OpenMath

OpenMath functionality in GAP

Version 10.0.4

26 May 2009

Marco Costantini
Alexander Konovalov
Andrew Solomon

Marco Costantini — Email: costanti at science dot unitn dot it
— Address: Department of Mathematics
University of Trento

Alexander Konovalov — Email: alexk at mcs dot st-andrews dot ac dot uk
— Homepage: http://www.cs.st-andrews.ac.uk/~alexk/
— Address: School of Computer Science
University of St Andrews
Jack Cole Building, North Haugh,
St Andrews, Fife, KY16 9SX, Scotland

Andrew Solomon — Email: andrew at illywhacker dot net
— Homepage: http://www.illywhacker.net/
— Address: Faculty of IT
University of Technology, Sydney
Broadway, NSW 2007
Australia

Abstract

The OpenMath package provides an OpenMath phrasebook for GAP: it allows GAP users to import and export mathematical objects encoded in OpenMath, for the purpose of exchanging them with other OpenMath-enabled applications.

Acknowledgements

On various stages the development of the OpenMath package was supported by:

- European Commission through ESPRIT grant EP 24969 "Accessing and Using Mathematical Information Electronically" (see http://web.archive.org/web/20040416013945/http://www.nag.co.uk/projects/OpenMath.html).
- EU FP6 Programme project 026133 "SCIEnce - Symbolic Computation Infrastructure for Europe" (see http://www.symbolic-computation.org/).

We acknowledge with gratitude this support.

Contents

Chapter 1

Introduction and installation

1.1 Brief description of the package

The GAP package OpenMath provides an OpenMath phrasebook for GAP: it allows GAP users to import and export mathematical objects encoded in OpenMath for the purpose of exchanging them with other OpenMath-enabled applications.

This manual describes:

- how to view OpenMath representation of an object;
- how to read OpenMath object from stream or write it to stream for the purposes of exchange with another OpenMath-enabled application;
- how to find which objects can be converted to/from OpenMath using this package;
- how to extend the package to support private OpenMath content dictionaries.

For the detailed information about OpenMath standard and content dictionaries see the OpenMath homepage `http://www.openmath.org`.

For practical purposes, the OpenMath package will be most efficient if used in conjunction with the GAP package SCSCP ([KL]) which implements the Symbolic Computation Software Composability protocol ([FHK+c]). This protocol specifies an OpenMath-based remote procedure call framework, in which all messages (procedure calls and returns of results of successful computation or error messages) are encoded in OpenMath using content dictionaries scscp1 and scscp2 ([FHK+a], [FHK+b]). Using the SCSCP package, GAP can communicate locally or remotely with any other OpenMath-enabled SCSCP-compliant application which may be not only another computer algebra system but also another instance of the GAP system or even, for example, an external C/C++ or Java application. Such communication will go into a seamless manner with the GAP/OpenMath conversion going in the background.

1.2 Installation of the package

To use the OpenMath package it is required to install the GAPDoc package [LN] to use the help system and parse OpenMath objects in the XML format.

To install the OpenMath package, unpack the archive and place the `openmath` directory in the `pkg` subdirectory of your GAP4.4 installation. When you don't have write access to the directory of your

main GAP installation, you can also install the package *outside the GAP main directory* by unpacking it inside a directory `MYGAPDIR/pkg`. Then to be able to load OpenMath you need to call GAP with the `-l ";MYGAPDIR"` option.

The package comes together with the OpenMath C library developed at INRIA and the program `gpipe` which should be compiled to parse the binary OpenMath format. The compilation should be performed in two stages: first the INRIA library, and then `gpipe`. To do this, change to the package directory `gap4r4/pkg/openmath` and enter the following commands:

Example

```
cd OMCv1.3c/src/
./configure
make
cd ../../
./configure ../../
make
```

However, it is possible to ignore this step if the binary OpenMath format will not be used.

Chapter 2

OpenMath functionality in GAP

2.1 Loading the package

The package is loaded as shown below (possibly loading required packages at the same time).

Example

```
gap> LoadPackage("openmath");
-------------------------------------------------------------------------------
Loading  openmath 10.0.4 (OpenMath functionality in GAP)
by Marco Costantini (costanti@science.unitn.it),
   Alexander Konovalov (http://www.cs.st-andrews.ac.uk/~alexk/), and
   Andrew Solomon (http://www.illywhacker.net/).
-------------------------------------------------------------------------------
true
```

2.2 Viewing OpenMath representation of an object

2.2.1 OMPrint

◊ OMPrint(*obj*) (function)

OMPrint writes the default XML OpenMath encoding of GAP object *obj* to the standard output.

Example

```
gap> OMPrint( [ 1, 1/2 ] );
<OMOBJ>
        <OMA>
                <OMS cd="list1" name="list"/>
                <OMI>1</OMI>
                <OMA>
                        <OMS cd="nums1" name="rational"/>
                        <OMI>1</OMI>
                        <OMI>2</OMI>
                </OMA>
        </OMA>
</OMOBJ>
```

```
gap> OMPrint( "This is a string" );
<OMOBJ>
        <OMSTR>This is a string</OMSTR>
</OMOBJ>
gap> OMPrint( 1-2*E(4) );
<OMOBJ>
        <OMA>
                <OMS cd="complex1" name="complex_cartesian"/>
                <OMI>1</OMI>
                <OMI>-2</OMI>
        </OMA>
</OMOBJ>
gap> x:=Indeterminate(Rationals,"x");;
gap> OMPrint(x^2+1);
<OMOBJ>
        <OMA>
                <OMS cd="polyd1" name="DMP"/>
                <OMR href="#polyringepTkNya5l8qdhAmO" />
                <OMA>
                        <OMS cd="polyd1" name="SDMP"/>
                        <OMA>
                                <OMS cd="polyd1" name="term"/>
                                <OMI>1</OMI>
                                <OMI>2</OMI>
                        </OMA>
                        <OMA>
                                <OMS cd="polyd1" name="term"/>
                                <OMI>1</OMI>
                                <OMI>0</OMI>
                        </OMA>
                </OMA>
        </OMA>
</OMOBJ>
gap> OMPrint( Group(1,2,3) );
<OMOBJ>
        <OMA>
                <OMS cd="group1" name="group_by_generators"/>
                <OMA>
                        <OMS cd="permut1" name="permutation"/>
                        <OMI>2</OMI>
                        <OMI>3</OMI>
                        <OMI>1</OMI>
                </OMA>
        </OMA>
</OMOBJ>
```

2.2.2 OMString

◊ OMString(*obj*) (function)

OMString returns a string with the default XML OpenMath encoding of GAP object *obj*. If used

with the noomobj option, then initial and final <OMOBJ> tags will be omitted.

Example

```
gap> OMString(42);
"<OMOBJ> <OMI>42</OMI> </OMOBJ>"
gap> OMString((1,2):noomobj);
"<OMA> <OMS cd=\"permut1\" name=\"permutation\"/> <OMI>2</OMI> <OMI>1</OMI> </OMA>"
```

2.3 Writing and reading OpenMath code to/from streams

2.3.1 OMGetObject

◊ OMGetObject(*stream*) (function)

stream is an input stream (see InputTextFile (**Reference: InputTextFile**), InputTextUser (**Reference: InputTextUser**), InputTextString (**Reference: InputTextString**), InputOutputLocalProcess (**Reference: InputOutputLocalProcess**), InputOutputTCPStream (**SCSCP: InputOutputTCPStream (for client)**), InputOutputTCPStream (**SCSCP: InputOutputTCPStream (for server)**)) with an OpenMath object on it. OMGetObject takes precisely one object off *stream* and returns it as a GAP object. Both XML and binary OpenMath encoding are supported: autodetection is used. This function requires either that the GAP package GAPDoc is available (for XML OpenMath), or that the external program gpipe, included in this package, has been compiled (for both XML and binary OpenMath). This may be used to retrieve objects from a file, for example:

Example

```
gap> test3:=Filename(DirectoriesPackageLibrary("openmath","tst"),"test3.omt");;
gap> stream := InputTextFile( test3 );;
gap> OMGetObject(stream);
912873912381273891
gap> OMGetObject(stream);
E(4)
gap> CloseStream(stream);
```

or it can be used to retrieve them from standard input - one may paste an OpenMath object directly into standard input after issuing GAP with the following commands:

Example

```
gap> stream := InputTextUser();;
gap> g := OMGetObject(stream);CloseStream(stream);
```

2.3.2 OMPutObject

◊ OMPutObject(*stream, obj*) (function)

OMPutObject writes (appends) the XML OpenMath encoding of the GAP object *obj* to output stream *stream* (see InputTextFile (**Reference: InputTextFile**), OutputTextUser (**Reference: OutputTextUser**), OutputTextString (**Reference: OutputTextString**), InputOutputTCPStream (**SCSCP: InputOutputTCPStream (for client)**), InputOutputTCPStream (**SCSCP: InputOutputTCPStream (for server)**)).

Example

```
gap> g := [[1,2],[1,0]];;
gap> t := "";
""
gap> s := OutputTextString(t, true);;
gap> OMPutObject(s, g);
gap> CloseStream(s);
gap> Print(t);
<OMOBJ>
        <OMA>
                <OMS cd="linalg2" name="matrix"/>
                <OMA>
                        <OMS cd="linalg2" name="matrixrow"/>
                        <OMI>1</OMI>
                        <OMI>2</OMI>
                </OMA>
                <OMA>
                        <OMS cd="linalg2" name="matrixrow"/>
                        <OMI>1</OMI>
                        <OMI>0</OMI>
                </OMA>
        </OMA>
</OMOBJ>
```

2.3.3 OMPlainString

◇ OMPlainString(*string*) (function)

OMPlainString wraps the string into a GAP object of a special kind called an OpenMath plain string. Internally such object is represented as a string, but OMPutObject (2.3.2) threat it in a different way: instead of converting it into a <OMSTR> object, an OpenMath plain string will be plainly substituted into the output (this explains its name) without decorating it with <OMSTR> tags.

It is assumed that OpenMath plain string contains valid OpenMath code; no actual validation is performed during its creation. Such functionality may be useful to compose some OpenMath code at the GAP level to communicate it to the other system, in particular, to send there symbols which are not supported by GAP, for example:

Example

```
gap> s:=OMPlainString("<OMS cd=\"nums1\" name=\"pi\"/>");
<OMS cd="nums1" name="pi"/>
gap> OMPrint(s);
<OMOBJ>
        <OMS cd="nums1" name="pi"/>
```

```
</OMOBJ>
```

2.4 Utilities

2.4.1 OMTest

◊ OMTest(*obj*) (function)

Converts *obj* to OpenMath and back. Returns true iff *obj* is unchanged (as a GAP object) by this operation. The OpenMath standard does not stipulate that converting to and from OpenMath should be the identity function so this is a useful diagnostic tool.

Example

```
gap> OMTest( [ [1..10],[1/2,2+E(4)], ZmodnZObj(2,6), (1,2), true, "string" ] );
true
```

Chapter 3

Extending the OpenMath package

3.1 Exploring the range of supported symbols

The OpenMath package supports such basic OpenMath objects as integers (<OMI>), character strings (<OMSTR>) and variables (<OMVAR>). Besides that, it supports a number of OpenMath content dictionaries (some of them only partially, dependently on their relevance to GAP). To see which symbols from which content dictionaries are supported for the conversion from OpenMath to GAP, explore the global record `OMsymRecord`. Its components have names of appropriate CDs, and subcomponents of each component have names of symbols from the corresponding CD. If the value of the component is not equal to `fail`, then it contains the function or the object which is used for conversion. The following example of the entry for the nums1 CD demonstrates a combination of all possible cases:

Example

```
gap> Print(OMsymRecord.nums1,"\n");
rec(
  e := fail,
  i := E(4),
  pi := fail,
  based_integer := fail,
  gamma := fail,
  infinity := infinity,
  NaN := fail,
  rational := function ( x )
        return OMgapId( [ OMgap2ARGS( x ), x[1] / x[2] ] )[2];
    end )
```

`OMsymRecord` contains all symbols for which conversion from OpenMath to GAP is supported except some special symbols related with errors and special procedures from the SCSCP package which are treated separately.

To check quickly if GAP can parse a given OpenMath object, copy the OpenMath code and paste it directly into standard input after the following commands:

Example

```
gap> stream := InputTextUser();;
```

```
gap> g := OMGetObject(stream);CloseStream(stream);
```

The main tool for the conversion from GAP to OpenMath is `OMPut( <stream>, <object> )`. A number of methods for `OMPut` are installed in the file `openmath/gap/omput.gi`.

To check quickly whether the object may be converted to OpenMath, call `OMprint` for that object, for example:

Example

```
gap> OMPrint( [ [1..10], ZmodnZObj(2,6), (1,2) ] );
<OMOBJ>
        <OMA>
                <OMS cd="list1" name="list"/>
                <OMA>
                        <OMS cd="interval1" name="integer_interval"/>
                        <OMI> 1</OMI>
                        <OMI> 10</OMI>
                </OMA>
                <OMA>
                        <OMS cd="integer2" name="class"/>
                        <OMI> 2</OMI>
                        <OMI> 6</OMI>
                </OMA>
                <OMA>
                        <OMS cd="permut1" name="permutation"/>
                        <OMI> 2</OMI>
                        <OMI> 1</OMI>
                </OMA>
        </OMA>
</OMOBJ>
```

The package is in the continuous development and will support even more symbols in the future. In the meantime, if you will have any requests for the support for particular symbols, please write to Alexander Konovalov who currently develops and maintains the package.

3.2 Adding support for private content dictionaries and symbols

There is also a way for the user to extend the package adding support for private and experimental CDs or separate symbols. We allocated the directory `openmath/private` for this purposes, and currently it contain the file `private.g` for conversions from OpenMath to GAP and the file `private.gi` for conversions from GAP to OpenMath for some symbols from private CDs contained in the `openmath/cds` directory.

In particular, we extended the package with the following private OpenMath symbols:

- `group1.group_by_generators` which allows us to input and output groups given by their generators as this is a natural way to create groups in GAP;
- `semigroup1.semigroup_by_generators` and `monoid1.monoid_by_generators` following the same considerations for semigroups and monoids;

- pcgroup1.pcgroup_by_pcgscode for PcGroups given by their pcgs code and order;
- record1.record for records as they are important data structures which we want to pass in a straightforward manner between different GAP instances;
- transform1.transformation to support transformations, transformation semigroups and their automorphism groups.

The file private.g is loaded from openmath/gap/new.g, and the private.gi is loaded from openmath/gap/read.g. If the user would like to add own code, this may be done either by adding it to these files or by placing additional files in openmath/private directory and load them similarly to private.g and private.gi. We will welcome user's contributions in the form of the code to support existing content dictionaries from the OpenMath web site or private content dictionaries, if they may be interesting for a wider community.

References

[FHK+a] Sebastian Freundt, Peter Horn, Alexander Konovalov, Sylla Lesseni, Steve Linton, and Dan Roozemond. OpenMath content dictionary `scscp1`. (`http://www.win.tue.nl/SCIEnce/cds/scscp1.html`). 4

[FHK+b] Sebastian Freundt, Peter Horn, Alexander Konovalov, Sylla Lesseni, Steve Linton, and Dan Roozemond. OpenMath content dictionary `scscp2`. (`http://www.win.tue.nl/SCIEnce/cds/scscp2.html`). 4

[FHK+c] Sebastian Freundt, Peter Horn, Alexander Konovalov, Steve Linton, and Dan Roozemond. Symbolic Computation Software Composability Protocol (SCSCP) specification, version 1.3, 2009. (`http://www.symbolic-computation.org/scscp`). 4

[KL] Alexander Konovalov and Steve Linton. SCSCP — Symbolic Computation Software Composability Protocol. GAP4 package (`http://www.cs.st-andrews.ac.uk/~alexk/scscp.htm`). 4

[LN] Frank Lübeck and Max Neunhöffer. GAPDoc — A Meta Package for GAP Documentation. GAP4 package (`http://www.math.rwth-aachen.de/~Frank.Luebeck/GAPDoc`). 4

Index