The OpenMath Thematic Network *IST–2000–28719*

Date: November 2003

Another Look at Formal Mathematical Properties

James Davenport

(`jhd@maths.bath.ac.uk`).

Contents

1 History

At the Twelth OpenMath Workshop (Eindhoven meeting 15/16.6.1999), there was a discussion about Formal Mathematical Properties. The minutes read as follows.

> AMC introduced a paper by himself and MK on "Defining Mathematical Properties". He said that CDs did not necessarily introduce the logical meaning of mathematical symbols. OpenMath should involve the logic community more. While OpenMath has Formal Mathematical Properties (FMPs), there is no differentiation between definitions and consequences. Also, some objects do not have FMPs, e.g. subset. He suggested a new tag, `DefMP`, which would be like FMPs, but the `DefMP`s would have to define the mathematical object uniquely. At least in theory, the FMPs would then be formally proved as consequences of the `DefMP`s.
>
> In the Esprit group, there were two objections: one that they would scare many potential users, and the other was that peope might want different `DefMP`s. To the first, he answered that there were many features of OpenMath that not everyone used. For the second, he noted that signatures had been moved to separate files, and maybe this would be appropriate for `DefMP`s.
>
> This led to a lively debate. GHG said that placing the `DefMP`s in separate files was a move against the general trend towards databases. MK in particular called for genuinely usable OpenMath tools, e.g. for Reduce and Maple. Many agreed with him.
>
> [Irrelevancies deleted.] MK proposed that, in the light of the DefMP discussion above, which seemed to conclude that the `DefMP`s should be in auxiliary files, `FMP`s should be moved to a different kind of file. AMC agreed, but DPC did not. SB proposed, and JHD seconded, that `FMP`s should stay where they were. This was agreed. A few amendments to the DTD for CDs were noted. DPC pointed out that 5.4 (CD Signature files) and 5.5 (CD Groups) were probably not final. MK suggested a DTD for `defmp` files, which would be inserted after 5.4, after it had been disucssed by e-mail. AS suggested that some tags like CDVersion should also be present in signature files.

2 Kinds of symbols

In the last few years, JHD has come to understand more of the motivation behind the DefMP proposal, and wishes to resurrect a variant of it, in which FMPs would be qualified with some description of their rôle. It seems to JHD that there are various kinds of symbol.

1. Those that are fundamentally primitive, and not defined at all. They may still have FMPs, but these FMPs are merely about them, rather than defining the symbol. An example would be

```
<OMS name="set" cd="set1"/>.
```

2. Those that OpenMath treats as primitive, and not defined at all in OpenMath. These might not be primitive in mathematics, but OpenMath has decided not to define them. They may still have FMPs, but these FMPs are merely about them, rather than defining the symbol. An example would be

   ```
   <OMS name="exp" cd="transc1"/>,
   ```

   whose only FMP is a representation of $\forall k \in \mathbf{Z} \exp(z + 2k\pi i) = \exp(z)$ (which is equally true of $\exp(2z)$ for example).
3. At the other end of the spectrum, there are those objects that OpenMath defines (because mathematicians use them) but which are logically redundant. An example of this is

   ```
   <OMS name="sin" cd="transc1"/>,
   ```

   whose FMP is a representation of $\sin(x) = \frac{\exp(ix)-\exp(-ix)}{2i}$, which means that all occurrences of `sin` can be removed from an OpenMath object without changing the semantics. *If the CD specified this,* a system which encountered a symbol like this could rewrite it knowing that there was no semantic loss.

   If it felt that sin is still “important”, and complex exponentials are not the right response to a real function, how about csc, which can be perfectly encapsulated via $\csc(x) = \frac{1}{\sin x}$?
4. It would be possible[1] (in fact the definition in `integer1` is not of this form, but rather in terms of products), to define

   ```
   <OMS name="factorial" cd="integer1"/>
   ```

   (whose STS states that it is a function $\mathbf{N} \to \mathbf{N}$) with an FMP encoding the recursive definition:

   ```
   <OMOBJ>
   <OMA>
   <OMS name="and" cd="logic1"/>
     <OMA>
       <OMS name="eq" cd="relation1"/>
       <OMA>
         <OMS name="factorial" cd="integer1"/>
         <OMS name="zero" cd="arith1"/>
       </OMA>
   ```

[1] If it is argued that this is artificial, since this is not in fact the FMP, consider the example of `Stirling1` in `combinat1`, whose FMP is the encoding of $Stirling1(n, m) = \sum_{k=0}^{n-m} (-1)^k * binomial(n-1+k, n-m+k) * binomial(2n-m, n-m-k) * Stirling2(n, m)$.

```
    <OMS name="one" cd="arith1"/>
  </OMA>
  <OMA>
    <OMS name="implies" cd="logic1"/>
    <OMA>
      <OMS name="gt" cd="relation1"/>
      <OMV name="n"/>
      <OMS name="zero" cd="arith1"/>
    </OMA>
    <OMA>
      <OMS name="eq" cd="relation1"/>
      <OMA>
        <OMS name="factorial" cd="integer1"/>
        <OMV name="n"/>
      </OMA>
      <OMA>
        <OMS name="times" cd="arith1"/>
        <OMV name="n"/>
        <OMA>
          <OMS name="factorial" cd="integer1"/>
          <OMA>
            <OMS name="minus" cd="arith1"/>
            <OMV name="n"/>
            <OMS name="one" cd="arith1"/>
          </OMA>
        </OMA>
      </OMA>
    </OMA>
  </OMA>
</OMA>
</OMOBJ>
```

In this case, it is possible to replace any particular numerical factorial by a computation, but it is impossible to replace, say $n!$ with a definition not involving factorials (unless one extracts some kind of Y-expression from that recursive definition, which is mere semantic trickery).

3 The OpenMath dilemma

The notation of mathematics is incredibly varied, and new notations and concepts are permanently being introduced. This poses problems for OpenMath's goal of encouraging interoperability between tools, and future-proofing of data.

4 Kinds of FMP

At the moment, the distinction we have made above is purely informal, and there are no clues in the CD as to the meaning of any FMP. The DefMp proposal mentioned above suggested that some FMPs were "defining", and should be treated differently. We propose a slightly weaker form: that some FMPs should be marked, and therefore *could* be treated specially. More concretely, we propose two special marks.

defining A **defining** FMP is one that can always be used as a definition of a symbol. An example of this is the FMP for `sin` mentioned above. In all contexts, it is legitimate to replace an occurrence of `sin` by the corresponding right-hand side. Such FMPs will generally begin with an `eq` operator, though this is not necessarily required. The following guarantees must be met by such an FMP.

- A symbol can have at most one of them.
- The replacement value must not, either directly or indirectly by a chain of such FMPs, involve the symbol being defined.

evaluating An **evaluating** FMP is one that can be used as a definition of how to evaluate a symbol on a concrete instance of its input argument(s). The following guarantees must be met by such an FMP.

- A symbol can have at most one of them.
- The replacement value must, after a finite number of applications of this, and any other evaluating or defining FMPs, lead to an expression free of the symbol being defined, whenever the symbol is applied to concrete instances of the correct type(s).

5 The requirements for uniqueness

These requirements could be seen as posing the following questions.

1. Why restrict to one defining FMP?
2. Why restrict to one evaluating FMP?
3. Can one have one defining *and* one evaluating?

The first two are required, in JHD's opinion, to avoid any ambiguity: if there are two definitions of a symbol, are they proven to be consistent? Note that, in the quote above, AMC called for greater interactions with the logic community. It may be that, in the fullness of time, we will be able to allow two defining FMPs accompanied (and there is currently no mechanism for doing this) with a machine-checkable proof of consistency.

The other reason for insisting on uniqueness is that a CD-reading tool, which has come across a symbol which its base application does not know, but which has a defining FMP, has no choice about what to do: it replaces it by the definition (and recurses if necessary). Otherwise the tool has to be far more complicated.

The third question also raises the question of consistency. However, it does not raise quite the same question of ambiguity, since such a tool would probably use an evaluating FMP if it (knew that it) had a definite value, and a defining one otherwise. Hence for the moment this proposal does not rule that out, though this could clearly be debated.

6 Concrete changes

We propose that `<FMP>` be allowed an attribute `type`, so that one could write

```
<FMP type="defining">
```

in the first case, and

```
<FMP type="evaluating">
```

in the second.

Existing systems could ignore these, but new systems might interpret them on the lines suggested above.

References

[1] Corless,R.M., Davenport,J.H., Jeffrey,D.J. & Watt,S.M., "According to Abramowitz and Stegun". *SIGSAM Bulletin* **34** (2000) 2, pp. 58–65.