MGL/MGX formats

Table of contents

1 For version 1.2.x or older 2
1.1 MGL Format 2
1.2 MGX Format 3
2 For version 1.3.x and later 4
2.1 What changed ? 4
2.2 DTD and Relax NG 1.4 4

1. For version 1.2.x or older

1.1. MGL Format

Mascopt uses XML to describe graphs and other network objects rather than simple text format.

Mascopt provides several input and output formats. We only present MGL and MGX which means Mascopt Graph Library and Mascopt Graph Extended. MGL format is the Mascopt's native format and is based on the XML standard. It provides a readable description of Mascopt's objects which can be easily extended for the user's own objects or specializations: as the MGL reader parse an XML file, it can also read a derived file, containing new tags.

One can create his own format implementing the interface WriterInterface which consist only in two methods: an add method to add objects to write and a write method to physically write the file.

The file is quite understandable. The information contained in a MGL file reflects the oriented-object structure of Mascopt. This is a simple manner to enable the share of objects. The listing bellow presents a sample of a code for a digraph.

```
<?xml version="1.0" ?>
<!DOCTYPE OBJECTS SYSTEM
"ftp://ftp-sop.inria.fr/mascotte/mascopt/dtd/mgl_v1.1.dtd">

<OBJECTS>
 <VERTICES>
  <VERTEX id="N0">
   <POSITION>
    <X>50.0</X> <Y>0.0</Y>
   </POSITION>
   <VALUE type="function" dataType="String"> node0 </VALUE>
  </VERTEX>
  <VERTEX id="N1">
   <POSITION>
    <X>10.0</X> <Y>50.0</Y>
   </POSITION>
  </VERTEX>
  <VERTEX id="N2">
   <POSITION>
    <X>0.0</X> <Y>0.0</Y>
   </POSITION>
  </VERTEX>
 </VERTICES>

 <LINKS>
  <ARC id="AE1">
```

```
   <VERTEX_REF idref="N1"/>
   <VERTEX_REF idref="N2"/>
  </ARC>
  <ARC id="AE0">
   <VERTEX_REF idref="N0"/>
   <VERTEX_REF idref="N2"/>
   <VALUE type="Capacity" dataType="Double"> 6.8 </VALUE>
   <VALUE type="length" dataType="Integer"> 110 </VALUE>
  </ARC>
  <ARC id="AE2">
   <VERTEX_REF idref="N0"/>
   <VERTEX_REF idref="N1"/>
  </ARC>
 </LINKS>

 <SETS>
  <VERTEX_SET id="NS0">
   <VERTEX_REF idref="N0"/>
   <VERTEX_REF idref="N1"/>
   <VERTEX_REF idref="N2"/>
  </VERTEX_SET>
  <ARC_SET id="AES0">
   <ARC_REF idref="AE0"/>
   <ARC_REF idref="AE1"/>
  </ARC_SET>
  <ARC_SET id="AES3">
   <ARC_REF idref="AE0"/>
   <ARC_REF idref="AE2"/>
  </ARC_SET>
 </SETS>

 <GRAPHS>
  <DIGRAPH id="G0">
   <VERTEX_SET_REF idref="NS0"/>
   <ARC_SET_REF idref="AES0"/>
  </DIGRAPH>
  <DIGRAPH id="G3">
   <VERTEX_SET_REF idref="NS0"/>
   <ARC_SET_REF idref="AES3"/>
  </DIGRAPH>
 </GRAPHS>
</OBJECTS>
```

1.2. MGX Format

MGX is an extended version of MGL. It solve the following problem: as all objects are shared between graphs, the valuation of an object may differ between a graph and another graph. For example, if an arc a0 is valued with Capacity=6.8, then this value is the same in all graphs containing this edge. As a result, we introduce the notion of context, which precise the context where the value is valid. The simplest way of using context is to put the graph itself as context. It express directly that a value is valid only in this graph.

With two graphs G0 and G3 we can now add contexted values named "Capacity" on arc a0. Then, the differences between MGL and MGX files are shown in the next listing. Note that the default value 6.8 is kept on a0: without context, the valuation system gives this value.

```
a0.setDoubleValue("Capacity",G0, new Double(1.0));
a0.setDoubleValue("Capacity",G3, new Double(3.2));

<ARC id="AE0">
 <VERTEX_REF idref="N0"/>
 <VERTEX_REF idref="N2"/>
 <VALUE type="Capacity" dataType="Double"> 6.8 </VALUE>
 <VALUE type="Capacity" dataType="Double" context="G0"> 1.0 </VALUE>
 <VALUE type="Capacity" dataType="Double" context="G3"> 3.2 </VALUE>
</ARC>
```

2. For version 1.3.x and later

2.1. What changed ?

We totally replaced the parser by a DOM Parser, in ordre to clear the code. It is slower but saffer than before. We validate the document two times: one time with a laxist DTD, then with a Relax NG grammar. With a relax NG grammar, we can express more constraints for the file and the validation of the xml file is improved.

2.2. DTD and Relax NG 1.4

The Validation of files is controlled by the version 1.4 of the DTD and relax NG. Have a look in this directory (ftp://ftp-sop.inria.fr/mascotte/mascopt/dtd) .

Now the files looks like this:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE OBJECTS PUBLIC "SYSTEM"
"ftp://ftp-sop.inria.fr/mascotte/mascopt/dtd/mgl_v1.4.dtd">
<OBJECTS
version="ftp://ftp-sop.inria.fr/mascotte/mascopt/dtd/mgl_v1.4.rng">
  <GRAPHS>
    <DIGRAPH id="G0">
      <VERTEX_SET_REF idref="NS0"/>
      <ARC_SET_REF idref="AES0"/>
    </DIGRAPH>
  </GRAPHS>
  <SETS>
    <VERTEX_SET id="NS0">
      <VERTEX_REF idref="V0"/>
      <VERTEX_REF idref="V1"/>
      <VERTEX_REF idref="V2"/>
```

```
    </VERTEX_SET>
    <ARC_SET id="AES0">
      <ARC_REF idref="AE1"/>
      <ARC_REF idref="AE0"/>
      <VERTEX_SET_REF idref="NS1"/>
    </ARC_SET>
    <VERTEX_SET id="NS1">
      <VERTEX_REF idref="V2"/>
      <VERTEX_REF idref="V0"/>
      <VERTEX_REF idref="V1"/>
    </VERTEX_SET>
  </SETS>
  <VERTICES>
    <VERTEX id="V0">
      <POSITION>
        <X>50.0</X>
        <Y>0.0</Y>
      </POSITION>
      <VALUE dataType="String" type="function">node0</VALUE>
    </VERTEX>
    <VERTEX id="V1">
      <POSITION>
        <X>10.0</X>
        <Y>50.0</Y>
      </POSITION>
    </VERTEX>
    <VERTEX id="V2">
      <POSITION>
        <X>0.0</X>
        <Y>0.0</Y>
      </POSITION>
    </VERTEX>
  </VERTICES>
  <LINKS>
    <ARC id="AE1">
      <VALUE dataType="Double" type="Capacity">6.8</VALUE>
      <VALUE dataType="Integer" type="length">110</VALUE>
      <VERTEX_REF idref="V0"/>
      <VERTEX_REF idref="V2"/>
    </ARC>
    <ARC id="AE0">
      <VERTEX_REF idref="V1"/>
      <VERTEX_REF idref="V2"/>
    </ARC>
  </LINKS>
</OBJECTS>
```