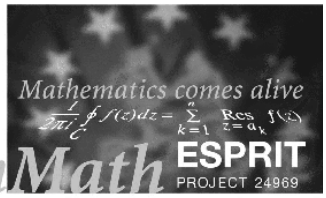

Conversion between MathML and OpenMath

D. Carlisle, J. Davenport, M.Dewar, N.Hur and W. Naylor[1]

Bath/NAG

Abstract

OpenMath and MathML (Content) are both semantic representations of mathematics. This document shows how all of MathML (Content) can be translated into OpenMath.

Introduction

This document begins with a table detailing the transformation required to convert between MathML and OpenMath, using the MathML CD Group. We then give various illustrations of the more complex conversions.

The first column gives the MathML element and attribute combination, the second column gives the equivalent OpenMath construct. Additional notes and comments are provided where appropriate.

MathML – OpenMath

`cn` or `cn type="real"`	If followed by an explicit floating point number, then this would normally be represented by an `OMF` (if expressible in terms of IEEE floats), or possibly via the `bigfloat1` CD. The use of symbolic constants under `cn` such as `&pi` is now deprecated and not supported by OpenMath.
`cn type="e-notation"` (added 18.4.2000 to MathML: syntax is mantissa, as in real, `<sep/>`, decimal exponent)	As above
`cd type="integer"`	Is supported by `OMI` in base 10 or 16. Integers expressed in other bases may be converted to a `<OMS name="based_integer" cd="nums1"/>` symbol applied to the digits in an `<OMSTR>`. If they are converted into `OMI`s, then the base information is lost.
`cn type="rational"`	This is represented by `<OMS name="rational" cd="nums1"/>`. Note that MathML allows 1/0 and OpenMath does not (through this constructor). It appears that MathML allows decimal points inside `cn type="rational"`, but this is a curiosity unsupported by OpenMath in this form.
`cn type="complex-cartesian"`	This equates to `<OMS name="complex_cartesian" cd="complex1"/>`
`cn type="complex-polar"`[2]	This equates to `<OMS name="complex_polar" cd="complex1"/>`

[2]This seems to be missing from appendix C of the Proposed Recommendation, but is clearly present in Chapter 4.4.

`ci`	This normally corresponds to an `OMV`. However, `ci type="XXX"` corresponds to an attribution from the `mathmltypes` CD on the `OMV`.
`csymbol`	If the URL points to an OpenMath CD, then this refers to the symbol of that name in the CD.
`apply`	Corresponds to `OMA`.
`reln` (now deprecated in MathML)	Corresponds to `OMA`.
`fn` (now deprecated in MathML)	Is ignored in OpenMath.
`interval`	According to the type attribute (which defaults to `closed`) corresponds to a symbol in the `interval1` CD.
`inverse`	Corresponds, with all its (lack of) semantics, to `<OMS name="inverse" cd="fns1"/>`.
`sep`	Handled by its parent constructor.
`condition` inside `int`	This becomes the first argument to `<OMS name="defint" cd="calculus1"/>`.
`int` with a general range	This may be performed by using `<OMS name="defint" cd="calculus1"/>` where the range argument is an `OMV`. See illustration 2.
`condition` inside `sum` and `product`	This becomes the first argument to `<OMS name="sum" cd="arith1"/>`.
`condition` inside `limit`	This should be a `tendsto` construct which gives the arguments of the OpenMath construction. See illustration 1.
`condition` inside `min` and `max`	Becomes a `<OMS name="suchthat" cd="set1"/>`.

`condition` inside `forall` and `exists`	The condition is moved inside the body (normally as `implies` for `forall`, but `and` for `exists`): see illustration 3.
`condition` inside `set`	Becomes a `<OMS name="suchthat" cd="set1"/>`.
`condition` inside `list`	Becomes a `<OMS name="suchthat" cd="list1"/>`.
`declare type="XXX"`	This can be supported by replacing all occurrences of the name by the name attributed by the type from the `mathmltypes` CD. Other instances of `declare` are not supported in OpenMath.
`lambda`	Corresponds to `OMBIND` with a first child of `<OMS name="lambda" cd="fns1"/>`. The `bvar` elements under the MathML `lambda` become children of the OpenMath `OMBIND`.
`compose`	Corresponds to `<OMS name="left_compose" cd="fns1"/>`.
`ident`	Corresponds to `<OMS name="identity" cd="fns1"/>`.
`domain`	Corresponds to `<OMS name="domain" cd="fns1"/>`.
`domainofapplication`	Corresponds to `<OMS name="domainofapplication" cd="fns1"/>`. However, when used inside `int` to give a general range of integration, it becomes an argument of `defint`, without any `domainofapplication` .
`codomain`	Corresponds to `<OMS name="image" cd="fns1"/>`.
`image`	Corresponds to `<OMS name="range" cd="fns1"/>`.

`quotient`	Corresponds to `<OMS name="quotient" cd="integer1"/>`. The `type` qualifier, if not `integer`, is not supported in OpenMath.
`factorial`	Corresponds to `<OMS name="factorial" cd="integer1"/>`. The `type` qualifier, if not `integer`, is not supported in OpenMath.
`divide`	Corresponds to `<OMS name="divide" cd="arith1"/>`. The `type` qualifier is ignored.
`max` and `min`	If these MathML constructors are applied to something other than a `condition`, this becomes an OpenMath application of `<OMS name="max" cd="minmax1"/>` to a set built with `<OMS name="set" cd="set1"/>`. If applied to a `condition`, this becomes an application of `<OMS name="max" cd="minmax1"/>` to a set built with `<OMS name="suchthat" cd="set1"/>`.
`minus`	Corresponds to `<OMS name="minus" cd="arith1"/>` or `<OMS name="unary_minus" cd="arith1"/>`, depending on the arity. However, applications to objects of type `set` or `multiset`, as envisaged in section C.2.3.6 of the MathML2 standard, should be translated into `<OMS name="setdiff" cd="set1"/>` (or `cd="multiset1"/>`).
`plus`	Corresponds to `<OMS name="plus" cd="arith1"/>`. However, `<apply><plus/></apply>` corresponds to `<OMS name="zero" cd="alg1"/>`. Furthermore, applications to objects of type `set` or `multiset`, as envisaged in section C.2.3.7 of the MathML2 standard, should be translated into `<OMS name="union" cd="set1"/>` (or `cd="multiset1"/>`).
`power`	This corresponds to `<OMS name="power" cd="arith1"/>`.

`real` and `imaginary`	These correspond to `<OMS name="real" cd="complex1"/>` (and `imaginary`).
`rem`	Corresponds to `<OMS name="remainder" cd="integer1"/>`. The `type` qualifier, if not `integer`, is not supported in OpenMath.
`times`	This corresponds to `<OMS name="times" cd="arith1"/>`.
`root`	This corresponds to `<OMS name="root" cd="arith1"/>`. However, note that the OpenMath semantics are those of `type="principal_branch"` (misspelled as `type="principle_branch"` in Appendix C of the Proposed Recommendation for MathML2), and the default type of `real` is not directly supported in OpenMath.
`gcd`	This corresponds to `<OMS name="gcd" cd="arith1"/>`. However, `<apply><gcd/></apply>` corresponds to `<OMS name="one" cd="alg1"/>` (logically, it should be 0, but this is the MathML statement, in C.2.3.16).
`and`	This corresponds to `<OMS name="and" cd="logic1"/>`.
`or`	This corresponds to `<OMS name="or" cd="logic1"/>`.
`xor`	This corresponds to `<OMS name="xor" cd="logic1"/>`. MathML has now adopted the OpenMath semantics for the non-binary case.
`not`	This corresponds to `<OMS name="not" cd="logic1"/>`.
`implies`	This corresponds to `<OMS name="implies" cd="logic1"/>`.

`forall`	Corresponds to `OMBIND` with a first child of `<OMS name="forall" cd="quant1"/>`. The `bvar` elements under the MathML `forall` become children of the OpenMath `OMBIND`. Any `condition` has to be moved inside the body, as in illustration 3.
`exists`	Corresponds to `OMBIND` with a first child of `<OMS name="exists" cd="quant1"/>`. The `bvar` elements under the MathML `exists` become children of the OpenMath `OMBIND`. Any `condition` has to be moved inside the body, as in illustration 3.
`abs`	This corresponds to `<OMS name="abs" cd="arith1"/>`.
`conjugate`	This corresponds to `<OMS name="conjugate" cd="complex1"/>`.
`arg`	This corresponds to `<OMS name="argument" cd="complex1"/>`.
`lcm`	This corresponds to `<OMS name="lcm" cd="arith1"/>`. However, `<apply><lcm/></apply>` corresponds to `<OMS name="zero" cd="alg1"/>` (logically, it should be 1, but this is the MathML statement, in C.2.3.25).
`floor`	This corresponds to `<OMS name="floor" cd="rounding1"/>`.
`ceiling`	This corresponds to `<OMS name="ceiling" cd="rounding1"/>`.
`eq`	This corresponds to `<OMS name="eq" cd="relations1"/>`. However, the MathML construct is n-ary, but the OpenMath construct is binary. See illustration 4.
`neq`	This corresponds to `<OMS name="neq" cd="relations1"/>`.

`gt`, `lt`, `geq` and `leq`.	These correspond to `<OMS name="gt" cd="relations1"/>` etc. However, the MathML construct is n-ary, but the OpenMath construct is binary. See illustration 5.
`equivalent`[3]	This corresponds to `<OMS name="equivalent" cd="logic1"/>`.
`approx`	This corresponds to `<OMS name="approx" cd="relation1"/>`. However, the MathML construct is n-ary, but the OpenMath construct is binary. See illustration 4.
`factorof`	This corresponds to `<OMS name="factorof" cd="integer1"/>`.
`int`	This corresponds either to `<OMS name="int" cd="calculus1"/>` or `<OMS name="defint" cd="calculus1"/>`. See illustration 6.
`diff`	This corresponds to `<OMS name="diff" cd="calculus1"/>`. See illustration 7.
`partialdiff`	This corresponds to `<OMS name="partialdiff" cd="calculus1"/>`. See illustration 8.
`lowlimit`, `uplimit`, `bvar` and `degree` are all handled by their parents.	
`divergence`	This corresponds to `<OMS name="divergence" cd="veccalc1"/>`.
`gradient`	This corresponds to `<OMS name="grad" cd="veccalc1"/>`.

[3] Chapter 4 and Appendix C of the Proposed Recommendation differ here: we follow Chapter 4.

`curl`	This corresponds to `<OMS name="curl" cd="veccalc1"/>`. There is not a direct OpenMath correspondence for the MathML form that takes three variables: a λ-expression has to be built.
`laplacian`	This corresponds to `<OMS name="Laplacian" cd="veccalc1"/>`.
`set`	The construct with n explicit arguments corresponds to `<OMS name="set" cd="set1"/>`, or `<OMS name="multiset" cd="multiset1"/>` if `type=multiset` is specified. The form with a `condition` translates to a use of `<OMS name="suchthat" cd="set1"/>`. See illustration 9. There is no OpenMath equivalent if `type=multiset` is specified with a `condition`.
`list`	The construct with n explicit arguments corresponds to `<OMS name="list" cd="list1"/>`. The form with a `condition` translates to a use of `<OMS name="make_list" cd="list1"/>`. See illustration 10.
`union`	This corresponds to `<OMS name="union" cd="set1"/>`, or `<OMS name="union" cd="multiset1"/>`. In the absence of a `definitionURL`, it defaults to `set1`.
`intersect`	This corresponds to `<OMS name="intersect" cd="set1"/>`, or `<OMS name="intersect" cd="multiset1"/>`. In the absence of a `definitionURL`, it defaults to `set1`.
`notin`, `in`, `subset`, `prsubset`, `notsubset` and `notprsubset`	These correspond to `<OMS name="notin" cd="set1"/>`, or `<OMS name="notin" cd="multiset1"/>` etc. In the absence of a `definitionURL`, these default to `set1`.

`setdiff`	This corresponds to `<OMS name="setdiff" cd="set1"/>`, or `<OMS name="setdiff" cd="multiset1"/>`. In the absence of a `definitionURL`, it defaults to `set1`.
`card`	This corresponds to `<OMS name="size" cd="set1"/>`, or `<OMS name="size" cd="multiset1"/>`. In the absence of a `definitionURL`, it defaults to `set1`.
`cartesianproduct`	This corresponds to `<OMS name="cartesian_product" cd="set1"/>`, or `<OMS name="cartesian_product" cd="multiset1"/>`. In the absence of a `definitionURL`, it defaults to `set1`.
`sum` and `product`	These correspond to `<OMS name="sum" cd="arith1"/>` etc. The `uplimit` etc. correspond to the first argument of the OpenMath construction.
`limit`	This corresponds to `<OMS name="limit" cd="limit1"/>`. The MathML `bvar` corresponds to the variable in the λ-expression that is the third argument of the OpenMath construction. The `lowlimit` corresponds to the first argument of the OpenMath limit (which is otherwise extracted from the `condition`), and the second argument defaults to `<OMS name="both_sides" cd="limit1"/>`, if not specified by the `type` attribute of the MathML `tendsto`. See illustration 1.
`tendsto`	This is handled by the parent `limit`.
`exp`, `ln`	These correspond to `<OMS name="exp" cd="transc1"/>` etc.

`log`	The two-argument form corresponds to `<OMS name="log" cd="transc1"/>`. The one-argument form has to have an explicit base of 10 supplied (the first argument of the OpenMath form).
`logbase`	Subsumed within `log`.
`cos` etc. (24 of them)	These correspond to the functions in the `transc1` CD.
`sdev`, `variance`, `median`, `mode` and `mean`	These correspond to either `<OMS name="sdev" cd="s-data1"/>` etc. or `<OMS name="sdev" cd="s-dist1"/>` etc., depending on whether the arguments are explicit data or a distribution. The unary case in MathML is that of a distribution.
`moment`	The definition of `moment` in MathML is undergoing debate. The (now deprecated) `degree` qualifier in MathML becomes the first argument of the OpenMath constructor `<OMS name="moment" cd="s-dist1"/>` (and possibly `<OMS name="moment" cd="s-data1"/>`), the second argument specifies the point about which the moment is to be taken.
`momentabout`	Subsumed in `moment`.
`vector`	This corresponds to `<OMS name="vector" cd="linalg3"/>` if the type is `column` (the default), otherwise to `<OMS name="vector" cd="linalg2"/>`.
`matrix` and `matrixrow`	These correspond to `<OMS name="matrix" cd="linalg2"/>` etc.
`determinant` and `transpose`	Correspond to `<OMS name="determinant" cd="linalg1"/>` etc.

`selector`	Corresponds to `<OMS name="vector_selector" cd="linalg1"/>` or `<OMS name="matrix_selector" cd="linalg1"/>`, depending on the MathML arity.
`vectorproduct`, `scalarproduct` and `outerproduct`	These correspond to `<OMS name="vectorproduct" cd="linalg1"/>` etc.
`integers`	This corresponds to `<OMS name="Z" cd="setname1"/>`.
`reals`	This corresponds to `<OMS name="R" cd="setname1"/>`.
`rationals`	This corresponds to `<OMS name="Q" cd="setname1"/>`.
`naturalnumbers`	This corresponds to `<OMS name="N" cd="setname1"/>`.
`complexes`	This corresponds to `<OMS name="C" cd="setname1"/>`.
`primes`	This corresponds to `<OMS name="P" cd="setname1"/>`.
`exponentiale`	This corresponds to `<OMS name="e" cd="nums1"/>`.
`imaginaryi`	This corresponds to `<OMS name="i" cd="nums1"/>`.
`notanumber`	This corresponds to `<OMS name="NaN" cd="nums1"/>`.
`true`	This corresponds to `<OMS name="true" cd="logic1"/>`.
`false`	This corresponds to `<OMS name="false" cd="logic1"/>`.

`emptyset`	This corresponds to `<OMS name="emptyset" cd="set1"/>`, or possibly `<OMS name="emptyset" cd="multiset1"/>` for the multiset form.
`pi`	This corresponds to `<OMS name="pi" cd="nums1"/>`.
`eulergamma`	This corresponds to `<OMS name="gamma" cd="nums1"/>`.
`infinity`	This corresponds to `<OMS name="infinity" cd="nums1"/>`. The attribute `type=complex` is handled by attributing from the `mathmltypes` CD.

Note that OpenMath does not support MathML's `definitionURL` unless the URL points to an OpenMath CD.

The MathML version:

```
<apply>
  <eq/>
  <apply>
    <limit/>
    <bvar> <ci> x </ci> </bvar>
    <condition>
      <apply>
        <tendsto type="above"/>
        <ci> x </ci> <cn> 0 </cn>
      </apply>
    </condition>
    <apply>
      <divide/> <cn> 1 </cn> <ci> x </ci>
    </apply>
  </apply>
  <infinity/>
</apply>
```

The OpenMath version:

```
<OMOBJ>
  <OMA>
    <OMS cd="relation1" name="eq"/>
    <OMA>
      <OMS cd="limit1" name="limit"/>
      <OMI> 0 </OMI>
      <OMS cd="limit1" name="above"/>
      <OMBIND>
        <OMS cd="funs1" name="lambda"/>
        <OMBVAR> <OMV name="x"/> </OMBVAR>
        <OMA>
          <OMS cd="arith1" name="divide"/>
          <OMI> 1 </OMI>
          <OMV name="x"/>
        </OMA>
      </OMBIND>
    </OMA>
    <OMS cd="nums1" name="infinity"/>
  </OMA>
</OMOBJ>
```

An example of a limit construct to illustrate the condition *tends to from above.* (1)

The MathML version:

```
<apply>
  <int/>
  <bvar>
     <ci> x </ci>
  </bvar>
  <lowlimit>
     <bvar>
       <ci> a </ci>
     </bvar>
  </lowlimit>
  <uplimit>
     <bvar>
       <ci> b </ci>
     </bvar>
  </uplimit>
  <apply>
     <ci> f </ci>
     <ci> x </ci>
  </apply>
</apply>
```

The OpenMath version:

```
<OMA>
  <OMS cd="calculus1" name="defint"/>
  <OMA>
    <OMS cd="interval1" name="interval"/>
    <OMV name="a"/>
    <OMV name="b"/>
  </OMA>
  <OMBIND>
    <OMS cd="fns1" name="lambda"/>
    <OMBVAR>
      <OMV name="x"/>
    </OMBVAR>
    <OMA>
      <OMV name="f"/>
      <OMV name="x"/>
    </OMA>
  </OMBIND>
</OMA>
```

However we could specify effectively the same integral without labelling the upper and lower bounds of the range of integration in the following manner:

```
<OMA>
  <OMS cd="calculus1" name="defint"/>
  <OMV name="R"/>
  <OMBIND>
    <OMS cd="fns1" name="lambda"/>
    <OMBVAR>
      <OMV name="x"/>
    </OMBVAR>
    <OMA>
      <OMV name="f"/>
      <OMV name="x"/>
    </OMA>
  </OMBIND>
</OMA>
```

An example to show the specification of a general range of integration. (2)

The MathML version:

```
<apply>
  <forall/>
  <bvar> <ci> x </ci> </bvar>
  <condition>
    <apply>
      <in/> <ci> x </ci>
      <naturalnumbers/>
    </apply>
  </condition>
  <apply>
    <in/>
    <apply>
      <plus/>
      <ci> x </ci> <cn> 1 </cn>
    </apply>
    <naturalnumbers/>
  </apply>
</apply>
```

The OpenMath version:

```
<OMOBJ>
  <OMBIND>
    <OMS cd="quant1" name="forall"/>
    <OMBVAR>
      <OMV name="x"/>
    </OMBVAR>
    <OMA>
      <OMS cd="logic1" name="implies"/>
      <OMA>
        <OMS cd="set1" name="in"/> <OMV name="x"/>
        <OMS cd="setname1" name="N"/>
      </OMA>
      <OMA>
        <OMS cd="set1" name="in"/>
        <OMA>
          <OMS cd="arith1" name="plus"/>
          <OMV name="x"/> <OMI> 1 </OMI>
        </OMA>
        <OMS cd="setname1" name="N"/>
      </OMA>
    </OMA>
  </OMBIND>
</OMOBJ>
```

An example: for all x in the natural numbers, x+1 is a natural number. (3)

The OpenMath equivalent to the MathML for a=b=c, which is:

```
<apply>
  <eq/>
  <ci> a </ci>
  <ci> b </ci>
  <ci> c </ci>
</apply>
```

is

```
<OMOBJ>
  <OMA>
    <OMS cd="logic1" name="and"/>
    <OMA>
      <OMS cd="relation1" name="eq"/>
      <OMV name="a"/>
      <OMV name="b"/>
    </OMA>
    <OMA>
      <OMS cd="relation1" name="eq"/>
      <OMV name="b"/>
      <OMV name="c"/>
    </OMA>
  </OMA>
</OMOBJ>
```

Illustration of the *unfolding* of the MathML nary equality operator to enable the use of the binary OpenMath **eq** noperator by using the OpenMath **and** operator.

(4)

The OpenMath equivalent to the MathML for a>b>c, which is:

```
<apply>
  <gt/>
  <ci> a </ci>
  <ci> b </ci>
  <ci> c </ci>
</apply>
```

is

```
<OMOBJ>
  <OMA>
    <OMS cd="logic1" name="and"/>
    <OMA>
      <OMS cd="relation1" name="gt"/>
      <OMV name="a"/>
      <OMV name="b"/>
    </OMA>
    <OMA>
      <OMS cd="relation1" name="gt"/>
      <OMV name="b"/>
      <OMV name="c"/>
    </OMA>
  </OMA>
</OMOBJ>
```

Illustration of the *unfolding* of the MathML nary greater than operator to enable the use of the binary OpenMath **gt** operator by using the OpenMath **and** operator.

(5)

The MathML version:

```
<apply>
  <int/>
  <bvar>
    <ci> x </ci>
  </bvar>
  <apply>
    <sin/>
    <ci> x </ci>
  </apply>
</apply>
```

The OpenMath version:

```
<OMOBJ>
  <OMA>
    <OMS cd="calculus1" name="int"/>
    <OMBIND>
      <OMS cd="fns1" name="lambda"/>
      <OMBVAR>
        <OMV name="x"/>
      </OMBVAR>
      <OMA>
        <OMS cd="transc1" name="sin"/>
        <OMV name="x"/>
      </OMA>
    </OMBIND>
  </OMA>
</OMOBJ>
```

Representation of the integral of the function $\sin(x)$ with respect to x. (6)

The MathML version:

```
<apply>
  <diff/>
  <bvar>
    <ci> x </ci>
  </bvar>
  <apply>
    <sin/>
    <ci> x </ci>
  </apply>
</apply>
```

The OpenMath version:

```
<OMOBJ>
  <OMA>
    <OMS cd="calculus1" name="diff"/>
    <OMBIND>
      <OMS cd="fns1" name="lambda"/>
      <OMBVAR>
        <OMV name="x"/>
      </OMBVAR>
      <OMA>
        <OMS cd="transc1" name="sin"/>
        <OMV name="x"/>
      </OMA>
    </OMBIND>
  </OMA>
</OMOBJ>
```

Representation of the differential of the function $\sin(x)$ with respect to x.

(7)

The MathML version:

```
<apply>
  <partialdiff/>
  <bvar> x </bvar>
  <bvar> z </bvar>
  <cn> 3 </cn>
  <apply>
    <times/>
    <ci> x </ci><ci> y </ci><ci> z </ci>
  </apply>
</apply>
```

The OpenMath version:

```
<OMOBJ>
  <OMA>
    <OMS cd="calculus1" name="partialdiff"/>
    <OMA>
      <OMS cd="list1" name="list"/>
      <OMI> 1 </OMI>
      <OMI> 0 </OMI>
      <OMI> 3 </OMI>
    </OMA>
    <OMBIND>
      <OMS cd="fns1" name="lambda"/>
      <OMBVAR>
        <OMV name="x"/>
        <OMV name="y"/>
        <OMV name="z"/>
      </OMBVAR>
      <OMA>
        <OMS cd="arith1" name="times"/>
        <OMV name="x"/>
        <OMV name="y"/>
        <OMV name="z"/>
      </OMA>
    </OMBIND>
  </OMA>
</OMOBJ>
```

Representation of the partial-differential expression $\frac{\partial^4}{\partial x \partial z^3}(xyz)$. (8)

The MathML version:

```
<set>
  <bvar> x </bvar>
  <condition>
    <apply>
      <in/>
      <apply>
        <divide/>
        <ci> x </ci>
        <cn> 2 </cn>
      </apply>
      <integers/>
    </apply>
  </condition>
</set>
```

The OpenMath version:

```
<OMOBJ>
  <OMA>
    <OMS cd="set1" name="suchthat"/>
    <OMS cd="setname1" name="Z"/>
    <OMBIND>
      <OMS cd="funs1" name="lambda"/>
      <OMBVAR> <OMV name="x"/> </OMBVAR>
      <OMA>
        <OMS cd="set1" name="in"/>
        <OMA>
          <OMS cd="arith1" name="divide"/>
          <OMV name="x"/>
          <OMI> 2 </OMI>
        </OMA>
        <OMS cd="setname1" name="Z"/>
      </OMA>
    </OMBIND>
  </OMA>
</OMOBJ>
```

Construct the set of even integers, using the suchthat constructor. Note that making the integers ($\mathbf{Z}$) the second argument of suchthat is a deep semantic transformation, as implied in the MathML version.

(9)

The MathML version:

```
<list>
  <bvar> x </bvar>
<condition>
  <apply>
    <and/>
    <apply>
      <geq/>
      <ci> x </ci> <cn> 0 </cn>
    </apply>
    <apply>
      <lt/>
      <ci> x </ci> <cn> 100 </cn>
    </apply>
    <apply>
      <in/>
      <apply>
        <divide/>
        <ci> x </ci> <cn> 2 </cn>
      </apply>
      <integers/>
    </apply>
  </apply>
</condition>
</list>
```

The OpenMath version:

```
<OMOBJ>
  <OMA>
    <OMS cd="list1" name="suchthat"/>
    <OMS cd="nums1" name="Z"/>
    <OMBIND>
      <OMS cd="funs1" name="lambda"/>
      <OMBVAR>
        <OMV name="x"/>
      </OMBVAR>
      <OMA>
        <OMS cd="logic1" name="and"/>
        <OMA>
          <OMS cd="relation1" name="geq"/>
          <OMV name="x"/>
          <OMS cd="alg1" name="zero"/>
        </OMA>
        <OMA>
          <OMS cd="relation1" name="lt"/>
          <OMV name="x"/>
          <OMI> 100 </OMI>
        </OMA>
        <OMA>
          <OMS cd="set1" name="in"/>
          <OMA>
            <OMS cd="arith1" name="divide"/>
            <OMV name="x"/> <OMI> 2 </OMI>
          </OMA>
          <OMS cd="funs1" name="Z"/>
        </OMA>
      </OMA>
    </OMBIND>
  </OMA>
</OMOBJ>
```

This example shows how to construct the list of even positive integers less than 100, using the suchthat constructor.

(10)