From: **Steve Linton** sal@dcs.st-and.ac.uk
Subject: Re: Domain Constructors
Date: 10 September 1996 at 09:18
To: Thomas Breuer thomas.breuer@math.rwth-aachen.de
Cc: gap-dev@math.rwth-aachen.de

Thomas writes:

> Concerning the arguments of 'MagmaWithOneGenerators', Steve wrote
>
> > Would it not be easier to disallow the third option:
> >
> > MagmaWithOneGenerators(<homogenous-coll-of-generators>)
>
> The problem with this would be (as it is in GAP-3) that one is forced to construct the identity. For f.p. magmas this is cheap, but for matrix groups one definitely wants to avoid it.

> So we need the possibility to omit the identity in 'MagmaWithOneGenerators'.

Ah well. You are clearly right.

I think this example does speak for allowing an Operation to have several arities though.

We could have something like:

```
        NewOperation("MagmaWithOneGenerators", [[ IsCollection ],
                                               [IsList , IsMWOElement ]]);
```

or perhaps better:

```
MagmaWithOneGenerators :=
        NewOperation("MagmaWithOneGenerators", [ IsCollection ]);

AddAllowedArguments( MagmaWithOneGenerators,                                   [IsList ,
IsMWOElement ]);
```

InstallMethod would then be allowed providing that the requirements matched one of the sets of allowed arguments of the operation.

This would restrict InstallOtherMethod to really rare situations where a method did not fit into ANY general framework for the Operation.

Steve

From: **Thomas Breuer** thomas.breuer@math.rwth-aachen.de
Subject: Re: Re: Domain Constructors
Date: 10 September 1996 at 09:08
To: gap-dev@math.rwth-aachen.de

Concerning the arguments of 'MagmaWithOneGenerators', Steve wrote

> Would it not be easier to disallow the third option:
>
> MagmaWithOneGenerators(<homogenous-coll-of-generators>)
>
> for MagmaWithOneGenerators?. Very little is lost, as this is usually called
> from MagmaWithOne anyway, and MWOG then becomes a nice regular Operation
> with two arguments of consistent meaning.

The problem with this would be (as it is in GAP-3) that one is forced to construct the identity. For f.p. magmas this is cheap, but for matrix groups one definitely wants to avoid it.
(Analogously, one wants to avoid constructing large zero objects.)

Also 'MagmaWithOneGenerators' (or better 'MagmaWithOneByGenerators'?) is the function that will be used in the library, probably at most the interactive user will call 'MagmaWithOne'.
So we need the possibility to omit the identity in 'MagmaWithOneGenerators'.

Thomas

From: **Steve Linton** sal@dcs.st-and.ac.uk
Subject: Re: Domain Constructors
Date: 10 September 1996 at 08:37
To: Martin Schoenert mschoene@hobbes.math.rwth-aachen.de
Cc: gap-dev@math.rwth-aachen.de

Would it not be easier to disallow the third option:

MagmaWithOneGenerators(<homogenous-coll-of-generators>)

for MagmaWithOneGenerators?. Very little is lost, as this is usually called from MagmaWithOne anyway, and MWOG then becomes a nice regular Operation with two arguments of consistent meaning.

Steve

From: **Martin Schoenert** mschoene@hobbes.math.rwth-aachen.de
Subject: Domain Constructors
Date: 10 September 1996 at 07:26
To: gap-dev@math.rwth-aachen.de

Domain Constructors
===================

One important class of constructors are the constructors that construct domains from generators lists. This document describes 'MagmaWithOne' and 'MagmaWithOneGenerators' as an example of how such constructors should look and what their methods should do.

MagmaWithOne

```
MagmaWithOne( <empty-list-of-generators>,      <one> )
MagmaWithOne( <homogenous-coll-of-generators>, <one> )
MagmaWithOne( <homogenous-coll-of-generators> )
MagmaWithOne( <generator1>, <generator2>... )
```

'MagmaWithOne' returns the smallest magma with one containing all the generators respectively the one (if the generators list is empty).

'MagmaWithOne' is an operation accepting a variable number of arguments. It decides how it was called (the difficult cases are 1 or 2 lists as arguments, because the lists can either be lists of generators or matrices). Then it calls 'MagmaWithOneGenerators'.

MagmaWithOneGenerators

```
MagmaWithOneGenerators( <empty-list-of-generators>,      <one> )
MagmaWithOneGenerators( <homogenous-coll-of-generators>, <one> )
MagmaWithOneGenerators( <homogenous-coll-of-generators> )
```

'MagmaWithOneGenerators' returns the smallest magma with one containing all the generators respectively the one (if the generators list is empty).

'MagmaWithOneGenerators' is an operation accepting 2 or 1 arguments (The 2 argument case is the ``proper'' case, the 1 argument case is the ``other'' case).

Methods for MagmaWithOneGenerators

The methods for 'MagmaWithOneGenerators' should take the following actions.

1) Test that the generators (and the one) lie in the same family. This test can be done via method selection of course (i.e. by installing the method appropriately it will only be called if the generators lie in the same family).

2) Test that the generators are multiplicative elements with one (if the generators lie in a family of multiplicative elements with one this test

can be done via method selection).

3) Test that the generators (and the one) fit together (e.g. are square
matrices of the same shape).

4) Create a record <magma> and objectify it with
'NewKind(FamilyObj(<gens>), IsMagmaWithOne and IsAttributeStoringRep)'
(unless the magma is represented in an other representation of course).

5) Set the generators attribute with
'SetGeneratorsMagmaWithOne(<magma>, AsList(<generators>))'.
The 'AsList' guarantees that the generators list is a proper list
and that it is immutable, i.e. safe against changes by the caller.
If <generators> is already an immutable list, then 'AsList' does nothing.
Set the one attribute with 'SetOne(<magma>, Immutable(<one>))'.
Again the 'Immutable' guarantees that the one is immutable.

6) Return the magma with one.

MagmaWithOneGeneratorsNC

There probably should be an operation 'MagmaWithOneGeneratorsNC'.

'NC' means ``NoCheck'', so the methods would not need to perform any checks.

Does 'NC' also mean ``NoCopy'', i.e., would the methods need to use 'AsList'
and 'Immutable'?

Martin.

-- .- .-. - .. -. .-.. --- ...- . ... .- -. -. .. -.- .-
Martin Sch"onert, Martin.Schoenert@Math.RWTH-Aachen.DE, +49 241 804551
Lehrstuhl D f"ur Mathematik, Templergraben 64, RWTH, 52056 Aachen, Germany