Towards Bootstrapping the Pragmatic to Strict Mapping in OMDoc

Michael Kohlhase
Jacobs University Bremen
http://kwarc.info/kohlhase

August 11, 2009

Abstract

We propose a representational infrastructure to make the OMDoc2 pragmatic-to-strict mapping that is fixed in OMDoc1.6 configurable. This will allow to legitimize many of the pragmatic OMDoc features from foundational metalogics and thus make OMDoc2 more flexible and expressive.

1 Introduction

The OMDoc2 format currently under development at KWARC has two sub-languages: *strict* OMDoc — a conceptually minimal set of mathematically well-understood representational primitives [RK09], and *pragmatic* OMDoc — a representational infrastructure that attempts a more pragmatic balance between theoretical expressivity and conceptual familiarity. The meaning of pragmatic OMDoc is given by the $\mathcal{P2S}$ mapping which interprets all pragmatic language features in terms of strict ones.

The strict/pragmatic language design was pioneered by the author for the upcoming MathML3 recommendation [Aus+09], and is used in OMDoc1.6 [Koh09], the first step towards OMDoc2. This language design has the advantage that only a small, regular sublanguage has to be given a mathematical meaning, but a larger vocabulary that is more intuitive to practicioners of the field can be used for actual representation. Moreover, semantic services like validation only need to be implemented for the strict subset and can be extended to the pragmatic langauge by translation. Ultimately, a representation format might even have multiple pragmatic front-ends geared towards different audiences. These are semantically interoperable by construction.

In OMDoc1.6, the $\mathcal{P2S}$ mapping is fixed by the specification; severely restricting the benefits of the strict/pragmatic language design. Moreover, the specification revelaed interactions with the respective foundational meta-logics that cannot be accounted for with a $\mathcal{P2S}$ mapping. Consider for instance the pragmatic element of an "implicit definition" which defines a mathematical object by describing it so accurately, that there is only one object that fits this description: the definiendum. For instance the exponential function is defined as the (unique) solution of the differential equation $f = f'$ with $f(0) = 1$. This form of definition is extensively used in practical mathematics, so pragmatic OMDoc should offer an infrastructure for it. Strict OMDoc only offers "simple definitions" of the form $c := d$, where c is a new symbol (the definiendum) and an expression d (the definiens) which does not contain c. In the OMDoc1.6 $\mathcal{P2S}$ mapping the implicit definition for the exponential function is mapped to the simple definition $e := \tau f.(f' = f \wedge f(0) = 1)$, where τ is a "definite description operator": Given an expression A with free variable x, such that there is a unique x that makes A valid, $\tau x.A$ returns that x, otherwise $\tau x.A$ is undefined. Note the $\mathcal{P2S}$ mapping is only justified, if the current meta logic supplies a definite description operator. In most areas of mathematics, this is unproblematic, since its existence is implicitly assumed. In more foundational contexts such as the philosophy of mathematics this poses a problem, since it prevents the representation of systems without a definite description operator.

To alleviate this and other problems, it seems natural to make the $\mathcal{P}2\mathcal{S}$ mapping configurable in the OMDoc2 format itself. All the more since we can extend an existing OMDoc2 mechanism for this: format transformations from the notation definition subsystem.

2 A Configurable $\mathcal{P}2\mathcal{S}$ Mapping

In [KMR08] we have presented a presentation system for OMDoc. It has been implemented in the JOMDoc system and became a part of the OMDoc1.6 format. In a nutshell, the system uses *notation definitions* such as the one in Listing 1. This is essentially a transformation rule that transforms OpenMath expressions into presentation MathML ones. Crucially, notation definitions are treated like statements in OMDoc; in particular they are tied to theories with which they are inherited.

Listing 1: A notation for a binomial coefficient

```
<notation>
  <prototype>
    <om:OMA>
      <om:OMS cd="arith1" name="binomial"/>
      <expr name="arg1"/>
      <expr name="arg2"/>
    </om:OMA>
  </prototype>
  <rendering format="mathml−presentation+xml">
    <m:mrow>
      <m:mfrac linethickness="0">
        <render name="arg1"/>
        <render name="arg2"/>
      </m:mfrac>
    </m:mrow>
  </rendering>
</notation>
```

For a configurable $\mathcal{P}2\mathcal{S}$ mapping, we propose to use the same mechanism; just for intra-OMDoc transformations. In our example, we would add a $\mathcal{P}2\mathcal{S}$ transformation rule to a meta-theory that supplies a description operator. Listing 2 shows a fragment of such a theory based on first-order logic.

Listing 2: A Meta Theory that supplies implicit definitions

```
<theory name="descriptions">
  <import from="fol.omdoc?fol"/>
  <constant name="that" role="binding"/>
  <notation>
    <prototype><om:OMS cd="descriptions" name="that"/></prototype>
    <rendering><m:mo>τ</m:mo></rendering>
  </notation>
  <axiom name="that−ax">∀y.τx.(x = y) = y</axiom>
  <p2srule>
    <prototype>
      <definition type="implicit">
        <attribute name="name"><expr name="name"/></attribute>
        <attribute name="var"><expr name="var"/></attribute>
        <expr name="content"/>
      </definition>
    </prototype>
    <rendering>
     <constant>
       <attribute name="name"><text name="name"/></attribute>
       <definition>
         <om:OMBIND>
           <om:OMS cd="descriptions" name="that"/>
           <om:OMBVAR>
             <om:OMV><attribute name="name"/><text name="var"/></attribute></om:OMV>
           </om:OMBVAR>
           <render name="content"/>
         </om:OMBIND>
       </definition>
    </rendering>
  <p2srule>
</theory>
```

The theory `descriptions` introduces the definite description operator as a new binding operator τ, and describes its meaning by an axiom $\forall y.\tau x.(x = y) = y$. The new pragmatic syntax is introduced in the first child of a new element `p2srule`, which is just a variant of the `notation` element the two kinds of transformation rules will probably have to be handled differently, therefore the separate elements. This rule will transform the new pragmatic syntax.

```
<definition type="implicit" name="expfun" var="f">f′ = f ∧ f(0) = 1</definition>
```

into the strict syntax

```
<constant name="expfun">
 <definition>τf.f′ = f ∧ f(0) = 1</definition>
</constant>
```

in a situation where the theory `descriptions` is imported as a meta-theory. Importantly, strict OMDoc supports structural validation, which now extends to the new syntax.

Note that while the introduction of implicit definition via a simple first-order description operator works in principle, it does not capture the structure of implicit definitions in OMDoc1.2 [Koh06], which uses attributes `existence` and `uniqueness` as pointers to assertions stating unique existence. In our example above, unique existence assertions are also needed to invoke the the axiom `that-ax`: $\exists^1 f.f' = f \wedge f(0) = 1$ can be used to show that the sets $\{f \big| f' = f \wedge f(0) = 1\}$ and $\{f \big| f = e\}$ where e is the exponential function are equal and thus that $e = \tau f.f' = f \wedge f(0) = 1$. So any further mathematical development of the exponential function will need this assertion, but the syntax defined by the $\mathcal{P2S}$ mapping in Listing 2 has no structural information to locate them. In fact in mathematical texts, the association is only made informally in the accompanying text.

As for machine-processing of mathematical knowledge the accompanying text is opaque, we will try to re-gain the structural constraints using a more structural meta-logic: dependently typed first-order logic (a dependently typed meta-logic [Ban03] is used in the Mizar sytem [Miz] one of the larges mathematics formalization projects). In DFOL[1] we can introduce a definite description operator ι of type $\Pi\alpha.\Pi A\colon(\alpha \to o).pf(\exists^1 A).\alpha$[2] Intuitively, ι takes as arguments a type α, a set A, and a proof that A has a unique member a (i.e. is a singleton) and returns the unique member of that set. This would give rise to a translation[3] like the one in Listing **??**. EdNote(1) EdNote(2) EdNote(3)

Listing 3: A Meta Theory for structured implicit definitions

```
<p2srule>
  <prototype>
    <definition type="implicit">
      <attribute name="name"><expr name="name"/></attribute>
      <attribute name="var"><expr name="var"/></attribute>
      <attribute name="exunique"><expr name="exunique"/></attribute>
      <expr name="idef"/>
    </definition>
    ...
    <type>
      <attribute name="for"><expr name="name"/></attribute>
      <expr name="tcont"/>
    </type>
    ...
    <proof>
      <attribute name="name"><expr name="exunique"/></attribute>
      <expr name="pcont"/>
    </proof>
  </prototype>
  <rendering>
   <constant>
     <attribute name="name"><text name="name"/></attribute>
     <definition>
       <om:OMA>
         <om:OMS cd="descriptions" name="iota"/>
         <render name="tcont"/>
         <om:OMBIND>
```

[1] EdNote: cite Florian's papers here

[2] EdNote: check if that is really the right type.

[3] EdNote: hmmmm, I am using λ here, I am not sure that this exists in DFOL

```
        <om:OMS cd="dfol" name="lambda"/>
        <om:OMBVAR>
          <om:OMV><attribute name="name"/><text name="var"/></attribute></om:OMV>
        </om:OMBVAR>
        <render name="idef"/>
      </om:OMBIND>
      <render name="pcont"/>
    </om:OMA>
  </definition>
</rendering>
<p2srule>
```

This already shows added difficulties for the implementation of the $\mathcal{P2S}$ mapping: The prototype contains three elements that together correspond to the strict definition. Naturally, the pragmatic syntax does not want to restrict itself to having these three elements in a consecutive block (a fact that we have tried to gloss by using the ellipses ... in the `prototype` element).

3 Research & Design Problems

There are some problems that still have to be solved:

3.1 Multi-Part Prototypes

Like in Listing 3. How are they matched, over what scope?

3.2 Prioritization & Scoping & Generic Rules

How are the $\mathcal{P2S}$ rules prioritized and scoped? This is especially important, if we want to introduce generic $\mathcal{P2S}$ rules, e.g. for attributes. Consider for instance the pragmatic way of specifying rethorical metadata in OMDoc1.2 via the type attribute and related ones. For instance in OMDoc1.2 we would have

```
<omtext type="transition" from="foo" to="bar">...</omtext>
```

which would be represented as

```
<omtext>
  <meta property="omdoc:transition"/>
  <link rel="omdoc:transition-from" href="foo"/>
  <link rel="omdoc:transition-to" href="bar"/>
  ...
</omtext>
```

Note that these attributes might give rise to $\mathcal{P2S}$ transformation rules in a generic `omdoc` theory via

```
<p2srule>
  <prototype>
    <element name="name">
      <attribute name="type"><text name="type"/></attribute>
      <attributes name="arest"/>
      <exprlist name="crest"/>
    </element>
  </prototype>
  <rendering>
    <element name="name"><attributes name="arest/>
    <meta><attribute name="property"><text>omdoc:</text><text name="type"/></attribute></meta>
    <iterate name="crest"/>
  </rendering>
</p2srule>
```

and similar ones for the `for` and `from` attributes. Note that in contrast to the $\mathcal{P2S}$ rules shown above, these rules are not bound to a specific new element, but introduce new attributes[4]. So we might also contemplate the following transformation rule syntax: EdNote(4)

[4] EdNote: is that syntax above a good way of talking about elements <*>

```
<p2srule>
  <prototype>
    <attribute name="type"><text name="type"/></attribute>
  </prototype>
  <rendering>
    <meta><attribute name="property"><text>omdoc:</text><text name="type"/></attribute></meta>
  </rendering>
</p2srule>
```

3.3 User Extensions

Do we allow the user (as opposed to the language designer) to extend the syntax? If we did, we might end up with a situation like in LaTeX, where we have thousands of OMDoc packages and classes.

3.4 Documentation

How are the $\mathcal{P2S}$ rules documented for the language user? There should be some form an explanation system for the currently active transformation, especially, when we have user extensions as envisioned in section 3.3

4 Conclusion

I have proposed a novel, more semantic way of handling the OMDoc2 pragmatic to strict mapping. Even though some practical problems still have to be solved (see Section 3), the approach points to a much more modular and flexible way of developing the OMDoc language. From all the examples that the author is aware, it seems to scale to interpreting the whole OMDoc1.2 vocabulary as pragmatic OMDoc2. In fact, the new idea of having pragmatic extensions be meta-theory induced points towards the idea that OMDoc metadata and the corresponding metadata ontologies [LK09] are actually meta-theories as well (albeit at a somewhat different level). BegNP(5)

In fact we can go even further and interpret the features of types and definitions that are built into strict OMDoc1.6 [RK09] as pragmatic extensions of an even more foundational system: definitions are just pragmatic notations for axioms of the form $c = t$[6], which are introduced by EdNote(6) the meta-theory that supplies the equality relation. Axioms in turn are just type statements of the form $true(A)$, so they EndNP(5)

References

[Aus+09] Ron Ausbrooks et al. *Mathematical Markup Language (MathML) Version 3.0.* W3C Working Draft of 24. September 2009. World Wide Web Consortium, 2009. URL: `http://www.w3.org/TR/MathML3`.

[Ban03] Grzegorz Bancerek. "On the structure of Mizar types". In: *Electronic Notes in Theoretical Computer Science* 85.7 (2003).

[KMR08] Michael Kohlhase, Christine Müller, and Florian Rabe. "Notations for Living Mathematical Documents". In: *Intelligent Computer Mathematics, 9th International Conference, AISC 2008 15th Symposium, Calculemus 2008 7th International Conference, MKM 2008 Birmingham, UK, July 28 - August 1, 2008, Proceedings.* Ed. by Serge Autexier et al. LNAI 5144. Springer Verlag, 2008, pp. 504–519. URL: `http://omdoc.org/pubs/mkm08-notations.pdf`.

[5] New Part: this is still quite sketchy; must discuss with Florian; also in terms of what this explains about foundations. Indeed the foundations supply type and equality judgments; just the relations that are extensions. What would a real foundational MMT look like?

[6] EdNote: Here Fulya's declaration patterns come in: t may not contain c.

[Koh06] Michael Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2]*. LNAI 4180. Springer Verlag, 2006. URL: `http://omdoc.org/omdoc1.2.pdf`.

[Koh09] Michael Kohlhase. "An Open Markup Format for Mathematical Documents OMDoc [Version 1.6 (pre-2.0)]". Draft Specification. 2009. URL: `https://svn.omdoc.org/repos/omdoc/trunk/doc/spec/main.pdf`.

[LK09] Christoph Lange and Michael Kohlhase. "A Mathematical Approach to Ontology Authoring and Documentation". In: *MKM/Calculemus 2009 Proceedings*. Ed. by Jacques Carette et al. LNAI 5625. Springer Verlag, 2009, pp. 389–404. URL: `https://svn.omdoc.org/repos/omdoc/trunk/doc/blue/foaf/mkm09.pdf`.

[Miz] *Mizar*. URL: `http://www.mizar.org` (visited on 11/16/2009).

[RK09] Florian Rabe and Michael Kohlhase. "A Web-Scalable Module System for Mathematical Theories". Manuscript, to be submitted to the Journal of Symbolic Computation. 2009. URL: `https://svn.kwarc.info/repos/kwarc/rabe/Papers/omdoc-spec/paper.pdf`.