Presenting Mathematical Content with Flexible Elisions

8th OpenMath Meeting

Michael Kohlhase, *Christoph Lange*, Florian Rabe

{m.kohlhase,ch.lange,f.rabe}@jacobs-university.de

Jacobs University, Bremen, Germany
(formerly International University Bremen)

KWARC – Knowledge Adaptation and Reasoning for Content

June 25, 2007

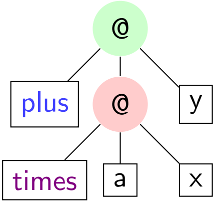
Abstract

- Mathematics has developed a complicated two-dimensional format.
- Mathematical notation influences mathematical thinking.
- Mathematicians frequently elide brackets or symbols to concentrate on essential facts.
- Experienced mathematicians can deduce elided material from the context.
- Content markup needs a *presentation process* (content objects → two-dimensional form)
- We propose an presentation infrastructure for an expressive content dictionary (CD) format that allows for flexible elisions.

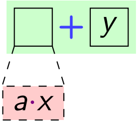
Presentation as Composition and Elision

Two steps of presentation

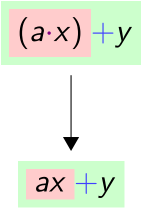
0. Content representations, built from *variables* and *symbols*, and *applications* and *binders*



1. 2D *composition* of presentations (formula tree → layout tree)



2. *elision* of parts that can be deduced from the context



Characteristics of Mathematical Symbols

- Visual appearance of a subformula determined by its operator; characteristics include:

  fixity: pre-/post-/in-/mixfix (e. g. $\Gamma \vdash_\Sigma t\colon \alpha$)

  brackets: left and right, mostly round.

  associativity: fully associative (like $+$), or left-/right-associative: $\alpha \to \beta \to \gamma := \alpha \to (\beta \to \gamma)$

- We consider *bracketed constructors* as *presentation components*, not as brackets in a strict sense: $]a; b],\ \{x \in \mathbb{N} | x > 5\},\ \binom{n}{k}, \ldots$

Notation Definitions

- XML content markup usually presented using XSLT templates.
- Two different approaches to save authors from coding XSLT:

Symbol-based (e. g. OMDoc1.2)

```
<presentation
 for="power" theory="arith1"
 role="application" class="1">
  <style format="TeX">
    <recurse select="*[2]"/>
    <text>^{</text>
    <recurse select="*[3]/>
    <text>}</text>
  </style>
<presentation>
```

Template-based (e. g. Naylor & Watt)

```
<Notation>
  <version style="1">
    <tex>\arg{a1}{n}^{\arg{a2}{m}}</tex>
  </version>
  <semantic-template>
    <OMA><OMS cd="power" name="arith1"/>
      <OMV name="n" id="a1/>
      <OMV name="n" id="a2/>
    </OMA>
  </semantic-template>
</Notation>
```

- Symbol-based approach is equivalent to depth-1 templates.
- In principle, one could do `(log e x)` → $\ln x$ with deeper templates, but in practice, this is a matter of *content* rewriting.

A Mixfix Presentation Model

ISABELLE models all symbol characteristics in one *mixfix declaration*:

$$f = w_0 \boxed{p_1|_{\pi_1}} w_1 \boxed{p_2|_{\pi_2}} \ldots \boxed{p_n|_{\pi_n}} w_n : p$$

- w_i: strings in the output language
- p: output precedence
- boxes: argument specifications; rendered recursively
- p_i: input precedences
- Typing judgment for LaTeX: $\boxed{p|_1}$\vdash_$\left\{\boxed{p|_2}\right\}$ $\boxed{p|_3}$: $\boxed{p|_4}$: p
- argument is bracketed when its operator binds weaker than the enclosing operator, i. e. $p_{out} > p_{in}$ (we use Prolog order)

Flexary Mixfixes

- OpenMath and Content MathML require flexible arities instead of Isabelle's fixed ones; e. g. the space of n-ary functions:

$$\boxed{\times : p - 1|_1^{n-1}} \longrightarrow \boxed{p|_n} : p$$

0 Content representation

```
<OMA>
  <OMS name="nfuns" cd="typ"/>
  <OMV name="A"/>
  <OMV name="B"/>
  <OMV name="C"/>
  <OMA>
    <OMS name="nfuns" cd="typ"/>
    <OMV name="D"/>
    <OMV name="E"/>
  </OMA>
</OMA>
```

1 Composition

$$A \times B \times C \rightarrow (D \rightarrow E)$$

2 Elision

$$A \times B \times C \rightarrow D \rightarrow E$$

Flexible Elisions

Situations where only part of the presentation is desired:

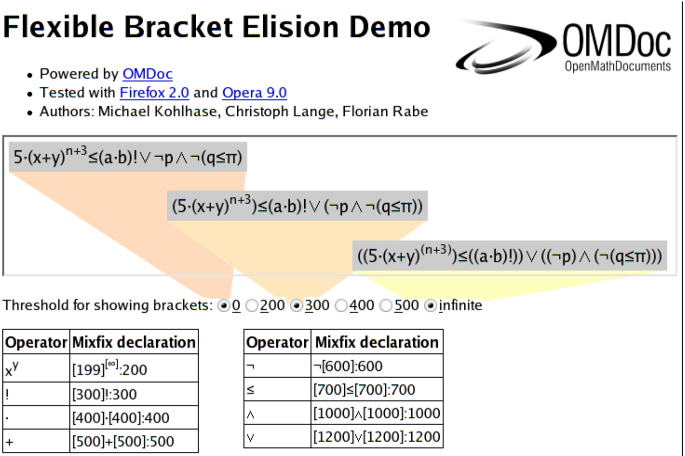
- redundant brackets due to operator precedences
- arguments have default values: $\log x = \log_{10} x$
- arguments' values can be inferred from other arguments
- arguments required, but readers can still infer them from the context: $[\![t]\!] = [\![t]\!]^{\phi}_{\mathcal{M}}$

Experts want more elisions than beginners ⇒ make them flexible!

- *visibility level* (for brackets: *precedence difference*; high level = high elidability) per *elision group* (e. g. "brackets")
  User can choose visibility threshold per group
- static output format (e. g. dead tree): choice at generation
- dynamic output format: elision annotations; interactive choice

Flexible Elisions in XHTML+JavaScript

Elidable brackets initially hidden; adjustable threshold for showing them



Flexible Bracket Elision Demo

- Powered by OMDoc
- Tested with Firefox 2.0 and Opera 9.0
- Authors: Michael Kohlhase, Christoph Lange, Florian Rabe

$5\cdot(x+y)^{n+3}\leq(a\cdot b)!\vee\neg p\wedge\neg(q\leq\pi)$

$(5\cdot(x+y)^{n+3})\leq(a\cdot b)!\vee(\neg p\wedge\neg(q\leq\pi))$

$((5\cdot(x+y)^{(n+3)})\leq((a\cdot b)!))\vee((\neg p)\wedge(\neg(q\leq\pi)))$

Threshold for showing brackets: ● 0 ○ 200 ● 300 ○ 400 ○ 500 ● infinite

Operator	Mixfix declaration
x^y	$[199]^{[\infty]}$:200
!	[300]!:300
·	[400]·[400]:400
+	[500]+[500]:500

Operator	Mixfix declaration
¬	¬[600]:600
≤	[700]≤[700]:700
∧	[1000]∧[1000]:1000
∨	[1200]∨[1200]:1200

An XML Encoding for Flexary Mixfix Declarations

Extensions to the declarative OMDoc syntax for presentations . . .

- making it more expressive
  (flexary mixfixes; embedded XSLT fragments no longer necessary ☺)
- allowing for flexible elisions
  (elision groups and visibility levels)

How is the notation definition for a symbol determined?

1. Look up a presentation for the resp. symbol and role.
2. Otherwise use “default” presentation for the home theory.
3. If there is more than one presentation: choice is non-trivial; see [Kohlhase/Müller/Müller] at MathUI.

Generating Presentations for Content Objects

Example: the typing jugdment $\Gamma \vdash_{\Sigma} t : T$ in LaTeX:

```
<symbol name="typing-judgment" role="application"/>
<presentation for="#typing-judgment" role="application" format="latex">
  <arg pos="1"/>
  <text>\vdash_{</text><arg pos="2"/><text>}</text>
  <arg pos="3"/>
  <text>:</text>
  <arg pos="4"/>
</presentation>
```

Input:

```
<OMA>
  <OMS name="typing-judgment" cd="typ"/>
  <OMS name="emptyset" cd="sets"/>
  <OMV name="Σ"/>
  <OMS name="true" cd="boolean"/>
  <OMS name="Boolean" cd="boolean"/>
</OMA>
```

Output:

```
\emptyset\vdash_{Σ}
  \mathit{true}:\mathit{Boolean}
```

Rendered: $\emptyset \vdash_{\Sigma} \mathit{true} : \mathit{Boolean}$

Generating Presentations for Content Objects

Example for flexary notation and multiple output formats:

```
<symbol name="times" role="application"/>
<presentation for="#times" role="constant" format="ascii">
  <text>*</text>
</presentation>
<presentation for="#times" role="constant" format="latex">
  <text>\ast</text>
</presentation>
<presentation for="#times" role="application"
 precedence="400" format="ascii latex">
  <text egroup="lbrack">(</text>
  <map begin="1" end="-1">
    <separator><arg pos="0"/></separator>
    <recurse precedence="400"/>
  </map>
  <text egroup="rbrack">)</text>
</presentation>
```

Input:

```
<apply><power/>
  <apply><times/>
    <ci>x</ci><ci>y</ci>
  </apply>
  <cn>2</cn>
</apply>
```

Output:

LaTeX: $(a * b)^2$

ASCII: (a*b)^2

Generating Presentations for OpenMath Objects

Bracket elision in Presentation MathML:

```
<presentation for="#plus" precedence="500">...</presentation>
<presentation for="#times" precedence="400">
  <element name="mo" egroup="lbrack">
    <text>(</text>
  </element>
  ...
</presentation>
```

Input:

```
<OMA>
  <OMS name="plus" cd="arith1"/>
  <OMA>
    <OMS name="times" cd="arith1"/>
    <OMV name="a"/>
    <OMV name="x"/>
  </OMA>
  <OMV name="y"/>
</OMA>
```

Output:

```
<mrow>
  <mrow>
    <mo style="display:none"
     omdoc:elevel="100">(</mo>
    <mi>a</mi><mo>·</mo><mi>x</mi>
    <mo style="display:none"
     omdoc:elevel="100">)</mo>
  </mrow>
  <mo>+</mo><mi>y</mi>
</mrow>
```

Conclusion and Outlook

- Content-oriented representation formats are independent from a specific output format
- Human-oriented presentations can be *generated*, w. r. t. user preferences, device constraints, . . .
- Need presentation algorithms that are: knowledge-based, extensible, adaptive, mathematical, efficient.
- Declarative notation definitions are most manageable.
- More general topic: *abbreviation*/*ellipses*
- Problem not addressed here: reverse presentation (*parsing*)
- Prototype implemented, evaluation in progress
  $\leadsto$ MathML 3 recommendation

References

- Kohlhase: OMDoc – An open markup format for mathematical documents [version 1.2] (2006)
- Kohlhase, Müller Ch., Müller N.: Documents with flexible notation contexts as interfaces to mathematical knowledge (2007)
- Manzoor, Libbrecht, Ullrich, Melis: Authoring Presentation for OpenMath (2005)
- Naylor, Watt: Meta style sheets for the conversion of mathematical documents into multiple forms (2001)
- Paulson: Isabelle reference manual (2005)

Direct Specification of Symbol Characteristics

- Syntactical sugar for mixfix notation
  - e. g. right-associative infix: $\boxed{p-1|_1} \rightarrow \boxed{p|_2} : p$
  - other pre-defined characteristics: bracket style, pre-/post-/infix
  - bracket styles for pre-/postfix: mathematical like $f(x)$, or LISP: (fx)

```
<presentation for="#arrow" format="ascii" role="application">
  <use fixity="infixr">
    <lbrack>(</lbrack>
    <rbrack>)</rbrack>
    <operator><text value=" -&gt; "/></operator>
  </use>
</presentation>
```

- Compatible to OMDoc 1.2; OpenMath standard content dictionaries are supported
  - Note: embedded XPath/XSLT no longer necessary and thus no longer supported!

A Template-Based Approach to Flexary Mixfix Notations

- Sometimes, “deep” pattern matching *is* more powerful: $\sin^2 x$
- Compatible to ActiveMath – not syntactically but conceptually
- Re-use most of the syntax of the symbol-pased approach
- Same syntax for input and output specification ⇒ both presenting content and parsing presentation to content supported ☺

```
<presentation format="OM" for="#typing-judgment">
  <OMA><OMS cd="types" name="typing-judgment"/>
    <arg name="context"/>
    <arg name="sig"/>
    <arg name="term"/>
    <arg name="type"/>
  </OMA>
</presentation>
```

```
<presentation format="pmathml" for="#typing-judgment">
  <mrow>
    <arg name="context"/>
    <msub><mo>⊢</mo><arg name="sig"/></msub>
    <arg name="term"/>
    <mo>:</mo>
    <arg name="type"/>
  </mrow>
</presentation>
```

(Note: literally included `<element>` constructors!)