Proposal for an extension of OpenMath by Defining Mathematical Properties

Arjeh Cohen, Technical University of Eindhoven, The Netherlands
Michael Kohlhase, University of Saarland, Germany

March 22, 1999

Abstract

In this note we propose a small extension of OpenMath that actually allows to *define* mathematical objects in content dictionaries. This is a step towards automation of consistency management of content dictionaries (which is presently a human refereeing process).

1 Introduction and Motivation

With the growth of the number of content dictionaries in OpenMath, the need will rise for tool support in managing the mathematical knowledge they contain. In particular it will become necessary to provide the developers of CDs with a means to ensure the consistency of CDs, since otherwise the ontological status of the CD itself will be compromised (anything is entailed by a contradiction). Moreover it seems simpler to guarantee the consistency than to specify the meaning of a CD containing inconsistent information.

In the current specification of content dictionaries, `CDDefinition` statements can be used to describe and specify properties of OpenMath symbols. This can be done informally using the `description` tag or formally using the `FMP` tag. In this note, we will only concentrate on the formal aspects, since we are interested in the formal verification of correctness properties of content dictionaries.

For instance in the content dictionary `arith`, the symbol `plus` is characterized by the property of being commutative, which leaves lots of room for interpretation (although a signature could help out a lot). Incidentally, the symbol `times` is specified in the same way in the content dictionary `comm`.

In mathematical logic (and in mathematical practice), a "definition" is taken to be the act of (uniquely) specifying a new concept in terms of known ones, and great care is taken to make sure that no inconsistencies can be introduced by definitions.

As first (very simple) example let us consider the content dictionary `set.ocd` for sets. In particular, let us look at the subset relation described there. Currently the symbol `subset` is described as a binary relation that "is the intuitive subset relation". The intended meaning of `subset` (namely that it corresponds to $\subseteq$) cannot be determined at the time of the definition. Only when we come across the definition of `prsubset` which represents the concept of a proper subset, does the intention become clear. In particular, it would be perfectly sensible to add the `FMP` that `subset` is irreflexive, which would contradict the intuition.

Elementary set theory defines the subset relation $\subseteq$ as

$$S \subseteq T, \text{ iff } x \in T \text{ forall } x \in S.$$

i.e., in terms of the (already defined) membership relation $\in$. This completely specifies the relation $\subseteq$, since from the form of the definition it is clear that it exists and that it is unique (this can perhaps best be seen by the equivalent definition

$$\subseteq := \lambda S \lambda T (\forall x. x \in S \Rightarrow x \in T),$$

where the right hand side is a *closed* term not containing $\subseteq$).

In particular, the formal mathematical property that $\subseteq$ is reflexive ($\forall S.S \subseteq S$) is then just a consequence of the definition. We can see that there are two kinds of formal mathematical property, defining ones and derived ones. Intuitively, we argue to make this distinction explicit in `CDDefinitions` and thus give the defining mathematical properties precedence over the derived ones.

2 The Proposal

We propose to extend the specification of the `CDDefinition` tag by a variant of the `FMP` tag. This `DefMP` is used completely analogously to `FMP` inside a `CDDefinition`, the only (crucial) difference is in the semantics, which we will specify now.

> If they are present, the set of `DefMP`s in the `CDDefinition` for a set of symbols[1] must determine the existence of a unique mathematical object for each symbol.

The importance of this new tag is that the property of unique existence can be checked formally, in many cases fully automatically by existing theorem proving systems. In some cases, such as the $\subseteq$ example above where the relation is given as a closed (λ)-term, this is nearly trivial, in some others (cf. sections 3.3 and 3.2) it may be quite involved.

Thus the current formal mathematical properties specified by the `FMP` tag have the status of making certain derived properties of the symbols explicit (they may be interpreted as theorems in the mathematical theory given by the `DefMP`s); They can be seen as sample properties or as challenges for automated theorem provers.

Concretely, we propose to augment the DTD for CDs given in Figure 4.1 of the current draft to be augmented with a line

```
<!ELEMENT DefMP     %inhel2; >    <!-- allow embedded OM -->
```

and change the clause for `CDDefinition` to

```
<!ELEMENT CDDefinition    (CDComment | Name | Description | PartOf |
                           FunctorClass | Signature | Example | FMP |
                           DefMP | CMP | Presentation)* >
```

3 Examples

In this section we will take a closer look at some concrete examples and work out the definitions.

3.1 Explicit Definitions; the Subset Relation

Since we have already talked about the definition of the subset relation, we will only give the XML encoding here:

```
<CDDefinition>
<Name> subset  </Name>
<Description> The subset relation we all know and love  </Description>
<DefMP>
   <OMOBJ>
     <OMA>
        <OMS cd="relation" name="eq"/>
        <OMS cd="set" name="subset">
```

[1] Note that the possibility of having multiple `Name` tags in a `CDDefinition` is warranted by the current OpenMath standard, at least in the DTD given in Figure 4.1 (not in the description after it). We will use this feature in the examples, e.g., for defining the natural numbers (cf. §3).

```
        <OMBind>
          <OMS cd="ecc" name="lambda"/>
          <OMBVAR><OMV name="S"/><OMV name="T"/></OMBVAR>
          <OMA>
             <OMBIND>
             <OMBIND>
               <OMS cd="quant" name="forall">
               <OMBVAR><OMV name="X"/></OMBVAR>
               <OMA>
                  <OMS cd="logic" name="implies">
                  <OMA><OMS cd="set" name="in"/><OMV name="X"/><OMV name="S"></OMA>
                  <OMA><OMS cd="set" name="in"/><OMV name="X"/><OMV name="T"></OMA>
               </OMA>
             </OMBIND>
          </OMA>
        </OMBIND>
      </OMA>
    </OMOBJ>
  </DefMP>
  <FMP> ∀ S. S ⊆ S </FMP>
  </CDDefinition>
```

As we have stated above, the unique existence property is very simple to check: we only have to make sure that the `CDDefinition`[2] only introduces one symbol (there is only one `Name` tag) and that the OpenMath object in the `DefMP` is an equation, where the left-hand side is the symbol and the right-hand side is a closed (λ)-term that does not contain that symbol.

This kind of explicit definition goes a long way, and allows to specify a lot of useful mathematics directly. In fact, it is commonly believed that using a few well-understood logical tools like axiomatic set theory or the description and choice operators (as undertaken in the Principia Mathematica), *all* of mathematics can be formalized this way. However, the form is often quite unnatural, therefore there are a few other commonly used definition mechanisms in mathematics that OpenMath may want to support. The most general one is that of so-called implicit definitions, which we will look at in the next paragraph.

3.2 Implicit Definitions; the Functio sin

The probably most prominent example of an implicit definition is the definition of a mathematical object as the (unique) solution of an equation. For instance the $\sqrt{2}$ is the unique positive solution of $x^2 = 2$. This would be expressed in a content dictionary on algebraic numbers as

```
<CDDefinition>
<Name> sqr-two  </Name>
<Description> The square root of two </Description>
<DefMP>
   <OMOBJ>
     <OMA>
        <OMS cd="˜"relation" name="eq"/>
        <OMA>
          <OMS cd="arith" name="power"/>
          <OMS cd="arith-num" name="sqr-two">
          <OMI>2</OMI>
        </OMA>
        <OMI>2</OMI>
     </OMA>
```

[2] Syntactical variations of this construction which make this kind of definition more readable are possible. We could define a new "definition symbol" `defines` in the CD `basic`, which is a binding object that in our example binds the variables S and T and makes the λ obsolete.

```
   </OMOBJ>
</DefMP>
<DefMP>
   <OMOBJ>
     <OMA>
        <OMS cd="relation" name="geq"/>
        <OMS cd="arith-num" name="sqr-two">
        <OMI>0</OMI>
     </OMA>
   </OMOBJ>
</DefMP>
</CDDefinition>
```

We can see that in this example we have needed two DefMPs to fully specify the square root. Naturally, the proof that these properties are indeed defining, i.e., that there is a unique mathematical object, requires some mathematical insight into the theory of real numbers and in general cannot be expected to be found by a theorem prover totally autonomously.

Another example that has been discussed in OpenMath is the case of the function sin. This can easily be defined to be the real part of the complex exponential function which in turn is the unique zero of the well-known differential equation $\partial f = f$. Again, the proof of unique existence is quite involved but standard, and a CD could point to the relevant literature in the Description.

In the next two paragraphs, we come to two kinds of definition mechanism common in mathematics and computer science where unique existence can be checked automatically in many cases.

3.3 Abstract Data Types; the Natural Numbers

So-called "abstract data types" are inductively defined mathematical structures that include lists, natural numbers, trees, logical formulae and proofs. For instance the natural numbers are defined to be the smallest set of objects that are generated by the constant 0 and the successor function. The well-known Peano Axioms make the underlying mathematical concepts explicit. We could use the DefMP mechanism explicated above directly and arrive at the following definition of the natural numbers:

```
<CDDefinition>
<Name> nat  </Name>
<Name> zero-nat </Name>
<Name> succ-nat  </Name>
<Description> The natural numbers defined as the smallest set objects generated
              by zero-nat and succ-nat
</Description>
<DefMP> zero-nat ∈ nat</DefMP>
<DefMP> ∀ x.x ∈ nat ⇒ succ-nat(x) ∈ nat</DefMP>
<DefMP> ∀ x.zero-nat ≠ succ-nat(x)</DefMP>
<DefMP> ∀ x, y.succ-nat(x) = succ-nat(y) ⇒  x = y</DefMP>
<DefMP> ∀ P.P(zero-nat) ∧ ∀ x.P(x) ⇒  P(succ-nat(x)) ⇒ ∀ x.P(x)</DefMP>
</CDDefinition>
```

The first two DefMPs say that nat is generated by zero-nat and succ-nat, the second two that nat is free in zero-nat and succ-nat, i.e., that every object has a unique representation in these constructors, and the fifth DefMP (the induction axiom) says that nat is indeed the smallest such set. Usually in the field of algebraic specification, this information is compactly represented as an abstract data type definition like

```
nat=ADT(zero-nat:nat,succ-nat:nat->nat)
```

which has the same semantics, since it entails generatedness and freeness axioms for all constructors (zero-nat and succ-nat in this case the first four DefMPs) and the respective induction axiom (the fifth DefMP). The abstract data type for lists would be of the form

```
list=ADT(nil:list,cons:list->el->list)
```

To model this, we could add a new n-ary symbol `adt` to the content dictionary `basic`[3] with the obviously semantics that would allow us to symbolize the definition of the natural numbers as

```
<CDDefinition>
<Name> nat  </Name>
<Name> zero-nat </Name>
<Name> succ-nat  </Name>
<Description> The natural numbers defined as the smallest set objects generated
              by zero-nat and succ-nat
</Description>
<DefMP>
   <OMA>
     <OMS cd="basic" name="adt"/>
     <OMS cd="nat" name="nat"/>
     <OMS cd="nat" name="zero-nat"/>
     <OMS cd="nat" name="succ-nat"/>
   </OMA>
</DefMP>
</CDDefinition>
```

Certainly this would be much more legible only at the cost of introducing a new symbol into `basic`.

3.4 Inductive Definitions; Addition

Recursive functions on sorts introduced in abstract data types can be defined inductively. For instance, addition on the natural numbers can be defined by the following two well-known equations.

$$\forall x.x + 0 = x \qquad \text{and} \qquad \forall x, y.x + s(y) = s(x + y)$$

These equations define a total function, since all cases for the second argument are covered (zero and successor) and if the equations are read as reduction rules from left to right, their application is terminating (the number of applications is bound by the number of occurrences of s in the first argument). Concretely, the definition of addition on the natural numbers would have the following form:

```
<CDDefinition>
<Name> plus  </Name>
<Description>
   Addition on the natural numbers defined by induction on the second argument.
</Description>
<DefMP>∀x.  x + 0 = x  </DefMP>
<DefMP>∀x, y.  x + s(y) = s(x + y)</DefMP>
<FMP>∀x, y.  x + y = y + x</FMP>
</CDDefinition>
```

Note that we did not have to extend the definition mechanism for this special implicit definition. The only difference is that there are deduction systems that can automatically generate the termination orderings necessary for the proof of unique existence for a great number of examples, adding to the tool support in consistency management of OpenMath content dictionaries.

4 Conclusion

We have proposed a modest extension to `CDDefinition` in OpenMath content dictionaries that allows to lay the infrastructure for automated tool-support for consistency management in content

[3]Maybe it would be better to group `adt` and `defines` mentioned above into a new CD `definition-tools`.

dictionaries. The new `DefMP` tag is analogous to the existing `FMP` tag but has semantics that it fulfills a unique existence condition. In most practical cases, this can be verified by existing theorem provers. The ΩMEGA system developed by the second author at the University of the Saarland has an OPENMATH interface and can supply the necessary deduction support.