A widget for MathML interaction

Luca Padovani
Department of Computer Science
University of Bologna

MoWGLI Project IST-2001-33562

http://mowgli.cs.unibo.it

Summary

- What is GtkMathView?
- Why does GtkMathView exist?
- Rendering
- Interaction
- Authoring
- Internals
- Comparison

MathML Presentation: example

MathML Presentation: example

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

$$\lim_{x \to 0} \frac{\sin x}{x} = 28$$

```
</math>
```

MathML Presentation: example

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

$$\lim_{x \to 0} \frac{\sin x}{x} = 28$$

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

MathML Presentation: example

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

$$\lim_{x \to 0} \frac{\sin x}{x} = 28$$

```
    </mrow>
    <mo> = </mo>
    <mn> 28 </mn>
  </mrow>
</math>
```

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> 28 </mn>
  </mrow>
</math>
```

$$\lim_{x \to 0} \frac{\sin x}{x} = 28$$

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> 28 </mn>
  </mrow>
</math>
```

$$\lim_{x \to 0} \frac{\sin x}{x} = 28$$

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> 28 </mn>
  </mrow>
</math>
```

$$\lim_{x \to 0} \frac{\sin x}{x} = 28$$

MathML Presentation Overview

- 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)

Purpose

Development of an interface for rendering of/interaction with MathML markup.

Purpose

Development of an interface for rendering of/interaction with MathML markup.

Why another MathML rendering engine?

- Reliability: current MathML rendering engines are still unsatisfactory (to us. . .)
- Efficiency: formalized mathematical documents may be too heavy for current MathML rendering engines
- Flexibility: creation of self-contained applications for browsing, annotating, interacting with mathematical documents

Architecture

GtkMathView is logically divided into

- a platform-independent rendering engine for MathML (C++)

Architecture

GtkMathView is logically divided into

- a platform-independent rendering engine for MathML (C++)
- a set of platform abstractions (fonts, font management, drawing primitives)

Architecture

GtkMathView is logically divided into

- a platform-independent rendering engine for MathML (C++)
- a set of platform abstractions (fonts, font management, drawing primitives)
- a set of interfaces:
  - GTK+ (C)
  - LablGTK (Ocaml)
  - PostScript

Formatting and Rendering

Formatting goes through 3 stages:

Formatting and Rendering

Formatting goes through 3 stages:

1. construction of the formatting tree from the source MathML tree

Formatting and Rendering

Formatting goes through 3 stages:

1. construction of the formatting tree from the source MathML tree
2. refinement and evaluation of MathML attributes

Formatting and Rendering

Formatting goes through 3 stages:

1. construction of the formatting tree from the source MathML tree
2. refinement and evaluation of MathML attributes
3. proper formatting

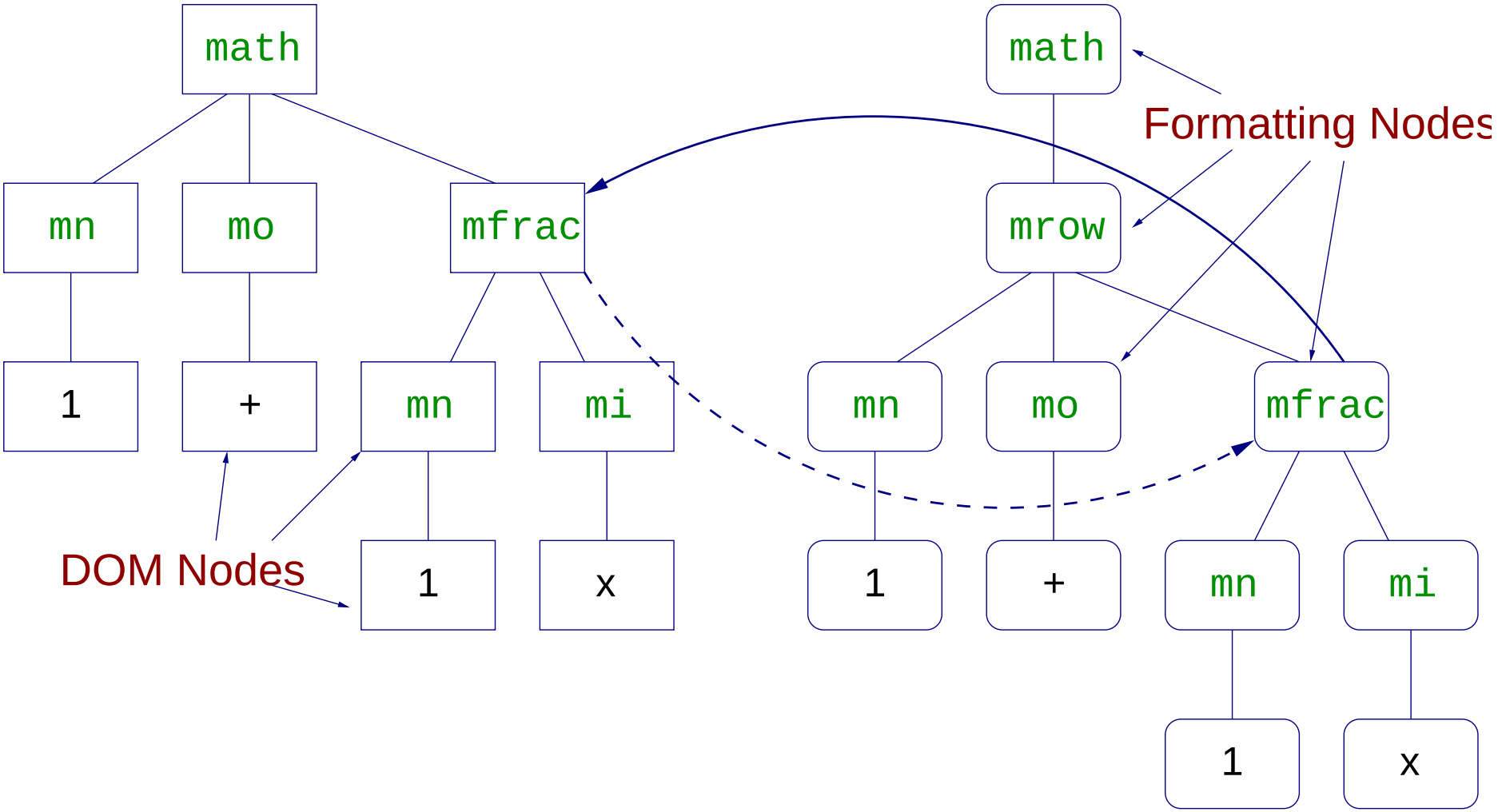
Formatting and Rendering

Formatting goes through 3 stages:

1. construction of the formatting tree from the source MathML tree
2. refinement and evaluation of MathML attributes
3. proper formatting

Construction, refinement and a large part of formatting (tables,. . .) are part of the context independent layer.

Formatting tree



Source MathML tree

Formatting tree

Reactive Rendering

GtkMathView listens to modifications of the source MathML document and updates the view accordingly (DOM events).

In a sense, GtkMathView supports the most general form of editing.

The MathML document can be shared among GtkMathView and the host application, several different views can be provided for the same document. ▶

Reactive Rendering

GtkMathView listens to modifications of the source MathML document and updates the view accordingly (DOM events).

In a sense, GtkMathView supports the most general form of editing.

The MathML document can be shared among GtkMathView and the host application, several different views can be provided for the same document. ▶

Issue: updating the view should be efficient. GtkMathView tries to minimize the number of MathML elements to re-format. This is not a trivial task in general as some MathML elements have non-local formatting semantics (think of the cells in a table)

Interactivity Support

Any event arriving from the graphical interface is delegated by GtkMathView to the host application.

Events are annotated with information that is specific to the rendered document (e.g. the DOM element on which the mouse pointer is).

Interactivity Support

Any event arriving from the graphical interface is delegated by GtkMathView to the host application.

Events are annotated with information that is specific to the rendered document (e.g. the DOM element on which the mouse pointer is).

GtkMathView separates the policy from the mechanism: it lets the application decide what to do about a particular event.

This way the amount of internal information GtkMathView has to handle is kept to a minimum, increased flexibility.

Actions

A click on the view fires the "click" signal:

$$click(x, s)$$

where x is the deepest MathML element (in the document tree) under the mouse pointer, s is the status of control keys on the keyboard.

Actions

A click on the view fires the "click" signal:

$$click(x, s)$$

where x is the deepest MathML element (in the document tree) under the mouse pointer, s is the status of control keys on the keyboard.

Even a simple event like this may have various interpretations:

- if x has an `href` attribute, should I follow the hyperlink?
- if x is an `maction` element, or if x has an `maction` ancestor, should I activate `maction`?
- if x has several `maction` ancestors, how do they interact with each other?

Actions

A click on the view fires the "click" signal:

$$click(x, s)$$

where x is the deepest MathML element (in the document tree) under the mouse pointer, s is the status of control keys on the keyboard.

Even a simple event like this may have various interpretations:

- if x has an `href` attribute, should I follow the hyperlink?
- if x is an `maction` element, or if x has an `maction` ancestor, should I activate `maction`?
- if x has several `maction` ancestors, how do they interact with each other?

Behavior is not explicit in the markup and may differ depending on the application's context. ▶ ▶

Selections

GtkMathView has 4 signals for selections, occurring in sequences matching the following regular expression:

$$begin(x, s)\ over(x, s)*\ (end(x, s) \mid abort())$$

In addition, it provides two methods for changing the selection status of a MathML element: $select(x), unselect(x)$.

Selections

GtkMathView has 4 signals for selections, occurring in sequences matching the following regular expression:

$$begin(x, s)\ over(x, s)*\ (end(x, s) \mid abort())$$

In addition, it provides two methods for changing the selection status of a MathML element: $select(x), unselect(x)$.

Simple structural selection is achieved by: ▶

$$\begin{array}{rcl} begin(x, s) & \Rightarrow & selected := x \\ & & select(selected) \\ over(x, s) & \Rightarrow & unselect(selected) \\ & & selected := \mathsf{min}(selected, x) \\ & & select(selected) \end{array}$$

Semantic Selection

Assuming MathML presentation has been generated from content/semantic markup, selection may be constrained by backpointers. ▶

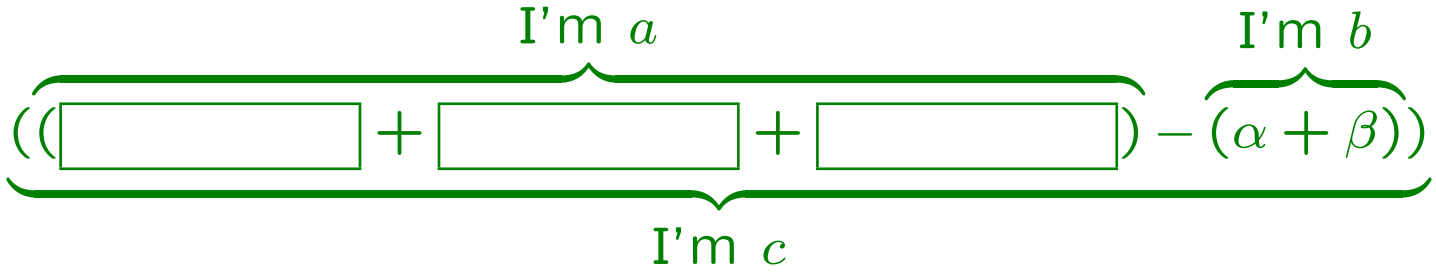
```
<apply id="i1">
  <power/>
  <apply id="i2">
    <plus/>
    <ci>x</ci>
    <cn>1</cn>
  </apply>
  <cn>2</cn>
</apply>
```

```
<mrow xref="i1">
  <msup>
    <mrow xref="i2">
      <mo>(</mo>
      <mrow>
        <mi>x</mi>
        <mo>+</mo>
        <mn>1</mn>
      </mrow>
      <mo>)</mo>
    </mrow>
    <mn>2</mn>
  </mrow>
</mrow>
```

$(x+1)^2$

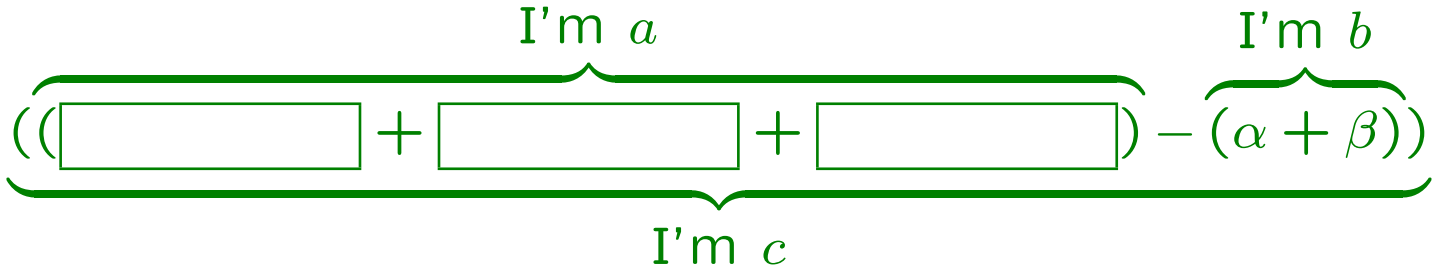
Broken structure

Sometimes structure cannot be preserved in order to achieve the desired rendering. Example:

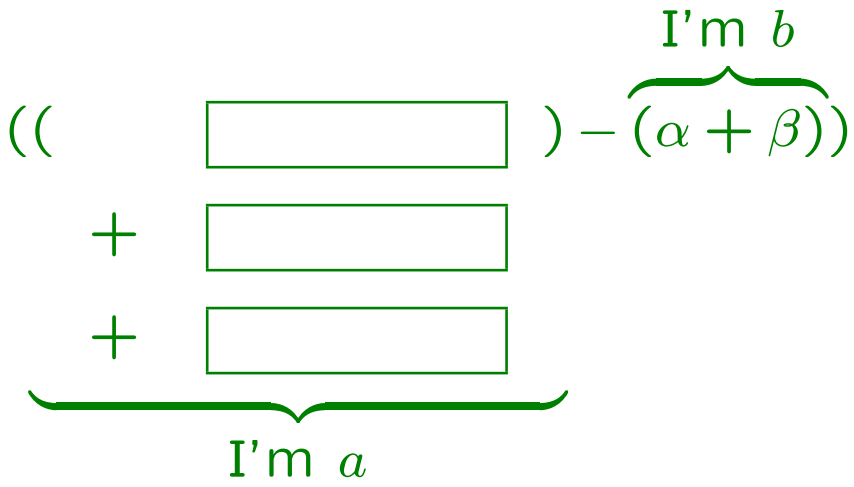


Broken structure

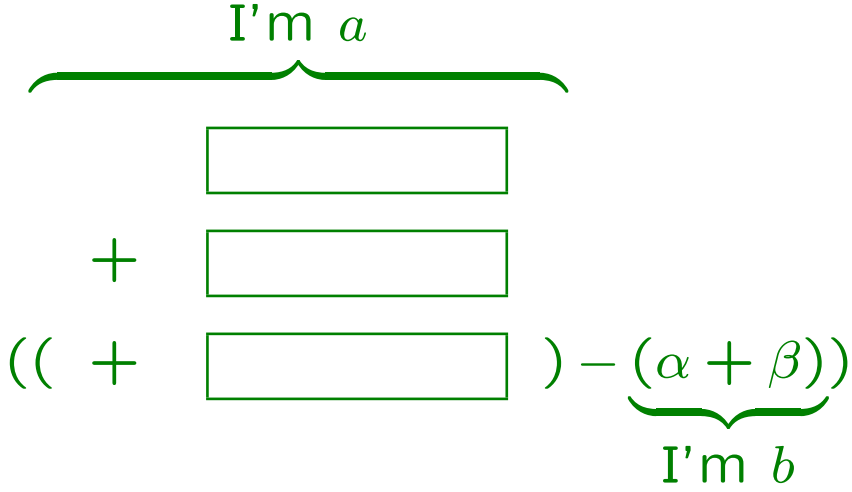
Sometimes structure cannot be preserved in order to achieve the desired rendering. Example:



If no line-breaking is supported and we want to preserve the structure, in MathML we can have



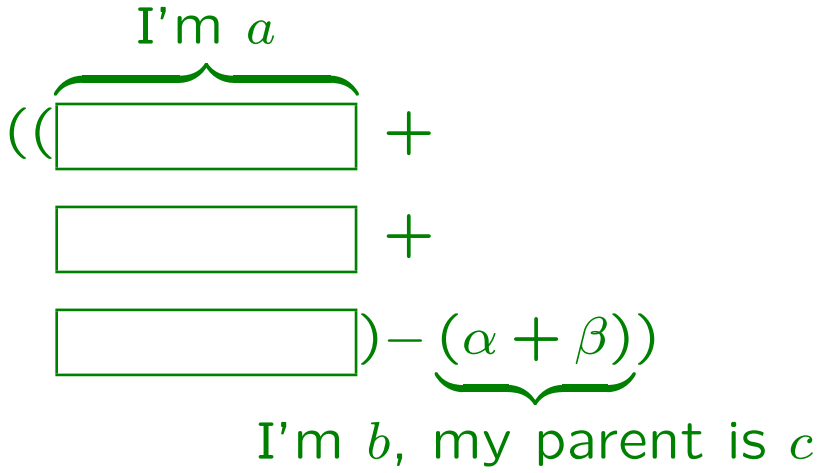
or



. . . none of which is really what we want.

Broken structure

We can't preserve the structure in the tags, but we can give hints to the application in the attributes:



Other selections

Other forms of selections we can think of:

- linear selection: select a set of subsequent items (in document order) regardless of the structure. Supported, not implemented.

Other selections

Other forms of selections we can think of:

- linear selection: select a set of subsequent items (in document order) regardless of the structure. Supported, not implemented.
- multiple selections: need to act on several items simultaneously. Supported. ▶

Other selections

Other forms of selections we can think of:

- linear selection: select a set of subsequent items (in document order) regardless of the structure. Supported, not implemented.
- multiple selections: need to act on several items simultaneously. Supported. ▶
- selections with holes: expressing patterns in a graphical way. Holes are metavariables. Supported, yet currently unused.

Other selections

Other forms of selections we can think of:

- linear selection: select a set of subsequent items (in document order) regardless of the structure. Supported, not implemented.
- multiple selections: need to act on several items simultaneously. Supported. ▶
- selections with holes: expressing patterns in a graphical way. Holes are metavariables. Supported, yet currently unused.
- structured selections: subparts are "more selected" than the whole. Currently unsupported.

Editing

GtkMathView's reactive rendering allows applications to support their own editing toolbox.

Editing

GtkMathView's reactive rendering allows applications to support their own editing toolbox.

The approach we're currently investigating tries to unify WYSIWYG editors and TeX to MathML converters:

- Editing MathML using menus and palettes is not effective (tedious, the user can't improve with time) so...
- ...we use a concrete syntax that is rendered on-the-fly (with reasonable feedback) and resembles how the mathematical formula is spelled (TEX) ▶

Issues: performances, cursor, re-editing, assistance (syntax highlighting, matching parentheses,...)

Performances: small documents

Several small-size MathML fragments from the official MathML testsuite (tests 10, 100, and 1000 respectively). Times are in seconds.

	A	B	C
Galeon	~ 2	~ 3	~ 12
Mozilla	~ 2	~ 3	~ 12
Amaya	~ 1	~ 1	~ 6
GtkMathView	0.4	0.8	6.4

The tests have been performed on a P4 1.7GHz, 512Mb RAM

Performances: medium documents

Some "complex" MathML fragments from the official MathML testsuite (tests `complex1`, `complex2`, `complex3`, and `complex4` respectively). Times are in seconds.

	A	B	C	D
Galeon	~ 2	~ 2	~ 15	~ 3
Mozilla	~ 2	~ 3	~ 14	~ 3
Amaya	~ 2	~ 1	~ 6	~ 1
GtkMathView	1	0.7	4.8	0.5

Performances: large documents

One large document generated from the COQ library.

	collapsed ($\sim$ 70Kb)	exploded ($\sim$ 600Kb)
Galeon	360	*
Mozilla	480	*
Amaya	$\sim$ 1	75
GtkMathView	0.7	3.3

*Galeon and Mozilla freeze after 30 minutes.

Galeon, Mozilla, and Amaya do not render the document correctly.

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

Open issues

Fine-grained integration and embedding

- current modularization of applications is coarse
- GtkMathView doesn't know about other markup languages, it renders MathML markup in isolation

Open issues

Fine-grained integration and embedding

- current modularization of applications is coarse
- GtkMathView doesn't know about other markup languages, it renders MathML markup in isolation

Automatic line-breaking

- one of the concrete advantages of having markup for math, rather than in an image
- MathML simplifies the task

http://helm.cs.unibo.it/mml-widget/