Proposal for a set of pragmatic notation definitions

Michael Kohlhase

September 20, 2008

Abstract

In this note, we propose abbreviations of notation definitions that are simpler to write and more declarative as well.

Contents

1 Introduction

The notation definitions presented in [?] allow to represent a comprehensive set of notations by pattern matching and has been implemented in the JOMDoc system [?]. One of the advantages of this system is that gives common tasks like bracket elision a straightforward transformation interpretation. However, in many cases, the notation definitions become unwieldy and clumsy for simple tasks like specifying that addition is associative with precedence 500; for such cases, we would like introduce syntactic shortcuts that are simpler to write and maintain. Here the strict/pragmatic distincition introduced in OMDoc1.6 comes to the rescue, we consider the notation definitions of [?] as *strict OMDoc* and the ones we propose here as *pragmatic OMDoc*. The idea is that the pragmatic notation definitions are translated into strict ones via the rules given below during the presentation process, so that the translation is the only extension necessary to the presentation process.

2 Declarative Notation Declarations

The concrete proposal is based on extensive practice in the STEX presentation package [?] that has been used to mark up more than 1700 symbols in the context of slides and notes for lectures and talks given by the KWARC group. Surprisingly the proposal in this note comes out relatively close to the notation proposal of OMDoc1.2 [?] and the notation proposal we made earlier in [?].

In a nutshell we use mixfix declarations for symbols referenced by the OpenMath triple (`cd`, `name`, and `cdbase`). The attribute `args` specifies the number of arguments, and the attribute `assoc` allows us to specify one argument as associative.

the attributes @p @pi @pii @piii, ... give outer and argument precedences. Finally, the role is the OM symbol role. The pragmatic elements are called mixfix* where * is a specification of the arities of the argumunds, i.e. a string of characters 'i' (arity 1 argument) and one character 'a' an associative sequence of arguments. Here is the strict-pragmatic translation for ¡mixfixai¿, which is e.g. used for funtype (I will make the concrete example later).

pragmatic	strict
<mixfix format="⟪foo⟫" args="⟪n⟫" assoc="⟪m⟫" name="⟪nam⟫" cd="⟪cd⟫" cdbase="⟪URI⟫" role="application" precedence="⟪p⟫"> ⟪rendering⟫ </mixfix>	<**notation**> <**prototype**> <OMA> <OMS cd="⟪cd⟫" name="⟪nam⟫" cdbase="⟪URI⟫"/> <**expr** name="arg1"/>...<**expr** name="arg⟪m − 1⟫"/> <exprlist name="arg⟪m⟫"> <**expr** name="aargs"/> </exprlist> <**expr** name="arg⟪m + 1⟫"/>...<**expr** name="arg⟪n⟫"/> </OMA> </**prototype**> <**rendering** format="⟪foo⟫" precedence="⟪p⟫"> ⟪rendering⟫ </**rendering**> </**notation**>

[1] EdN:1

The next level of pragmatism would be another abbreviation, e.g. `infix` for `mixfix`, in general

pragmatic	strict
<infix format="⟪foo⟫" name="⟪nam⟫" cd="⟪cd⟫" cdbase="⟪URI⟫" precedence="⟪n⟫" [elide="⟪e⟫"]> ⟪op⟫ </infix>	<mixfix format="⟪foo⟫" args="2" name="⟪nam⟫" cd="⟪cd⟫" cdbase="⟪URI⟫" precedence="⟪n⟫" role="application"> ⟪foo − group − open⟫ <**render** name="arg1" precedence="⟪l⟫/> ⟪op⟫ <render name="arg2" precedence="⟪r⟫/> ⟪foo − group − close⟫ </mixfix>

[1] EdNote: @Florian, you had something about implicit arguments. I am not sure how this needs to be integrated here; probably for the stuff below only.

Where the `elide` attribute is optional and can take the values `left` and `right`. If no `elide` is given, then $\langle\!\langle n \rangle\!\rangle = \langle\!\langle l \rangle\!\rangle = \langle\!\langle r \rangle\!\rangle$, if `elide` has the value `left`, then $\langle\!\langle l \rangle\!\rangle = \langle\!\langle n \rangle\!\rangle - 1$ and $\langle\!\langle r \rangle\!\rangle = \langle\!\langle n \rangle\!\rangle$ and if if `elide` has the value `right`, then $\langle\!\langle r \rangle\!\rangle = \langle\!\langle n \rangle\!\rangle - 1$ and $\langle\!\langle l \rangle\!\rangle = \langle\!\langle n \rangle\!\rangle$.[2] EdN:2

This would allow to specifiy the presentation for a binary plus operator that elides brackets to the left (sometimes called an 'infixl' operator) as

```
<infix format="TeX" cd="arith1" name="plus" precedence="500" elide="left">+</infix>
```

which is really much more pragmatic than the strict verion. Here we made use of the idea that the argument precedences are inherited from the `precedence` attribute. For prefix operators we have another abbreviation:

pragmatic	strict
<prefix format="⟪*foo*⟫" name="⟪*nam*⟫" cd="⟪*cd*⟫" cdbase="⟪*URI*⟫" precedence="⟪*n*⟫" precargs="⟪*m*⟫"> ⟪*op*⟫ </infix>	<mixfix format="⟪*foo*⟫" args="1" assoc="1" name="⟪*nam*⟫" cd="⟪*cd*⟫" cdbase="⟪*URI*⟫" precedence="⟪*n*⟫" role="application"> ⟪*foo − group − open*⟫ ⟪*op*⟫ <iterate name="arg1"> <**render** name="aargs"/> </iterate> ⟪*foo − group − close*⟫ </mixfix>

and analogously for postfix operators.

[2] EdNote: or the other way around, I am a bit confused at the moment.

3 Examples

The example below is mainly used to show some pragmatic notation elememnts and test the schema.

```
<?xml version="1.0" encoding="utf-8"?>

<omdoc xml:id="mixfix.omdoc" version="1.6"
       xmlns:dc="http://purl.org/dc/elements/1.1/"
       xmlns:cc="http://creativecommons.org/ns"
       xmlns:m="http://www.w3.org/1998/Math/MathML"
       xmlns="http://omdoc.org/ns">

<metadata>
  <dc:title>A Test for Pragmatic Notation Elements</dc:title>
  <dc:creator role="trl">Michael Kohlhase</dc:creator>
  <dc:date action="updated">2008-09-20</dc:date>
  <dc:format>application/omdoc+xml</dc:format>
  <dc:rights>Copyright (c) 2008 Michael Kohlhase</dc:rights>
  <!--
  Id: mixfix.omdoc80562008-09-2005:31:30Zkohlhase
  HeadURL: https://svn.omdoc.org/repos/omdoc/trunk/doc/blue/pres-pragmatic/examples/mixfix.omdoc
  -->
</metadata>

<mixfix format="MathML" role="application"
    cd="lambda-calc" name="well-typed"
    args="3" assoc="1">
  <m:mrow>
    <iterate name="arg1">
      <separator><m:mo>,</m:mo></separator>
      <render name="aargs"/>
    </iterate>
    <m:mo>&amp;#x22A2;</m:mo>
    <render name="arg2"/>
    <m:mo>:</m:mo>
    <render name="arg3"/>
  </m:mrow>
</mixfix>

<infix cd="arith1" name="plus" precedence="500" elide="left">+</infix>
<prefix cd="arith1" name="unary_minus" precedence="700">-</prefix>
<prefix cd="arith1" name="factorial" precedence="500">!</prefix>
</omdoc>
```

4 An extension for the OMDoc RelaxNG Schema

```
# A RelaxNG for Open Mathematical documents (OMDoc 1.6) Module PRES (pragmatic extensions)
# Id: pragpres.rnc80562008-09-2005:31:30Zkohlhase
# HeadURL: https://svn.omdoc.org/repos/omdoc/trunk/doc/blue/pres-pragmatic/pragpres.rnc
# See the documentation and examples at http://www.omdoc.org
# Copyright (c) 2004-2008 Michael Kohlhase, released under the GNU Public License (GPL)

default namespace omdoc = "http://omdoc.org/ns"
include "../../../rnc/omdoc.rnc"

omdoc.class |= mixfix | infix | prefix | postfix

args.att = attribute args {xsd:positiveInteger},
           attribute assoc {xsd:positiveInteger}?
synrole = "constant" | "application" | "binder" | "attribution" | "error"
role.att = attribute role {synrole}

mixfix =  element mixfix {triple.att,args.att,role.att,rendering.att,renderexp}

elide.att = attribute elide {"left" | "right"}

infix = element infix {triple.att,rendering.att,elide.att?,opexp}

prefix = element prefix {triple.att,rendering.att,opexp}
postfix = element postfix {triple.att,rendering.att,opexp}

opexp = grammar{include "../../../rnc/mathml3/mathml3-presentation.rnc" {start = PresExp}}|text
```