The `polymake` XML File Format

Michael Joswig

TU Berlin

12 July 2016

joint w/ Ewgenij Gawrilow
Simon Hampe

`polymake`

Software for polyhedral geometry and more

An XML File Format

Goal: maximum flexibility

`Relax NG`

A schema language for XML

polymake

- software for polyhedral geometry and applications
  - convex polytopes, cones, polyhedral fans, matroids, ...
  - linear/combinatorial optimization
  - toric/tropical geometry
  - combinatorial topology
- open source, GNU Public License
- co-authored (since 1996) with Ewgenij Gawrilow and many other contributors
  - current version 3.0 from January 2016

```
polytope> $P = product( cube(3), dodecahedron() );
polytope> print $P->F_VECTOR;
160 480 576 352 114 18
```

polymake.org

`polymake` Design Concept

many object types

- wide range of mathematical concepts
- each version of the software may define new objects
- extension system for third party authors

individual objects ...

- may have many properties of various types
- only has a small subset of those properties present

properties ...

- may be added with a new version
- encoding may change with a new version

$\rightsquigarrow$ file format needs to be sufficently flexible

Example: polymake XML File Encoding a Square [1/3]

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <?pm chk="56e977e8"?>
3  <object name="square"
4      type="polytope::Polytope&lt;Rational&gt;"
5      version="3.0"
6      xmlns="http://www.math.tu-berlin.de/polymake/#3">
7    <description><![CDATA[2-cube]]></description>
8    <property name="VERTICES">
9      <m>
10       <v>1 0 0</v>
11       <v>1 1/3 0</v>
12       <v>1 0 1/3</v>
13       <v>1 1/3 1/3</v>
14     </m>
15   </property>
16   <property name="FACETS"
17       type="SparseMatrix&lt;Rational,NonSymmetric&gt;">
18     <m cols="3">
19       <v> <e i="1">1</e> </v>
20       <v> <e i="0">1/3</e> <e i="1">-1</e> </v>
```

Example: polymake XML File Encoding a Square [2/3]

```
13          <v>1  1/3  1/3</v>
14        </m>
15      </property>
16      <property name="FACETS"
17          type="SparseMatrix&lt;Rational,NonSymmetric&gt;">
18        <m cols="3">
19          <v> <e i="1">1</e> </v>
20          <v> <e i="0">1/3</e> <e i="1">-1</e> </v>
21          <v> <e i="2">1</e> </v>
22          <v> <e i="0">1/3</e> <e i="2">-1</e> </v>
23        </m>
24      </property>
25      <property name="LINEALITY_SPACE"><m /></property>
26      <property name="BOUNDED" value="true" />
27      <property name="N_FACETS" value="4" />
28      <property name="N_VERTICES" value="4" />
29      <property name="VOLUME" value="1/9" />
30      <property name="TRIANGULATION">
31        <object name="unnamed#0">
32          <property name="FACETS">
```

Example: polymake XML File Encoding a Square [3/3]

```
25      <property name="LINEALITY_SPACE"><m /></property>
26      <property name="BOUNDED" value="true" />
27      <property name="N_FACETS" value="4" />
28      <property name="N_VERTICES" value="4" />
29      <property name="VOLUME" value="1/9" />
30      <property name="TRIANGULATION">
31        <object name="unnamed#0">
32          <property name="FACETS">
33            <m>
34              <v>0 1 2</v>
35              <v>1 2 3</v>
36            </m>
37          </property>
38          <property name="F_VECTOR">
39            <v>4 5 2</v>
40          </property>
41        </object>
42      </property>
43    </object>
```

Two Features

- top level object type and version

```
<object name="square"
    type="polytope::Polytope&lt;Rational&gt;"
    version="3.0"
    xmlns="http://www.math.tu-berlin.de/polymake/#3">
```

- content elements serialized with few XML tags

```
  <property name="VERTICES">
    <m>
      <v>1 0 0</v>
      <v>1 1/3 0</v>
      <v>1 0 1/3</v>
      <v>1 1/3 1/3</v>
    </m>
  </property>
```

Feature Overview

- top level object type and version
- content elements serialized with few XML tags
- property types
  - encoded by `polymake` version
  - implemented via C++ class template library
- template types come with their serialization
  - future goal: let `polymake` write RNG schema for all properties of all top level objects of a given version
- XSLT transformation for conversion between versions

Example: An Array of Polynomials

```
<data type="Array&lt;Polynomial&lt;QuadraticExtension&gt;&g
      version="3.0"
      xmlns="http://www.math.tu-berlin.de/polymake/#3">
  <v>
    <t>
      <m>
        <t>
          <v dim="2"> <e i="0">2</e> </v>
          <t>0 1/5 5</t>
        </t>
        <t>
          <v dim="2"> <e i="1">3</e> </v>
          <t>-1 0 0</t>
        </t>
      </m>
      <t id="1">
        <v>x y</v>
      </t>
    </t>
  </v>
</data>
```

RELAX NG: An XML Schema Language

Clark, Murata et al. 2001

- complexity between DTD and XML Schema
- concept based on finite tree automata
  - great expressive power (e.g., composability, nesting)
  - efficient to parse
  - no inheritance
- supports types and XML namespaces
- infrastructure
  - has both an XML syntax and a compact non-XML syntax
  - software support for validation and conversion

relaxng.org

RELAX NG compact syntax

`start = ...`	Defines the pattern for the root element.
`PatternName = ...`	Defines a pattern with a chosen name.
`element, attribute`	Define XML tags / attributes.
`{ ... }`	Describes the content of an element or attribute.
`|`	Pattern alternatives.
`&`	Combine two patterns in arbitrary order.
`?,+,*`	Quantifiers: At most one, one or more, zero or more.

polymake RNG Top Level polymake RNG Content Elements

Conclusion

- XML file format which is fairly light weight
- easily encodes all kinds of objects
  occurring in geometric or algebraic computations
- flexible enough for software environment
  which evolves over time

polymake RNG Schema — Top Level [1/2]

```
start = TopObject | LooseData

TopObject = element object { TopAttribs, ObjectContent }

TopAttribs = attribute type {
    xsd:string { pattern = "[a-zA-Z][a-zA-Z_0-9]*::.*" } },
  attribute version { xsd:string { pattern = "[\d.]+" } }?,
  attribute tm { xsd:hexBinary }?

ObjectContent =
  attribute name { text }?,
  attribute ext { text }?,
  element description { text }?,
  element credit { attribute product { text }, text }*,
  ( Property* & Attachment* )

Property = element property {
    SimpleName,
    attribute ext { text }?,
    ( ( attribute undef { "true" }, empty )
```

polymake RNG Schema — Top Level [2/2]

```
17  Property = element property {
18      SimpleName,
19      attribute ext { text }?,
20      ( ( attribute undef { "true" }, empty )
21        | ( attribute type { text }?, PropertyData )
22        | Text | SubObject+ ) }
23
24  SubObject = element object
25    { attribute type { text }?, ObjectContent }
26
27  Attachment = element attachment {
28    SimpleName, attribute ext { text }?, AttachmentData }
29
30  LooseData = element data {
31    TopAttribs, attribute ext { text }?,
32    element description { text }?, PropertyData }
33
34  SimpleName = attribute name
35    { xsd:string { pattern = "[a-zA-Z][a-zA-Z_0-9]*" } }
```

polymake RNG Schema — Content Elements [1/2]

```
37  PropertyData = ( attribute value { text }, empty )
38    | IdReference | Complex | element m { SubObject+ }
39
40  AttachmentData =
41    ( attribute type { text }?, attribute value { text }, emp
42    | ( attribute type { text }, attribute construct { text }
43      Complex ) | Text
44
45  Text = attribute type { "text" }, text
46
47  Complex = Vector | Matrix | Tuple
48
49  VectorContents = text
50    | ( attribute dim { xsd:nonNegativeInteger }?,
51        ( element e { ElementIndex, text }*
52        | element t { ElementIndex?, TupleContents }+ ) )
53
54  ElementIndex = attribute i { xsd:nonNegativeInteger }
```

polymake RNG Schema — Content Elements [2/2]

```
56  IdReference = element r {
57    attribute id { xsd:nonNegativeInteger }?, empty }
58
59  Vector = element v { VectorContents }
60
61  MatrixContents =
62    ( attribute cols { xsd:nonNegativeInteger }?, Vector* )
63    | ( attribute dim { xsd:nonNegativeInteger },
64        element v { ElementIndex, VectorContents }*)
65    | Matrix+ | Tuple+
66
67  Matrix = element m { MatrixContents }
68
69  TupleContents = attribute id { xsd:nonNegativeInteger }?,
70    ( text | ( Vector | Matrix | Tuple
71               | IdReference | element e { text } )+ )
72
73  Tuple = element t { TupleContents }
```