SCIENCE
Symbolic
Computation
Infrastructure for
Europe

OpenMath in SCIEnce: Content Dictionaries

`scscp1, scscp2, order1, polynomial4, matrix1`

Sebastian Freundt

Peter Horn

Alexander Konovalov

Sylla Lesseni

Steve Linton

Dan Roozemond

22nd OpenMath Workshop,

Grand Bend,

July 9th, 2009.

`www.symbolic-computation.org`

SCSCP

Symbolic Computation
Software Composability Protocol

scscp1 and scscp2

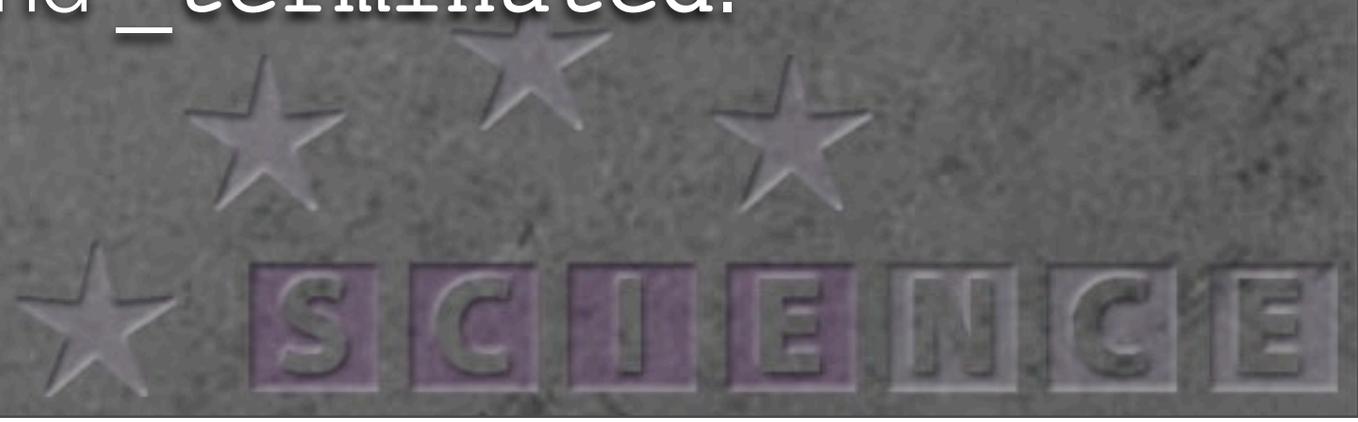
Two Content Dictionaries that describe the symbols used in SCSCP.

SCSCP
Symbolic Computation
Software Composability Protocol

`scscp1` and `scscp2`

Main symbols (`scscp1`):

- **`procedure_call:`** The actual procedure call. Only argument is an OMA describing the procedure to be called and the arguments.
- **`procedure_completed:`** The result (if you're lucky)
- **`procedure_terminated:`** The result (otherwise)
- **`call_id:`** The unique identifier that comes with `_call`, `_completed`, and `_terminated`.



SCSCP
Symbolic Computation
Software Composability Protocol

scscp1 and scscp2

```
<OMATTR>
    <OMATP>
        <OMS cd="scscp1" name="call_id"/>
        <OMSTR>scscp.symcomp.org:26133:7617:eBFyqFae</OMSTR>
        <OMS cd="scscp1" name="option_return_object"/>
        <OMSTR/>
    </OMATP>
    <OMA>
        <OMS cd="scscp1" name="procedure_call"/>
        <OMA>
            <OMS cd="scscp_transient_1" name="GroupIDService"/>
            <OMA>
                <OMS cd="permgp1" name="group"/>
                <OMS cd="permutation1" name="right_compose"/>
                <OMA>
                    <OMS cd="permut1" name="permutation"/>
                    <OMI>2</OMI>
                    <OMI>1</OMI>
                </OMA>
            </OMA>
        </OMA>
    </OMA>
</OMATTR>
```

```
(scscp1.procedure_call(
  scscp_transient_1.GroupIDService(
    permgp1.group(
      permutation1.right_compose,
      permut1.permutation(2, 1)
)))){
  scscp1.option_return_object->'',
  scscp1.call_id->'scscp.symcomp.org:26133:7617:eBFyqFae'
}
```

SCSCP
Symbolic Computation
Software Composability Protocol

scscp1 and scscp2

```
<OMATTR>
    <OMATP>
        <OMS cd="scscp1" name="call_id"/>
        <OMSTR>scscp.symcomp.org:26133:7617:eBFyqFae</OMSTR>
    </OMATP>
    <OMA>
        <OMS cd="scscp1" name="procedure_completed"/>
        <OMA>
            <OMS cd="list1" name="list"/>
            <OMI>2</OMI>
            <OMI>1</OMI>
        </OMA>
    </OMA>
</OMATTR>
```

```
(scscp1.procedure_completed([2, 1]))
{
  scscp1.call_id->'scscp.symcomp.org:26133:7617:eBFyqFae'
}
```

SCSCP
SYMBOLIC COMPUTATION
SOFTWARE COMPOSABILITY PROTOCOL

scscp1 and scscp2

```
<OMATTR>
    <OMATP>
        <OMS cd="scscp1" name="call_id"/>
        <OMSTR>1324765</OMSTR>
        <OMS cd="scscp1" name="option_return_object"/>
        <OMSTR/>
    </OMATP>
    <OMA>
        <OMS cd="scscp1" name="procedure_call"/>
        <OMA>
            <OMS cd="scscp2" name="get_allowed_heads"/>
        </OMA>
    </OMA>
</OMATTR>
```

```
(scscp1.procedure_call(
  scscp2.get_allowed_heads()
)){
  scscp1.option_return_object->'',
  scscp1.call_id->'1324765'
}
```

SCIENCE

SCSCP
SYMBOLIC COMPUTATION
SOFTWARE COMPOSABILITY PROTOCOL

scscp1 and scscp2

```
<OMATTR>
    <OMATP>
        <OMS cd="scscp1" name="call_id"/>
        <OMSTR>1324765</OMSTR>
    </OMATP>
    <OMA>
        <OMS cd="scscp1" name="procedure_completed"/>
        <OMA>
            <OMS cd="scscp2" name="symbol_set"/>
            <OMS cd="scscp_transient_1" name="GroupIDService"/>
            <OMS cd="group1" name="group"/>
            <OMA>
                <OMS cd="meta" name="CDName"/>
                <OMSTR>permut1</OMSTR>
            </OMA>
            <OMA>
                <OMS cd="metagrp" name="CDGroupName"/>
                <OMSTR>scscp</OMSTR>
            </OMA>
        </OMA>
    </OMA>
</OMATTR>
```

```
(scscp1.procedure_completed(scscp2.symbol_set(
  scscp_transient_1.GroupIDService,
  group1.group,
  meta.CDName('permut1'),
  metagrp.CDGroupName('scscp')
))){scscp1.call_id->'1324765'}
```

SCSCP
Symbolic Computation
Software Composability Protocol

scscp1 and scscp2

Other symbols (scscp1):

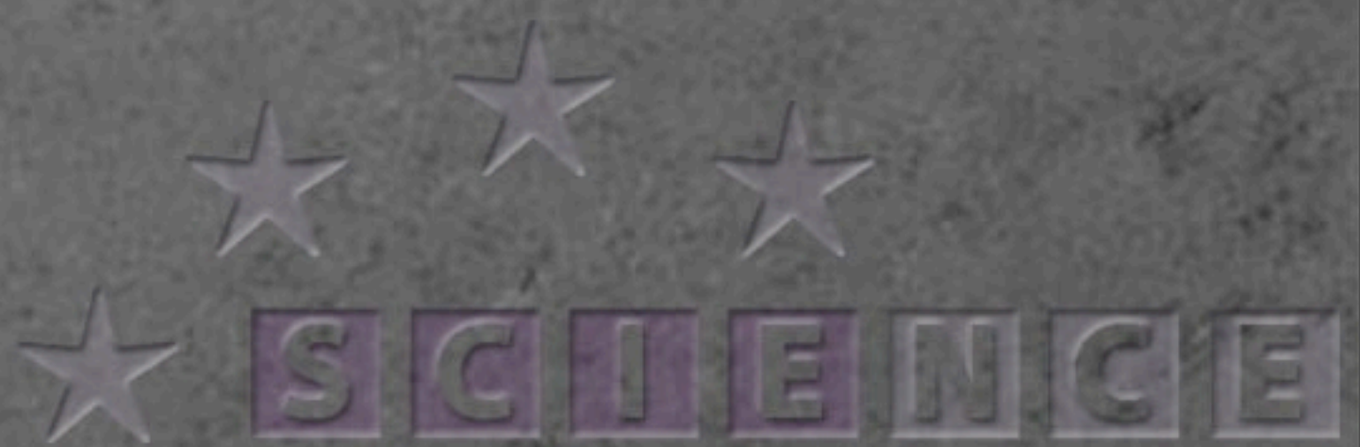
- option_debuglevel, option_max_memory, option_min_memory, option_return_cookie, option_return_nothing, option_return_object, option_runtime
- info_memory, info_message, info_runtime
- error_memory, error_runtime, error_system_specific

SCSCP
Symbolic Computation
Software Composability Protocol

scscp1 and scscp2

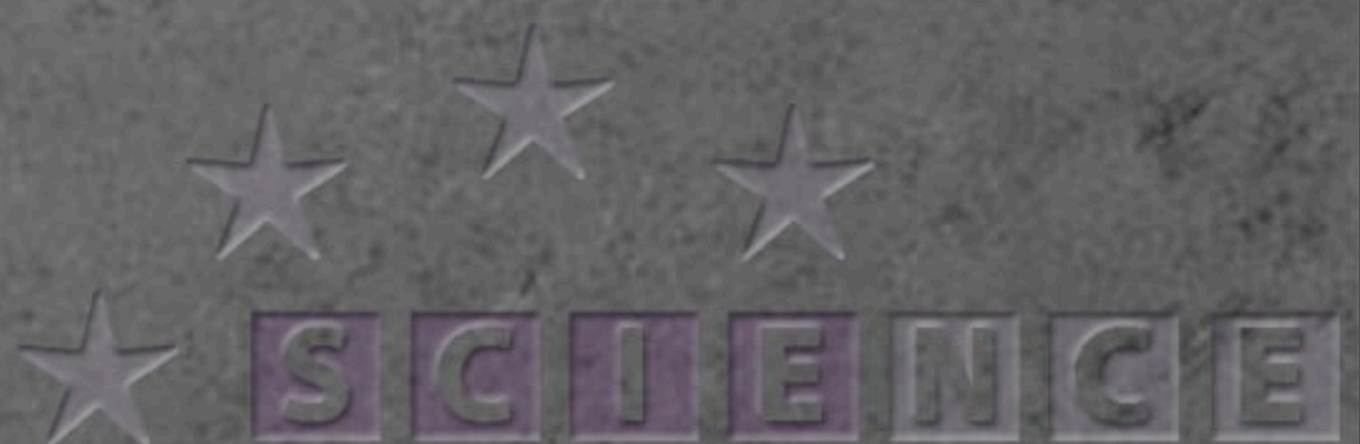
Other symbols (scscp2):

- store_session, store_persistent, retrieve, unbind
- get_allowed_heads, is_allowed_head, get_signature, signature, get_service_description, service_description
- get_transient_cd



order1

A Content Dictionary for (orders of) number fields: order1 provides the basic functions and constructors in this branch of mathematics.



order1

```
<OMA>
   <OMS cd="order1" name="order"/>
   <OMS cd="ringname1" name="Z"/>
   <OMA>
      <OMS name="DMP" cd="polyd1"/>
      <OMA>
         <OMS name="poly_ring_d" cd="polyd1"/>
         <OMS name="Z" cd="ringname1"/>
         <OMI>1</OMI>
      </OMA>
      <OMA>
         <OMS name="SDMP" cd="polyd1"/>
         <OMA>
            <OMS name="term" cd="polyd1"/>
            <OMI>1</OMI>
            <OMI>2</OMI>
         </OMA>
         <OMA>
            <OMS name="term" cd="polyd1"/>
            <OMI>3</OMI>
            <OMI>0</OMI>
         </OMA>
      </OMA>
   </OMA>
</OMA>
```

```
order1.order(
  ringname1.Z,
  polyd1.DMP(
    polyd1.poly_ring_d(ringname1.Z, 1),
    polyd1.SDMP(
      polyd1.term(1, 2),
      polyd1.term(3, 0)
)))
```

order1

```
<OMA>
   <OMS cd="logic1" name="implies"/>
   <OMA>
      <OMS cd="relation1" name="eq"/>
      <OMV name="A"/>
      <OMA>
         <OMS cd="order1" name="ring_integers"/>
         <OMV name="K"/>
      </OMA>
   </OMA>
   <OMA>
      <OMS cd="logic1" name="and"/>
      <OMA>
         <OMS cd="ring1" name="is_subring"/>
         <OMV name="K"/>
         <OMV name="A"/>
      </OMA>
      <OMA>
         <OMS cd="order1" name="is_Dedekind"/>
         <OMV name="A"/>
      </OMA>
   </OMA>
</OMA>
```

```
$A = order1.ring_integers($K) ==> ring1.is_subring($K, $A) or order1.is_Dedekind($A)
```

order1

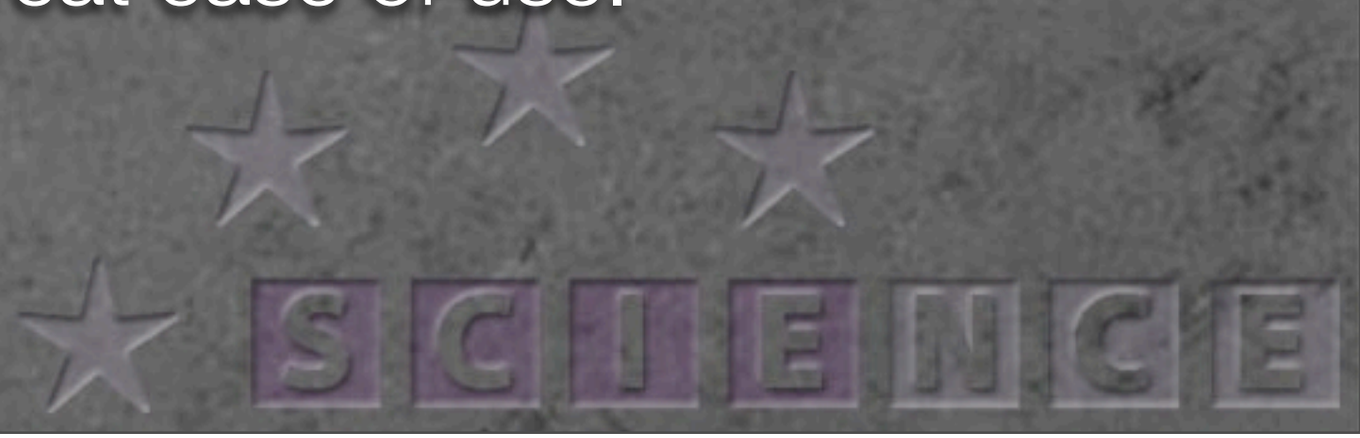
The symbols:

- `is_Dedekind, is_nonzero_divisor, is_principal_ideal_domain`
- `order, maximal_order, is_maximal_order, algebraic_integer`
- `number_field, algebraic_number, ring_integers, primitive_element`

`polynomial4`

More efficient and complete transmission of properties of polynomials, in particular:

- designed with polyd in mind,
- symbols to hold both quotient and remainder of polynomial division,
- extended support for factorisation, over any ring,
- increased verbosity for great ease of use.



polynomial4

```
<OMA>
   <OMS name="factorise" cd="polynomial4"/>
   <OMA>
      <OMS name="DMP" cd="polyd1"/>
      <OMA>
         <OMS name="poly_ring_d_named" cd="polyd1"/>
         <OMS name="Z" cd="setname1"/>
         <OMV name="X"/>
      </OMA>
      <OMA>
         <OMS name="SDMP" cd="polyd1"/>
         <OMA>
            <OMS name="term" cd="polyd1"/>
            <OMI> 1 </OMI>
            <OMI> 2 </OMI>
         </OMA>
         <OMA>
            <OMS name="term" cd="polyd1"/>
            <OMI> -1 </OMI>
            <OMI> 0 </OMI>
         </OMA>
      </OMA>
   </OMA>
</OMA>
```

```
polynomial4.factorise(
  polyd1.DMP(
    polyd1.poly_ring_d_named(setname1.Z, $X),
    polyd1.SDMP(
      polyd1.term(1, 2), polyd1.term(-1, 0)
)))
```

polynomial4

```
<OMA>
  <OMS name="factorisations" cd="polynomial4"/>
  <OMS name="factorisations_complete" cd="polynomial4"/>
  <OMA>
    <OMS name="factors" cd="polynomial4"/>
    <OMS name="definitely_irreducible" cd="polynomial4"/>
    <OMA id="R">
      <OMS name="poly_ring_d_named" cd="polyd1"/>
      <OMS name="Z" cd="ringname1"/>
      <OMV name="X"/>
    </OMA>
    <!-- the common coefficient -->
    <OMA><OMS name="one" cd="alg1"/><OMS name="Z" cd="ringname1"/></OMA>
    <!-- the 1st factor -->
    <OMA>
      <OMS name="factor" cd="polynomial4"/>
      <OMA>
        <OMS name="DMP" cd="polyd1"/>
        <OMR href="#R"/>
        <OMA>
          <OMS name="SDMP" cd="polyd1"/>
          <OMA><OMS name="term" cd="polyd1"/><OMI> 1</OMI><OMI>1</OMI></OMA>
          <OMA><OMS name="term" cd="polyd1"/><OMI>-1</OMI><OMI>0</OMI></OMA>
        </OMA>
      </OMA>
      <OMA>
        <OMS name="multiplicity" cd="polynomial4"/>
        <OMI> 1 </OMI>
      </OMA>
    </OMA>
    <!-- the 2nd factor -->
    <OMA>
      <OMS name="factor" cd="polynomial4"/>
      <OMA>
```

```
      </OMA>
    </OMA>
    <OMA>
      <OMS name="multiplicity" cd="polynomial4"/>
      <OMI> 1 </OMI>
    </OMA>
  </OMA>
  <!-- the 2nd factor -->
  <OMA>
    <OMS name="factor" cd="polynomial4"/>
    <OMA>
      <OMS name="DMP" cd="polyd1"/>
      <OMR href="#R"/>
      <OMA>
        <OMS name="SDMP" cd="polyd1"/>
        <OMA>
          <OMS name="term" cd="polyd1"/>
          <OMI> 1 </OMI>
          <OMI> 1 </OMI>
        </OMA>
        <OMA>
          <OMS name="term" cd="polyd1"/>
          <OMI> 1 </OMI>
          <OMI> 0 </OMI>
        </OMA>
      </OMA>
    </OMA>
    <OMA>
      <OMS name="multiplicity" cd="polynomial4"/>
      <OMI> 1 </OMI>
    </OMA>
  </OMA>
</OMA>
```

polynomial4

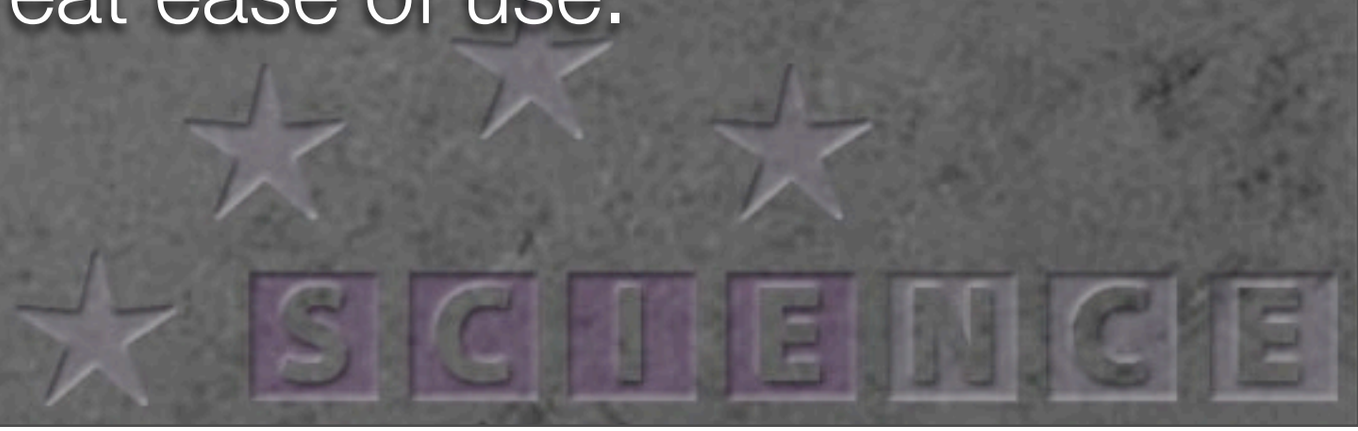
The symbols:

- factorise, factorisations, factorisations_complete, factorisations_incomplete, factors, factor, multiplicity, ground_ring_injected
- divide, quotient_remainder, quotient, remainder

matrix1

More efficient and complete transmission of matrices, in particular:

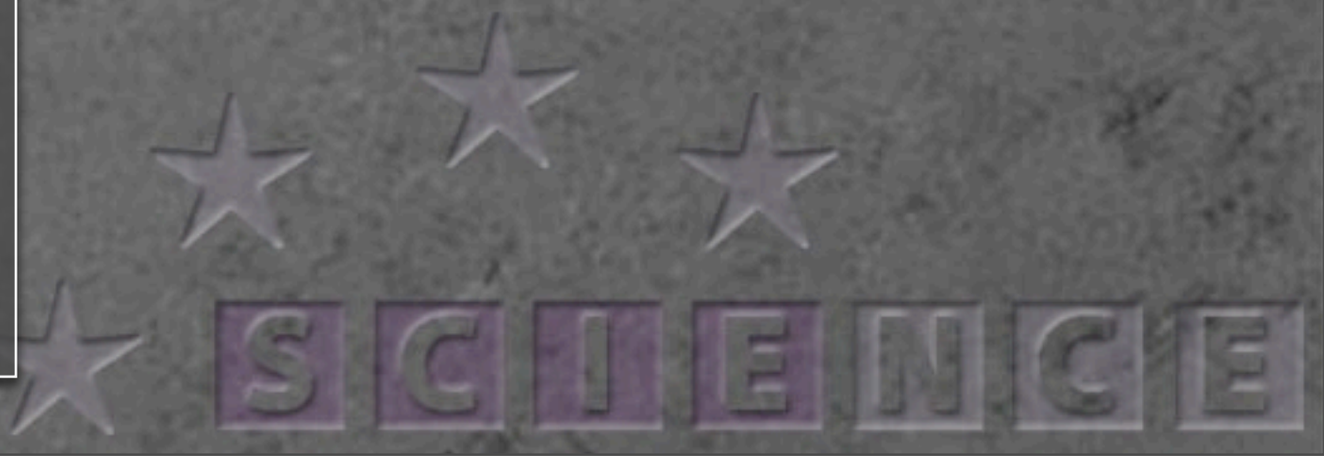
- the possibility to specify the entry domain of the matrix beforehand, giving more efficient transmission of matrices over finite fields,
- support for sparse matrices, including matrices built from blocks, bands, or just entries,
- increased verbosity for great ease of use.



matrix1

```
<OMA>
    <OMS name="matrix" cd="matrix1"/>
    <OMA>
        <OMS name="matrix_domain" cd="matrix1"/>
        <OMA>
            <OMS name="entry_domain" cd="matrix1"/>
            <OMS name="Z" cd="ringname1"/>
        </OMA>
        <OMA>
            <OMS name="row_dimension" cd="matrix1"/>
            <OMI>3</OMI>
        </OMA>
        <OMA>
            <OMS name="column_dimension" cd="matrix1"/>
            <OMI>3</OMI>
        </OMA>
    </OMA>
    <OMA>
        <OMS cd="matrix1" name="dense"/>
        <OMI>1</OMI><OMI>2</OMI><OMI>3</OMI><OMI>4</OMI><OMI>5</OMI>
        <OMI>6</OMI><OMI>7</OMI><OMI>8</OMI><OMI>9</OMI>
    </OMA>
</OMA>
```

```
matrix1.matrix(
  matrix1.matrix_domain(
    matrix1.entry_domain(ringname1.Z),
    matrix1.row_dimension(3),
    matrix1.column_dimension(3)
  ),
  matrix1.dense(1, 2, 3, 4, 5, 6, 7, 8, 9)
)
```



matrix1

```
<OMA>
   <OMS name="matrix" cd="matrix1"/>
   <OMA>
      <OMS name="matrix_domain" cd="matrix1"/>
      [...]
   </OMA>
   <OMA>
      <OMS cd="matrix1" name="sparse"/>
      <OMA>
         <OMS cd="matrix1" name="sparse_entry"/>
         <OMI>10</OMI>
         <OMI>20</OMI>
         <OMA>
            <OMS cd="matrix1" name="block"/>
            <OMA>
               <OMS name="row_dimension" cd="matrix1"/>
               <OMI>2</OMI>
            </OMA>
            <OMA>
               <OMS name="column_dimension" cd="matrix1"/>
               <OMI>2</OMI>
            </OMA>
            <OMA>
               <OMS cd="matrix1" name="dense"/>
               <OMI>11</OMI>
               <OMI>12</OMI>
               <OMI>21</OMI>
               <OMI>22</OMI>
            </OMA>
         </OMA>
      </OMA>
   </OMA>
</OMA>
```

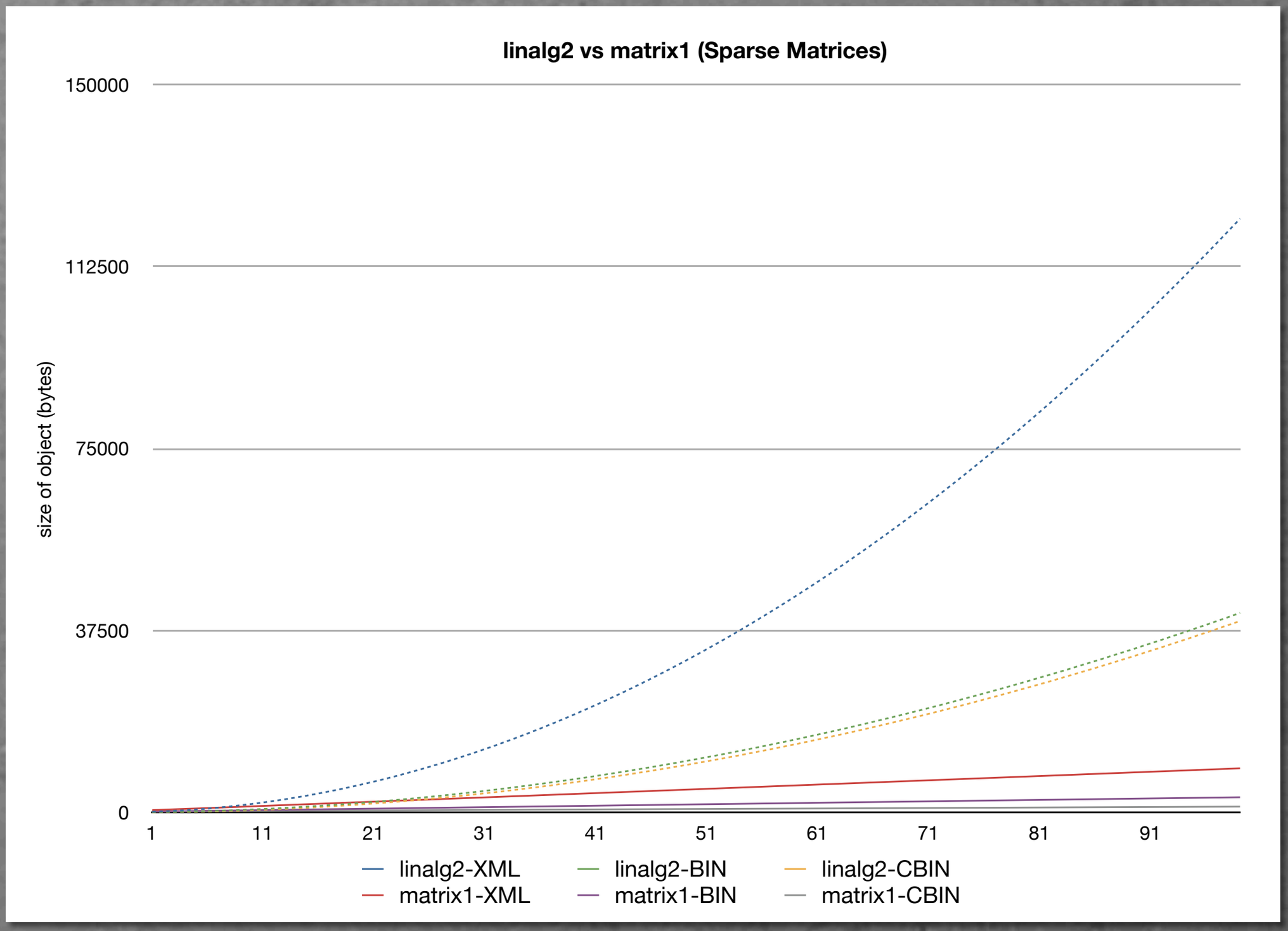
```
matrix1.matrix(matrix1.matrix_domain(matrix
1.entry_domain(fieldname1.Q),
matrix1.row_dimension(30),
    matrix1.column_dimension(30)
  ),
  matrix1.sparse(
    matrix1.sparse_entry(10, 20,
      matrix1.block(
        matrix1.row_dimension(2),
        matrix1.column_dimension(2),
        matrix1.dense(11, 12, 21, 22)
  )
)))
```

matrix1

The symbols:

- `matrix`
- `matrix_domain, entry_domain, row_dimension, column_dimension`
- `dense, sparse, sparse_entry`
- `diagonal, block, banded, upper_band, lower_band`

matrix1


linalg2 vs matrix1 (Sparse Matrices)
150000
112500
75000
37500
0
size of object (bytes)
1
11
21
31
41
51
61
71
81
91
linalg2-XML
matrix1-XML
linalg2-BIN
matrix1-BIN
linalg2-CBIN
matrix1-CBIN

Thank you!

Please come and see our posters!