OpenMath Content Dictionaries: the Current State

James H. Davenport
Department of Computer Science
University of Bath
BATH BA2 7AY U.K.
J.H.Davenport@bath.ac.uk
CD Editor: OpenMath Thematic Network

September 5, 2001

1 Content Dictionaries

- Contain names, with formal and informal properties.
- Define the semantics of the mathematical object, so `factor` means "the factorization of".
- Type information in an associated Small Type System (STS) file.

Compare

```
<OMS name="mean" cd="s-data1"/>
<OMS name="mean" cd="s-dist1"/>.
```

Both correspond to the MathML symbol `<mean/>`, but have different semantics.

2 Summary of current state

- MathML-compatible (a moving target).
- Much effort to get the semantics absolutely definite (branch cuts etc.).

$$\arctan_{\mathrm{Derive}}(z) = \overline{\arctan_{\mathrm{Maple}}(\overline{z})}.$$

- Some useful extensions, simple proofs.
- Various forms of polynomial and Gröbner base.
- Dimensions and units.

3 MathML-induced Changes

<reln> and <fn> deprecated, so <OMA> now translates more uniformly into <apply>.

Arithmetic Add <arg/>, <real/>, <imaginary/>, <lcm/>, <floor/> and <ceiling/>.

Relations Add <equivalent/>, <approx/> (what semantics?) and <factorof/>.

Set Theory Add <card/>, corresponding to OpenMath's

```
<OMS name="size" cd="set1"/>,
```

and <cartesianproduct/> (spelled with an "_" in OpenMath).

Elementary Functions MathML added <arccot/>, <arcsec/> and <arccsc/>, as well as the hyperbolic equivalents.

Set Symbols MathML added <integers/>, <reals/>, <rationals/>, <naturalnumbers/>, <complexes/> and <primes/>. In OpenMath:

```
<OMS name="R" cd="setname1"/>.
```

Constants MathML added <exponentiale/>, <imaginaryi/>, <notanumber/>, <true/>, <false/>, <pi/>, <eulergamma/> and <infinity/>. In OpenMath they are

```
<OMS name="e" cd="nums1"/>.
```

functions MathML added domain, codomain and image. It also introduced domainofapplication, as in $\int_C f$:

```
<apply>
  <int/>
  <domainofapplication>
    <ci> C </ci>
  </domainofapplication>
  <ci> f </ci>
</apply>
```

This particular example was already catered for in OpenMath, as in

```
<OMOBJ>
  <OMA>
    <OMS name="defint" cd="calculus1"/>
```

```
    <OMV name="C"/>
    <OMV name="f"/>
  </OMA>
</OMOBJ>
```

Piecewise MathML added three symbols for piece-wise definitions of functions: `piecewise`, `piece` and `otherwise`. These were encoded into OpenMath as elements of the new `piece1` CD.

Vectors MathML added the symbols `divergence`, `grad`, `curl` and `laplacian`. Similarly, `vectorproduct`, `scalarproduct` and `outerproduct`.

4 Extensions to the MathML CDs

Arithmetic The `arith2` CD contains two symbols: `inverse` intended to represent the additive or multiplicative inverse of an element, and `times`, an explicitly commutative version of the `times` symbol in the `arith1` CD.

The `fns2` CD contains three symbols.

`apply_to_list` which represents the application of an n-ary function to all the elements of a list.

`kernel` which represents the usual algebraic object.

`right_compose` (logically redundant).

Lists The `list2` CD contains `cons`, `first` and `rest`. I **propose** `nil`, `append` and `reverse`.

Set Names The `setname2` CD contains several others: `A` (the algebraic numbers), `Boolean`, `GFp`, `GFpn`, `H` (the Hamiltonian, or hyper-complex, numbers), `QuotientField` (which takes an integral domain as argument) and `Zm`.

Linear Algebra MathML, and OpenMath's `linalg2` CD, define matrices as built up from rows. The `linalg3` CD defines a column-oriented view of matrices, via `matrix`, `matrixcolumn` and `vector`.

The `linalg4` CD contains some additional linear algebra symbols representing abstract concepts: `characteristic_eqn`, `columncount`, `eigenvalue` (this takes two arguments: the first should be the matrix, the second should be an index to specify the eigenvalue), `eigenvector`, `rank`, `rowcount` and `size`.

The `linalg5` CD contains various symbols for defining matrices of special shapes. They are: `anti-Hermitian`, `banded`, `constant`, `diagonal_matrix`, `Hermitian`, `identity`, `lower-Hessenberg`, `lower-triangular`, `scalar`, skew-symmetric, symmetric, tridiagonal, upper-Hessenberg, upper-triangular and zero.

5 Polynomials

There are (currently) 5 CDs.

poly An abstract view of polynomials, also operations like conversion

polyd A distributed view of polynomials, also with orderings and Gröbner base concepts

polyr A recursive view of polynomials

polyslp A straight-line program view

polysts Types for STS to work correctly for the above CDs

5.1 The poly CD

The poly CD supports generic views of polynomials.

convert This takes a polynomial in one polynomial ring, and the specification of a second polynomial ring, and expresses the polynomial represented in that second ring.

degree The total degree function.

degree_wrt The degree with respect to a specific variable (the second argument to the symbol).

expand This symbol represents the conversion of a factored or squarefreed form into an expanded polynomial over the same ring, so that, for example, factored(recursive) → recursive.

factor This is a call for a factorisation. The result should be an expression built with factored.

factored The constructor for a factorization. Its arguments are formal powers where the polynomials are supposed to be irreducible (except possibly for a content from the ground ring) and relatively prime.

gcd This is an n-ary symbol, representing the greatest common divisor of its polynomial arguments.

lcm This is an n-ary symbol, representing the least common multiple of its polynomial arguments.

power Takes a polynomial and a (non-negative) integer and produces a formal power. `power` from `arith1` would suggest the expanded form.

resultant This takes two polynomials and a variable as arguments, and represents the resultant of the two polynomials with respect to that variable.

squarefree This is a call for a square free decomposition.

squarefreed As for `factored` above.

5.2 The polyr CD

The `polyr` CD deals with polynomials described in recursive format, so that the polynomial $2 * y^3 * z^5 + x + 1$ in $\mathbf{Z}[z][y][x]$ can be conceptually encoded as

```
poly_r_rep(x,
    term(1,1),
    term(0,poly_r_rep(y,
              term(3,poly_r_rep(z,
                        term(5,2))),
              term(0,1))))
```

poly_r_rep This takes a variable and then any number of `term` arguments in decreasing degree order, and constructs a polynomial in that variable with those terms.

term Takes two arguments: a degree (from $\mathbf{N}$) and a coefficient, and makes a term.

polynomial_ring_r This constructs the data type of a (recursive) polynomial ring, e.g. $\mathbf{Z}[x, y, z]$ (implemented as $\mathbf{Z}[z][y][x]$) would be:

```
<OMOBJ>
  <OMA>
    <OMS name="polynomial_ring_r" cd="polyr"/>
    <OMS name="Z" cd="setname1"/>
    <OMV name="x"/>
    <OMV name="y"/>
    <OMV name="z"/>
  </OMA>
</OMOBJ>
```

As can be seen, the first argument is the coefficient ring (which could itself be a polynomial domain) and the rest are variables.

polynomial_r This constructs a polynomial in a specific ring: the first argument is a `polynomial_ring_r` and the second is a `poly_r_rep` in that ring.

5.3 The polyd CD

The `polyd` CD deals with polynomials described in distributed format, so that the polynomial $x^2y^6 + 3y^5$ can be encoded (including the type of the ring to which it belongs) as

```
DMP(poly_ring_d(Z, 2),
    SDMP(term(1, 2, 6), term(3, 0, 5)))
```

DMP This symbol takes two arguments: a distributed polynomial ring (built with the `poly_ring_d` symbol) and a polynomial (built with `SDMP`) and returns the polynomial in that ring.

DMPL As DMP, except that it takes an arbitrary number of `SDMP`s, and returns a list of polynomials (all in the same ring).

groebner This symbol represents the construction of a Gröbner basis: the first argument is an ordering, and the second a list of polynomials (i.e. a `DMPL`). If sent to a computational engine, the result should be a `groebner_basis` object.

groebner_basis This is the constructor for an auto-reduced Gröbner basis. The first argument to this symbol is an ordering, and the second is a `DPML` representing the basis.

plus This takes a `DMPL` as its (single) argument, and returns a `DMP` (in the same ring) representing the sum of the polynomials in the `DMPL`.

poly_ring_d This constructs a distributed polynomial ring (i.e. an object of type `polynomial_ring`). Its two arguments are the coefficient ring and the *number* of variables. Hence these are essentially polynomials in anonymous variables. *Is this right?*

power Takes two arguments, a `DMP` and a non-negative integer, and should return a `DMP` representing the appropriate power of the input `DMP`.

reduce The represents the reduction of the first argument, a polynomial (i.e. a `DMP`) with respect to the second argument, a Gröbner basis (i.e. a `groebner_basis` object). The result, if this is passed to a computational agent, should be a `DMP`.

SDMP The constructor for multivariate polynomials without any indication of variables or domain for the coefficients. Its arguments are "monomial"s, built with the `term` constructor. No monomials should differ only by the coefficient. SDMPs can be attributed with the "ordering" symbol to indicate a particular ordering of its monomials.

`term` This symbol takes $n+1$ arguments (where n is the number of variables in the relevant `poly_ring_d`): the first is the coefficient, and the rest are non-negative integers representing the exponents of the various variables.

`times` This takes a `DMPL` as its (single) argument, and returns a `DMP` (in the same ring) representing the product of the polynomials in the `DMPL`.

`ordering` This specifies how the monomials are ordered. Thus the polynomial $x^2y^6 + 3y^5$ can be more fully encoded as follows

```
<OMOBJ>
 <OMATTR>
  <OMATP>
   <OMS name="ordering" cd="polyd"/>
   <OMS name="graded_lexicographic" cd="polyd"/>
  </OMATP>
  <OMA>
   <OMS name="DMP" cd="polyd"/>
   <OMA>
    <OMS name="poly_ring_d" cd="polyd"/>
    <OMS name="Z" cd="setname1"/>
    <OMI> 2 </OMI>
   </OMA>
   <OMA>
    <OMS name="SDMP" cd="polyd"/>
...
```

`elimination` One of the orderings. It takes three arguments: the first is a number k of variables, the second is an ordering to apply to the first k variables, and the third is an ordering to apply as a tie-breaker to the rest of the variables.

```
<OMA>
  <OMS name="elimination" cd="polyd"/>
  <OMI> 1 </OMI>
  <OMS name="lexicographic" cd="polyd"/>
  <OMS name="graded_reverse_lexicographic" cd="polyd"/>
</OMA>
```

`graded_lexicographic`

`graded_reverse_lexicographic`

lexicographic

reverse_lexicographic

5.4 The polyslp CD

The polyslp CD deals with polynomials described in straight-line program format so that x^2y^2 can be represented as:

```
<OMOBJ>
  <OMA>
    <OMS cd="polyslp" name="polynomial_SLP"/>
    <OMA>
      <OMS cd="polyslp" name="poly_ring_SLP"/>
      <OMS cd="setname1" name="Z"/>
      <OMV name="x"/>
      <OMV name="y"/>
    </OMA>
    <OMA>
      <OMS cd="polyslp" name="prog_body"/>
      <OMA>
        <OMS cd="polyslp" name="inp_node"/>
        <OMV name="x"/>
      </OMA>
      <OMA>
        <OMS cd="polyslp" name="inp_node"/>
        <OMV name="y"/>
      </OMA>
      <OMA>
        <OMS cd="polyslp" name="op_node"/>
        <OMS cd="opnode" name="times"/>
        <OMI> 1 </OMI>
        <OMI> 1 </OMI>
      </OMA>
      <OMA>
        <OMS cd="polyslp" name="op_node"/>
        <OMS cd="opnode" name="times"/>
        <OMI> 2 </OMI>
        <OMI> 2 </OMI>
      </OMA>
      <OMA>
        <OMS cd="opnode" name="return"/>
        <OMA>
          <OMS cd="polyslp" name="op_node"/>
          <OMS cd="opnode" name="times"/>
          <OMI> 3 </OMI>
          <OMI> 4 </OMI>
        </OMA>
      </OMA>
    </OMA>
  </OMA>
</OMOBJ>
```

const_node This takes one argument, which is a value in the coefficient ring of the poly_ring_SLP.

depth This unary symbol represents the maximum depth of an SLP, i.e. the longest path from any node to a return node.

inp_node This takes one argument, which is the name of one of the variables in the poly_ring_SLP.

left_ref Takes as argument a node of an slp. Returns the value of the left hand pointer of the node.

length This unary symbol represents the length (number of arguments to prog_body) in an SLP.

monte_carlo_eq This represents a Monte-Carlo equality test, it takes three arguments, the first two are slps representing polynomials, the third argument is the maximum probability of incorrectness that is required of the equality test.

node_selector Takes an slp as the first argument, the second argument is the position of the required node. Returns the node of the slp at this position.

op_node This constructor takes three arguments. The first argument is a symbol from the `opnode` CD, meant to specify whether the node is a plus, minus, times or divide node, the second and third arguments are integers, which are the numbers of the lines which are the arguments of the operation.

poly_ring_SLP The constructor of the polynomial ring. The first argument is a ring, (the ring of the coefficients), the rest are the variables, in any order.

polynomial_SLP This actually builds a polynomial in a given SLP ring (the first argument). The second argument has to be a `prog_body`.

prog_body This takes n arguments, which are the instructions of a straight-line program. In particular they must be of types `const_node`, `inp_node` or `op_node`, possibly wrapped inside the `return` symbol from the `opnode` CD.

quotient A quotient function for polynomials represented by SLPs. It is a requirement that this is an exact division.

return_node Takes an slp as the argument, and returns the return node of the slp.

right_ref Takes as argument a node of an slp. Returns the value of the right hand pointer of the node.

slp_degree A unary symbol taking an SLP as argument and representing the apparent multiplicative degree of the SLP, without performing any cancellation.

The related `opnode` CD contains the symbols for the four binary arithmetic operations (`divide`, `minus`, `plus` and `times`), as well as the unary `return` symbol.

6 Dimensions and Units

There have been several well-publicised problems with the misunderstanding of units. However, before units can be formalised, dimensions have to be.

6.1 Dimensions

The CD `dimensions1` contains some fundamental and derived dimensions. The fundamental ones are `charge`, `length`, `mass`, `temperature` and `time`. The derived ones are `area`, `volume`, `speed`, `velocity`, `acceleration`, `force`, `pressure`, `current` and `voltage`. Formal Mathematical Properties link the derived ones to the fundamental ones.

6.2 Units

There are two CDs currently that capture units: `units_metric1` and `units_imperial1`. The definitions in these are fairly obvious. Though this has not yet been done, the conversion of imperial units to metric (for the fundamental dimensions) should be encoded as Formal Mathematical Properties, so that conversions could then be deduced for the other units by means of the Formal Mathematical Properties in the `dimensions1` CD.

7 Proofs

It is often said that OpenMath cannot handle proofs.

```
<OMA>
  <OMS cd="logic3" name="complete_prop_theorem"/>
  <OMA>
    <OMS cd="logic1" name="implies"/>
    <OMV name="A"/>
    <OMV name="A"/>
  </OMA>
  <OMA>
    <OMS cd="logic3" name="proof"/>
    <OMA>
      <OMS cd="list1" name="list"/>
      .....
    </OMA>
  </OMA>
</OMA>
```

Individual lines of the proof would look like

```
<OMA>
  <OMS cd="logic3" name="axiom_instance"/>
```

$$((a \Rightarrow ((a \Rightarrow a) \Rightarrow a)) \Rightarrow ((a \Rightarrow (a \Rightarrow a)) \Rightarrow (a \Rightarrow a)))$$

$$((a \Rightarrow (b \Rightarrow c)) \Rightarrow ((a \Rightarrow b) \Rightarrow (a \Rightarrow c)))$$

```
</OMA>
```

or lines using

```
<OMS cd="logic3" name="ModusPonens"/>
```

Question. Axiom 4 of predicate calculus is normally written as

$$(\forall x A(x) \Rightarrow A(t)).$$

This seems to call for a "substitution" symbol. Should this be just in `logic3`, or be more general? If it is to be more general, it should probably be of the form "multiple parallel substitution", as in

$$(\mathit{Expression}, \mathit{List} \quad \mathit{Symbol}, \mathit{List} \quad \mathit{Expression}) \rightarrow \mathit{Expression},$$

since this effect is hard to achieve in other ways.
The axiom has caveats (t must be free for x in A) in predicate logic, but OpenMath should not attempt to stipulate them. Should it have tests for them, e.g.

$$\texttt{free}?(A, x, t)?$$

8 Immediate Suggestions

- Augment `list2`.
- Let `polyd` name its variables (attribute ?).
- Some meaning to approx (maybe attributes ?).
- More work on the units CDs, especially FMPs.
- Think about formal proofs.

9 Future work on CDs

Special Functions Some work has been done. As with elementary functions, there is a great need for precision over branch cuts etc.

```
<OMA>
  <OMA>
    <OMS name="J" cd="Bessel"/>
    <OMV name="nu"/>
  </OMA>
  <OMV name="z"/>
</OMA>
```

or

```
<OMA>
  <OMS name="BesselJ" cd="specfun1"/>
  <OMV name="nu"/>
  <OMV name="z"/>
</OMA>
```

Abstract Algebra Many more CDs need to be written and/or formalised. Problems of consistency:

Degree S_{12}, M_{12} are permutation groups acting on 12 symbols;

Size F_{20} is a permutation group of size 20, normally acting on 5 elements;

?? D_{12}?

Also need to deal with ideals etc., rather than just lists of polynomials (different lists can represent the same ideal).

Algorithms A CD to describe algorithmic concepts would be useful, partly from the point of view of the wider publication-related aspects of OpenMath, and partly for use in concepts such as symbolic differentiation; i.e. differentiating an algorithm.

Logics While basic classical logic (propositional, predicate) is catered for, there is nothing on other forms of logic (intuitionistic etc.). Different concepts of equality also need to be handled.

10 Yesterday's Decisions

- Augment `list2` with (at least) nil, reverse, append.

* Agreed.

- Attributes to approx: abserr, relerr and O.

* Agreed for abserr and relerr; O referred to the asymptotics CD (in draft).

- More work on the units CDs, especially FMPs.
  (get a draft to MathML for MathML3)

- Special functions:

  - curried where sensible;
  - `<OMS name="J" cd="Bessel"/>`.

  Input from UWO and INRIA.

* Agreed — check with NIST.

- Publish a draft of `logic3`.

* Agreed.

11 Today's Decisions

- Let `polyd` name its variables:

  Either an attribute `variable_names`;

  Or a second constructor which named the variables.

* **Change** the `poly_ring_d` constructor to require variable names: cross-check with CoCoa phrasebooks.

- `substitute` command:

  Either in `logic3` (one symbol);

  Or in a `subst1` CD, as a multiple-in-parallel operator.

* Use lambda-abstraction to express substitution.

- Do we want `is_groebner` as well as (instead of?) `groebner_basis`?

* Agreed (as well as).

12 Near Future Decisions

- Does Arjeh really need another polynomial CD?
- Abstract algebra: James cooperate with Arjeh.
- Algorithms CD: James cooperate with Arjeh.

When does James visit Arjeh (or v.v.)?