D `</mfrac>`$]\!]_C$

Formatting context update:

$$
\begin{aligned}
C' &= C\,[\mathit{element} \leftarrow \mathit{this}] \\
C'' &= C'[\text{if } C'.\mathit{displayStyle} = \text{true then} \\
&\qquad\qquad \mathit{displayStyle} \leftarrow \text{false} \\
&\qquad \text{else} \\
&\qquad\qquad \mathit{scriptLevel} \leftarrow C'.\mathit{scriptLevel} + 1; \\
&\qquad\qquad \mathit{aSize} \leftarrow C'.\mathit{aSize} \times C'.\mathit{sizeMult}; \\
&\qquad\qquad \mathit{size} \leftarrow \max\{C'.\mathit{minSize}, C'.\mathit{aSize} \times C'.\mathit{sizeMult}\}]
\end{aligned}
$$

$[\![$`<mfrac>` N D `</mfrac>`$]\!]_C$

Formatting context update:

$$
\begin{aligned}
C' &= C\,[element \leftarrow this] \\
C'' &= C'[\text{if } C'.displayStyle = \text{true then} \\
&\qquad\qquad displayStyle \leftarrow \text{false} \\
&\qquad \text{else} \\
&\qquad\qquad scriptLevel \leftarrow C'.scriptLevel + 1; \\
&\qquad\qquad aSize \leftarrow C'.aSize \times C'.sizeMult; \\
&\qquad\qquad size \leftarrow \max\{C'.minSize, C'.aSize \times C'.sizeMult\}]
\end{aligned}
$$

Fraction formatting:

$$
\begin{aligned}
& [\![\texttt{<mfrac>}\ N\ D\ \texttt{</mfrac>}]\!]_C \\
= \quad & \mathbf{h}_{\texttt{mfrac}}(C', H[F, [\![N]\!]_{C''}, F], H[F, [\![D]\!]_{C''}, F]) \qquad \text{Rules 15-15d} \\
= \quad & S_a[V_3[H[F, [\![D]\!]_{C''}, F], K_d, R_h, K_n, H[F, [\![N]\!]_{C''}, F]]]
\end{aligned}
$$

$[\![$ <msubsup> A B C </msubsup> $]\!]_C$

Formatting context update:

$$
\begin{aligned}
C' &= C\,[element \leftarrow this] \\
C'' &= C'[displayStyle \leftarrow \textsf{false}; \\
&\qquad scriptLevel \leftarrow C'.scriptLevel + 1; \\
&\qquad aSize \leftarrow C'.aSize \times C'.sizeMult\,; \\
&\qquad size \leftarrow \max\{C'.minSize, C'.aSize \times C'.sizeMult\,\}]
\end{aligned}
$$

$[\![$<msubsup> A B C </msubsup>$]\!]_C$

Formatting context update:

$$
\begin{aligned}
C' &= C\,[element \leftarrow this] \\
C'' &= C'[displayStyle \leftarrow \mathsf{false}; \\
&\qquad scriptLevel \leftarrow C'.scriptLevel + 1; \\
&\qquad aSize \leftarrow C'.aSize \times C'.sizeMult; \\
&\qquad size \leftarrow \max\{C'.minSize, C'.aSize \times C'.sizeMult\}]
\end{aligned}
$$

Script formatting:

$$
\begin{aligned}
& [\![\texttt{<msubsup>}\ A\ B\ C\ \texttt{</msubsup>}]\!]_C \\
= \quad & \mathbf{h}_{\texttt{script}}(C', [\![A]\!]_{C'}, [\![B]\!]_{C''}, [\![C]\!]_{C''}) \qquad \text{Rules 18-18f} \\
= \quad & H[[\![A]\!]_{C'}, S_{-b}[V_1[H[K_{-d_1}, [\![B]\!]_{C''}], K_s, H[K_{d_2}, [\![C]\!]_{C''}]]]]
\end{aligned}
$$

More on scripts

Mere knowledge of the base element's bounding box isn't always enough for placing scripts correctly:

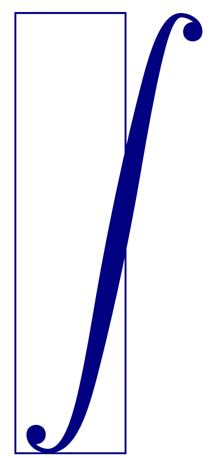
- weird font metrics the formatting engine cannot count on

$$\int$$

More on scripts

Mere knowledge of the base element's bounding box isn't always enough for placing scripts correctly:

- weird font metrics the formatting engine cannot count on



More on scripts

Mere knowledge of the base element's bounding box isn't always enough for placing scripts correctly:

- weird font metrics the formatting engine cannot count on
- structure concealing actual shape

More on scripts

Mere knowledge of the base element's bounding box isn't always enough for placing scripts correctly:

- weird font metrics the formatting engine cannot count on
- structure concealing actual shape

$$\left|\frac{1+\dfrac{1}{x}}{2+\dfrac{1}{x}}\right|$$

More on scripts

Mere knowledge of the base element's bounding box isn't always enough for placing scripts correctly:

- weird font metrics the formatting engine cannot count on
- structure concealing actual shape

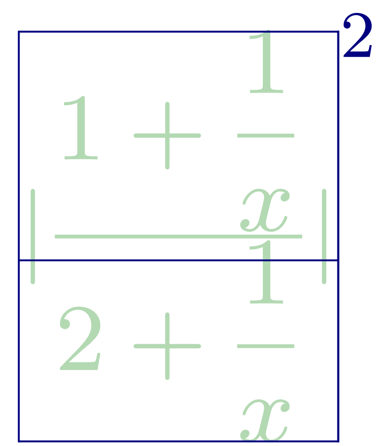
$$\left|\frac{1+\dfrac{1}{x}}{2+\dfrac{1}{x}}\right| \qquad \boxed{\left|\frac{1+\dfrac{1}{x}}{2+\dfrac{1}{x}}\right|}^{2}$$

More on scripts

Mere knowledge of the base element's bounding box isn't always enough for placing scripts correctly:

- weird font metrics the formatting engine cannot count on
- structure concealing actual shape

$$|\frac{1+\frac{1}{x}}{2+\frac{1}{x}}|$$



It is fundamental to communicate information on the type and content of areas when these must be combined together

Tricky $\TeX$ (I)

$\TeX$ is able to work things out with little effort. . .

$$|\frac{1+\frac{1}{x}}{2+\frac{1}{x}}|^2$$

Tricky TEX (I)

TEX is able to work things out with little effort. . .

$$\left|\frac{\displaystyle 1+\frac{1}{x}}{\displaystyle 2+\frac{1}{x}}\right|^2$$

. . . because there is little concern about structure:

```
|\frac{\displaystyle1+\frac{1}{x}}
      {\displaystyle2+\frac{1}{x}}|^2
```

Tricky $\TeX$ (II)

`\sin x`	$\sin x$	`\sin(x)`	$\sin(x)$

Tricky TEX (II)

`\sin x` $\sin x$ `\sin(x)` $\sin(x)$

TEX solves this problem by defining the `sin` macro as

```
\def\sin{\mathop{\rm sin}}
```

Tricky $\TeX$ (II)

`\sin x` $\sin x$ `\sin(x)` $\sin(x)$

$\TeX$ solves this problem by defining the `sin` macro as

```
\def\sin{\mathop{\rm sin}}
```

and then applying the rules

$$\begin{aligned} \mathrm{Op} + \mathrm{Ord} &\Rightarrow \text{thin space} \\ \mathrm{Op} + \mathrm{Open} &\Rightarrow \text{no space} \end{aligned}$$

Tricky $\TeX$ (II)

`\sin x` $\sin x$ `\sin(x)` $\sin(x)$

$\TeX$ solves this problem by defining the `sin` macro as

```
\def\sin{\mathop{\rm sin}}
```

and then applying the rules

$$\begin{aligned} \mathrm{Op} + \mathrm{Ord} &\quad\Rightarrow\quad \text{thin space} \\ \mathrm{Op} + \mathrm{Open} &\quad\Rightarrow\quad \text{no space} \end{aligned}$$

A MathML formatter can address the problem more generally:

```
<mrow>
  <mi> sin </mi>
  <mo> &#x2061; </mo>
  <mi> x </mi>
</mrow>
```

```
<mrow>
  <mi> sin </mi>
  <mo> &#x2061; </mo>
  <mrow>
    <mo> ( </mo>
    <mi> x </mi>
    <mo> ) </mo>
  </mrow>
</mrow>
```

Implementation(s)

Two implementations:

- PocketMath

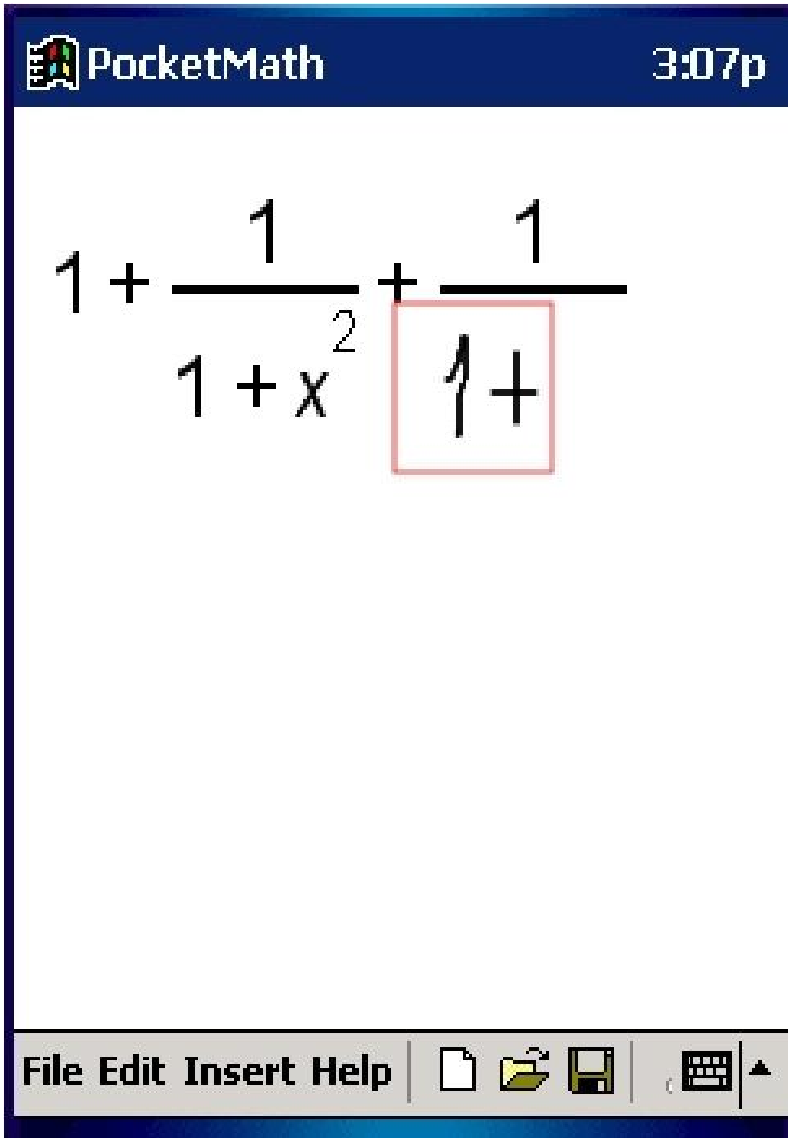
  (Maple)

- GtkMathView

  (GPL, `http://helm.cs.unibo.it/mml-widget/`)

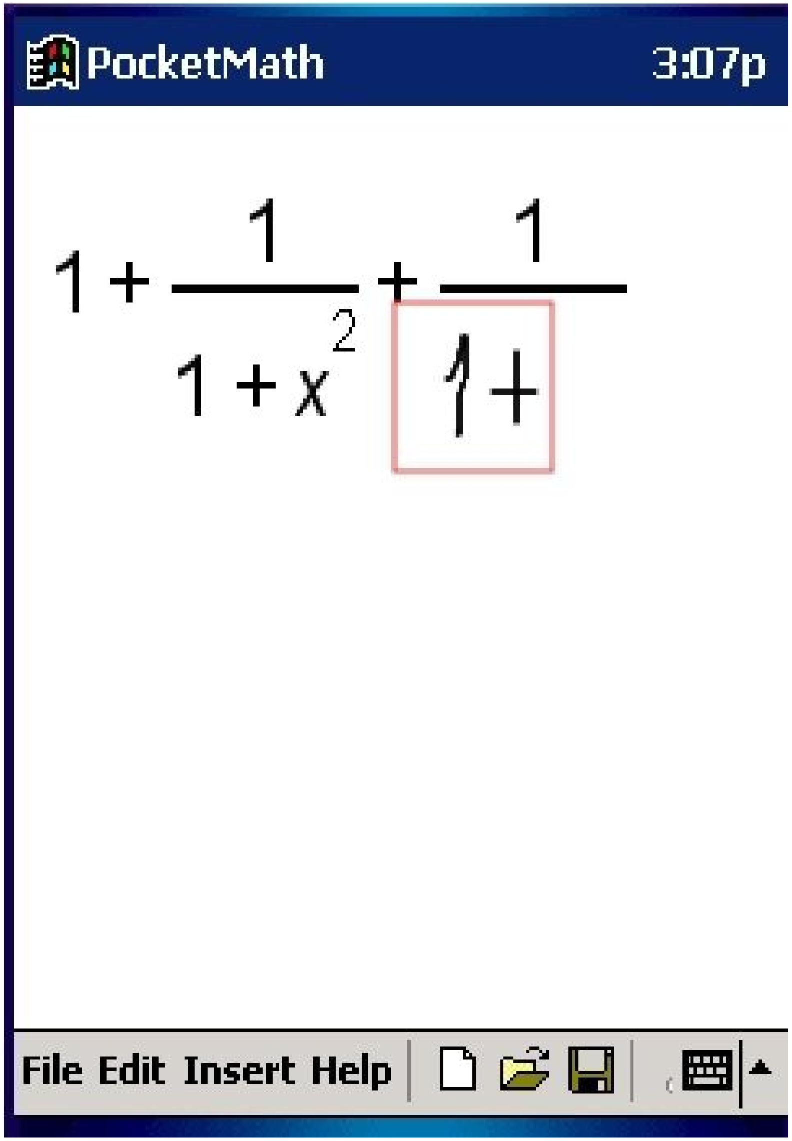
MathML in hand-held devices

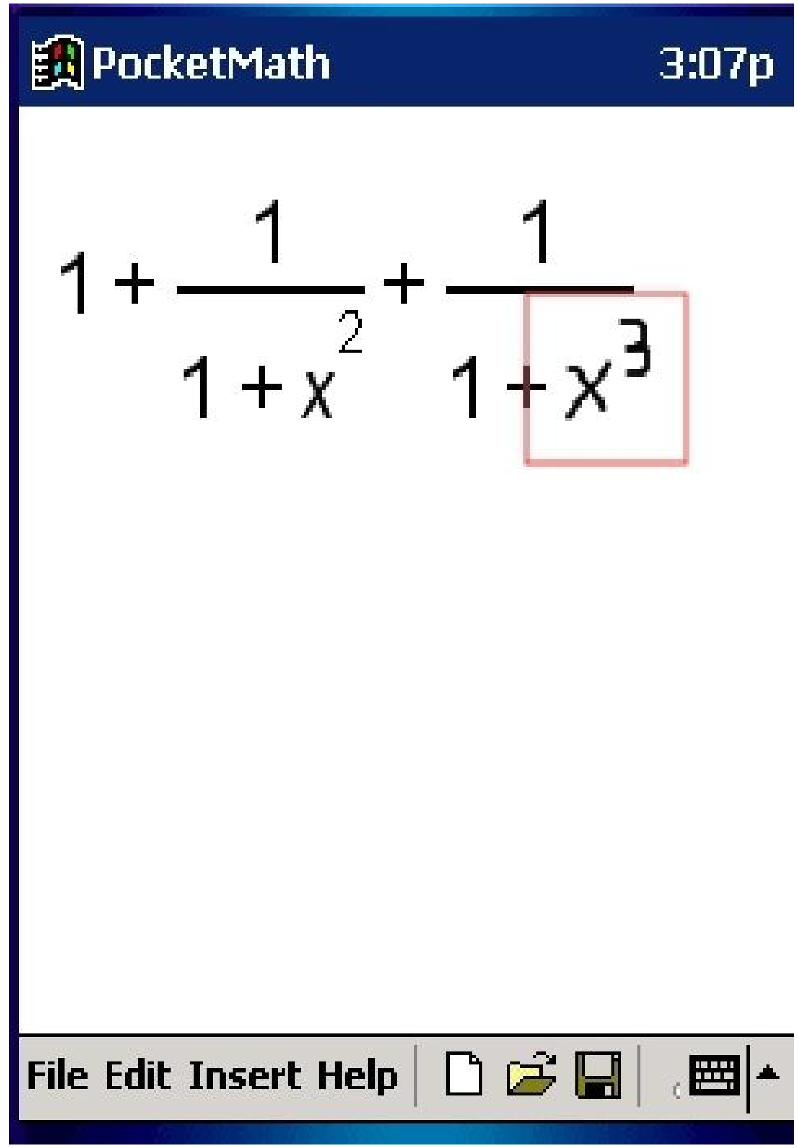
(with Stephen M. Watt, Ontario Research Centre for Computer Algebra)



MathML in hand-held devices

(with Stephen M. Watt, Ontario Research Centre for Computer Algebra)





Wiley encyclopedias and textbooks

(with John Pedersen, John Wiley & Sons, Inc.)

- Burger's Medicinal Chemistry and Drug Delivery (Abraham)
- Encyclopedia of Catalysis (Horvath)
- Encyclopedia of Smart Materials (Schwartz)
- Encyclopedia of Software Engineering (Marciniak)
- Encyclopedia of Polymer Science and Technology
- Handbook of Chemicals and Gases for the Semiconductor Industry (Misra)
- Occupational Toxicants and MAK Values (Deutsche Forschungsgemeinschaft)
- Stevens' Handbook of Experimental Psychology (Pashler)
- Textbook of Biochemistry (Devlin)
- Ullmann's Encyclopedia of Industrial Chemistry (German branch of Wiley)

Also

- a number of Higher Ed/College textbooks being processed

Math in TEX + TEX fonts

- TEX is a single-minded system that does a very good job in a fixed environment: quality printing on paper with TEX's fonts
- TEXfonts have built-in "intelligence"
- any other system that uses TEXfonts will have to understand that "intelligence" and handle it correctly (see Mozilla)
- because TEX has a "standard" implementation, tweaks in TEX markup are harmless

MathML

- MathML can be formatted using TeX formatting rules and also TeX fonts
- it is possible to design the formatter so that it can be adapted very easily for different environments
- because of the semantically rich markup, a smart MathML formatter can handle automatically cases that need author's assistance in TeX (tweaks, line-breaking)