OpenMath issues arising from Algebra Interactive

Arjeh M. Cohen

May 22, 2004

History of Algebra Interactive

'99 First edition, no OM

'04 Second edition, with OM

Algebra Interactive II, New features

- OM objects
- Mathdox, incorporating namespaces

  c=Core, q=Query, x=XML, cont=Context, macro=Macro
- backengines Magma, GAP, Mathematica, CoCoA
- customization
- examples dependent on user input
- labeled digraph display and manipulation
- context (extended CDs, editor, scoping)

Issues

1. presentation
2. sequences
3. deconstruction
4. application
5. bindings
6. casting

1. Presentation

- attributes: style for display
- α-conversion
- sequences

Presentation

style attribute for display

```
<OMA style="sub">
    <OMS cd="group3" name="symmetric_groupn"/>
    <OMI>3</OMI>
</OMA>
```

renders as Sym_3 but as $\mathrm{Sym}(3)$ without the style attribute.

```
<OMA><OMS cd="fns2" name="apply_to_list"/>
  <OMA><OMS cd="fns2" name="map"/>
    <OMBIND><OMS cd="fns1" name="lambda"/>
      <OMBVAR><OMV name="x"/></OMBVAR>
      <OMA><OMS cd="permutation1" name="cycle"/>
        <OMV name="x"/>
        <OMA><OMS cd="arith1" name="plus"/>
          <OMV name="n"/>
          <OMA><OMS cd="arith1" name="unary_minus"/>
            <OMV name="x"/>
          </OMA>
          <OMI>1</OMI>
        </OMA>
      </OMA>
    </OMBIND>
    <OMA><OMS name="integer_interval" cd="interval1"/>
      <OMI>1</OMI>
      <OMA><OMS name="arith1" cd="divide"/>
        <OMA><OMS name="arith1" cd="minus"/>
          <OMV name="n"/> <OMI>1</OMI>
        </OMA>
        <OMI>2</OMI>
      </OMA>
    </OMA>
```

Presentation, example

```
apply_to_list(permutation,
              map(x -> cycle(x, n-x+1), [1..(n-1)/2])
             )
```

should give $(1, n)(2, n-1)\ldots((n-1)/2, (n+3)/2)$

α conversion gives $(1, n-1+1)\ldots((n-1)/2, n-(n-1)/2+1)$

Presentation, bad hack

```
permutation(sequence(
              cycle(1,2) ,
              "...",
              cycle((n-1)/2,(n+3)/2)
                      )
            )
```

in order to render $(1, n) \ldots ((n-1)/2, (n+3)/2)$

2. Sequences

A sequence is not a list, but is convenient:

- for notational purposes
- for representing the childrens of a construct

Sequences, notational

$x, y \in \mathbb{Z}$ expressible as

```
<OMA><OMS cd="set1" name="in"/>
  <OMA><OMS cd="sequence1" name="sequence"/>
       <OMV name="x"><OMV name="y">
  </OMA>
  <OMS cd="setname1" name="Z"/>
</OMA>
```

Sequences, representing children

The arguments of $f(x, y)$ are x, y, so alternative to $f(x, y)$ is

```
<OMA><OMV  name="f"/>
  <OMA><OMS cd="sequence1" name="sequence"/>
    <OMV name="x"><OMV name="y">
  </OMA>
</OMA>
```

Consider $f(x_1, ..., x_n)$ instead

3. Deconstruction

Proposal: a CD with symbols like `argument`

```
<OMA><OMS cd="deconstr1" name="arg"/>
  <OMA> F
    <OMV name="arg1"/> <OMV name="arg2"/> ...
  </OMA>
  <OMI>i</OMI>
</OMA>
```

for an integer `i` refers to
the i-th argument `<OMV name="arg`i`"/>` of `F`.

If $i = 0$, then `argument(M,i)` stands for `F`.

Deconstruction, bind

For OMBIND, the interpretation of `arg` might be

```
<OMA><OMS cd="deconstr1" name="arg"/>
     <OMBIND> F
              <OMBVAR> v </OMBVAR>
              A
     </OMBIND>
     <OMI>i</OMI>
</OMA>
```

stands for F if $i = 0$, for v if $i = 1$ and for A if $i = 2$.

Deconstruction, error

For OME, the interpretation of `arg` might be

```
<OMA><OMS cd="deconstr1" name="arg"/>
      <OME>
                F     A
      </OME>
      <OMI>i</OMI>
</OMA>
```

stands for F if $i = 0$ and for A if $i = 1$.

Deconstruction, attribution

For OMATTR the interpretation of `arg` might be

```
<OMA><OMS cd="deconstr1" name="arg"/>
     <OMATTR>
             <OMATP> P1 P2 P3 P4 ... </OMATP>
              F
     </OMATTR>
     <OMI>i</OMI>
</OMA>
```

stands for F if $i = 0$ and for Pi if $i > 0$.

Deconstruction, arguments

The CD might also contain a symbol `arguments` which,
when applied to an OM object M,
returns the sequence

`arg(M,0), arg(M,1),...`

Deconstruction, example

For $4/(-6) = -2/3 \in \mathbb{Q}$ compare
`arg(rational(4,-6),1)` which is 4,
with `numerator(rational(4,-6))` which might be -2.

Deconstruction, conclusion

For a symbol with role application
provide symbol names in the same CD for deconstructors
(group1)

4. Application

Which symbols and variables may play the role of application?

- subscripting
- list entry
- permutation action
- polynomial evaluation

Application, subscripting

```
<OMA style="sub"><OMSTR>a</OMSTR>
     <OMI>127</OMI>
</OMA>
```

or

```
<OMA><OMS  cd="indexing" name="indexed_symbol">
     <OMSTR>a</OMSTR>
     <OMI>127</OMI>
</OMA>
```

or, in case of more indices,

```
<OMA><OMS  cd="indexing" name="indexed_symbol">
     <OMV name="x"/>
     <OMI>126</OMI>     <OMI>127</OMI>
</OMA>
```

Application, list entry

```
<OMA>
  <OMA><OMS cd="list1" name="list"/>
    <OMI>3</OMI><OMI>6</OMI><OMI>9</OMI>
  </OMA>
  <OMI>2</OMI>
</OMA>
```

evaluates to 6

5. Bindings

- Compare $\{g(x) \mid f(x) \in A\}$ to $\{x \in B \mid f(x) \in A\}$ and $\{g(x) \mid x \in A\}$
- fns3.mapsto $(x^2 : y^2 : xy) \mapsto xy/(x^2 + y^2)$

6. Casting

- casting to declare expected types
- casting for efficiency

Cast an arithmetic expression A to a polynomial in the ring $R[x, y]$

```
polynomial(R[x,y],A)
```

Casting, efficient data representation

- quotient_ring(R,I), where I = ideal(R,B), or ideal (B,R), or ideal(B)
- polynomial(Ring, term data), where Ring determines the interpretation of term data

Casting, for lists

Rather than

```
list(modmelt(Zmodm(7),2), modmelt(Zmodm(7),5),
     modmelt(Zmodm(7),4), modmelt(Zmodm(7),3))
```

want

```
Flist(Zmodm(7), list(2, 5, 4, 3))
```

which (here) is equivalent to

```
map(x -> modmelt(Zmodm(7),x), list(2, 5, 4, 3))
```

(replace list by matrix or polynomial)

Conclusion

OM is very useful for Algebra Interactive

Conclusion

Interactive is very useful for OM

Thanks

to the organizers

to the audience