From: **Martin Schoenert** mschoene@hobbes.math.rwth-aachen.de
Subject: Re: Mailing list, Constructors, Info, ...
Date: 10 September 1996 at 07:22
To: gap-dev@math.rwth-aachen.de

Steve wrote

> I don't quite see how the ValueFilter parts work. In the example, IsQuaternionJ1Rep and IsQuaternionRep are both direct children of IsPositionalRep, and so would surely have equal value. I think we are probably OK using explicit numerical ranks here. At worst, locally correct code may not be globally most efficient, but such cases could be fixed by hand.

Sorry forgot that. My idea was to introduce properties 'IsQuaternionJ0' and 'IsQuaternionJ1' and have the representations imply those properties. Then they would be more valuable than 'IsQuaternionRep' because of those implications. And of course there is also the idea of giving categories, properties, and representations different values once we have more experience how well equal valuation for all categories, properties, and representations works.

Steve continued

> Is NewConstructor simply a synonym of NewOperation?

Yes, currently it would be.

Steve continued

> I have done a GAP-level simulation of the proposed Info keyword (although obviously this version does evaluate the arguments before not printing them). This required implementing ApplyFunc in the kernel, which leaves me with a question: What file should ApplyFunc be in? I could see cases for calls.c or funcs.c or a file on its own. When this is resolved I'll check it all in.

'ApplyFunc' is now called 'CallFuncList' and is defined in 'calls.c'.

Martin.

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

From: **Steve Linton** sal@dcs.st-and.ac.uk
Subject: Mailing list, Constructors, Info, ...
Date: 9 September 1996 at 09:53
To: gap4-dev@keith.dcs.st-and.ac.uk

Hi everyone,

I have set up an alias gap4-dev@keith.dcs.st-and.ac.uk for technical GAP4 discussion.

Constructors
============

Martin's basic proposals seem sensible, as far as I understand them.

I don't quite see how the ValueFilter parts work. In the example, IsQuaternionJ1Rep and IsQuaternionRep are both direct children of IsPositionalRep, and so would surely have equal value. I think we are probably OK using explicit numerical ranks here. At worst, locally correct code may not be globally most efficient, but such cases could be fixed by hand.

I think Martin is right that restricting the representation should be done by
having multiple constructors.

Is NewConstructor simply a synonym of NewOperation?

Info
====

I have done a GAP-level simulation of the proposed Info keyword (although obviously this version does evaluate the arguments before not printing them). This required implementing ApplyFunc in the kernel, which leaves me with a question: What file should ApplyFunc be in? I could see cases for calls.c or funcs.c or a file on its own. When this is resolved I'll check it all in.

I have done the Info stuff using a new family: InfoClassFamily with a suitable
representation and methods, for what are basically just wrapped small ints. An InfoSelector is just a Set of InfoClasses, and Info takes an InfoSelector, a level and zero or more arguments to print.

This works nicely, but means that Info can't be used in four or five library files that need to be loaded before info.g. Can anyone suggest a better approach? I append the file, which won't work until ApplyFunc is sorted.

On trivial point, I noticed that \or is not an operation. One can see why, because of the special evaluation rules that apply, but it would sometimes be nice. I wanted to 'or' together InfoClasses and ended up having to '+' them, which is less nice.

Steve

Here is info.g:

```
#############################################################################
##
#W  info.g                    GAP library                    Steve Linton
##
#H  @(#)$Id:$
##
#Y  Copyright (C)  1996,  Lehrstuhl D fuer Mathematik,  RWTH Aachen,  Germany
##
##  This file sets up a GAP level prototype of the new Info messages
##  system, parts of which will eventually have to be moved into
##  the kernel
##
##  Informational messages are controlled by the user setting desired
##  levels of verbosity for InfoClasses. A set of InfoClasses is an
##  InfoSelector, and classes and selectors may be built up with \+
#N  I wanted to use \or, but this isn't at operation
##
##  An message is associated with a selector and a level and is
##  printed when the desired level for any of the classes in the selector
##  equals or exceeds  the level of the message
##
##  Since there is just one representation of info classes, and no plans
##  for more, or for generic methods, this file is not split into
##  declarations, generic methods and representation-specific stuff,
##  not yet, anyway
##
#N  There may be a case for doing this without using method selection at all
#N  as it could then be installed earlier in library loading
#N
Revision.info_g :=
  "@(#)$Id:$";


#############################################################################
##
#R  IsPositionalObjectRep(<obj>)                 new name for IsLObjectRep
##
##
#N  This name is due to change, when it does, delete this section
##

IsPositionalObjectRep := IsLObjectRep;


#############################################################################
##
#C  IsInfoClass(<obj>)                         the category of Info Classes
##

IsInfoClass := NewCategory("IsInfoClass", IsObject);

#############################################################################
##
#V  InfoClassFamily                              the family of Info Classes
##

InfoClassFamily := NewFamily("InfoClassFamily", IsInfoClass, IsInfoClass);

#############################################################################
##
#C  IsInfoSelector(<obj>)                the category of sets of InfoClasses
##
##  Such sets are what we actually use in message selection
##


IsInfoClassCollection :=
IsInfoClassCollection :=
 NewCategoryCollections("IsInfoClassCollection",IsInfoClass);

IsInfoSelector := IsInfoClassCollection and IsSet;

#############################################################################
##
#R  IsInfoClassListRep(<obj>)                    length one positional rep
##
##  An InfoClass is represented as a length one positional object with
##  a positive integer in ic![1]. No other representations are
##  anticipated
##

IsInfoClassListRep := NewRepresentation("IsInfoClassListRep",
                             IsPositionalObjectRep,[]);

#############################################################################
##
#V  CurrentInfoLevels        the current desired verbosity levels set by user
#V  InfoClassNames              names of all info classes, used for printing
##
##  these are both lists, and should have the same length, which defines the
##  number of Info classes that exist
##


CurrentInfoLevels := [];
InfoClassNames := [];

#############################################################################
##
#O  InfoClass( <num> )         make a number into the corresponding InfoClass
##
##  This function is not intended for user consumption, users should use
##  NewInfoClass( <name> )  and then remember the result in a global
##
#N  Does this need to be an Operation? is it ever going to have another
#N  method?
##

InfoClass := NewOperation("InfoClass",
                 [IsInt and IsPosRat] );


#############################################################################
##
#M  InfoClass( <num> )         make a number into the corresponding InfoClass
##


InstallMethod(InfoClass, true,
      [ IsInt and IsPosRat ], 0,
      function(num)
  if num > Length(CurrentInfoLevels) then
      Error("There are not that many Info Classes");
  fi;
  return Objectify(NewKind(InfoClassFamily,IsInfoClassListRep),
            [num]);
end);

###########################################################################
##
#F  NewInfoClass( <name> )                       make a new Info Class
##
##  This is how Info Classes should be obtained
##
#N  Is there any reason to make this an Operation?
##

NewInfoClass:= function( name )
  if not IsString(name) then
      Error("Usage: NewInfoClass(<name>)");
  fi;
  Add(CurrentInfoLevels,0);
  Add(InfoClassNames,name);
  return InfoClass(Length(CurrentInfoLevels));
end;

###########################################################################
##
#M  Basic methods for Info Classes: =, < (so that we can make Sets),
##                                  PrintObj
##

InstallMethod(\=, IsIdentical, [IsInfoClassListRep, IsInfoClassListRep], 0,
      function(ic1,ic2)
  return ic1![1] = ic2![1];
end);

InstallMethod(\<, IsIdentical, [IsInfoClassListRep, IsInfoClassListRep], 0,
      function(ic1,ic2)
  return ic1![1] < ic2![1];
end);

InstallMethod(PrintObj, true, [IsInfoClassListRep], 0,
      function(ic)
  Print(InfoClassNames[ic![1]]);
end);

###########################################################################
##
#M  <info class> + <info class>
#M  <info selector> + <info class>
#M  <info class> + <info selector>
#M  <info selector> + <info selector
##
##  Used to build up InfoSelectors, these are essentially just taking unions
##

InstallOtherMethod(\+, IsIdentical, [IsInfoClass, IsInfoClass], 0,
      function(ic1,ic2)
  return Set([ic1,ic2]);
end);
```

```
InstallOtherMethod(\+, true, [IsInfoClass, IsInfoSelector], 0,
      function(ic,is)
  return Union(is,[ic]);
end);

InstallOtherMethod(\+, true, [IsInfoSelector, IsInfoClass], 0,
      function(is,ic)
  return Union(is,[ic]);
end);

InstallOtherMethod(\+, IsIdentical, [IsInfoSelector, IsInfoSelector], 0,
      function(is1,is2)
  return Union(is1,is2);
end);

##############################################################################
##
#O  SetInfoLevel( <class>, <level>)   set desired verbosity level for a class
##
#N  Is it sensible to SetInfoLevel for a selector?
#N  Would RaiseInfoLevel (which would not lower it if it were already higher
#N  or LowerInfoLevel (the opposite) be any use?
##

SetInfoLevel := NewOperation("SetInfoLevel",
                    [IsInfoClass, IsInt and IsPosRat]);

##############################################################################
##
#M  SetInfoLevel( <class>, <level>)   set desired verbosity level for a class
##

InstallMethod(SetInfoLevel, true,
      [IsInfoClassListRep, IsInt and IsPosRat], 0,
      function(ic,lev)
  CurrentInfoLevels[ic![1]] := lev;
end);

##############################################################################
##
#O  InfoLevel( <class> )             get desired verbosity level for a class
##
#N Does this make sense for a selector (gets the max of the levels for the
#N classes that make up the selector, presumably)?
#N
#N In the final version, this will have to be done directly, not via an
#N Operation, as the kernel needs to do it quickly, and for a whole selector
##

InfoLevel := NewOperation("InfoLevel",
                    [IsInfoClass]);

##############################################################################
##
#M  InfoLevel( <class> )             get desired verbosity level for a class
##

InstallMethod(InfoLevel, true,
InstallMethod(InfoLevel, true,
      [IsInfoClassListRep], 0,
      function(ic)
  return CurrentInfoLevels[ic![1]];
end);

#############################################################################
##
#F  Info( <selector>, <level>, <data> ... )         possiby print information
##
##  If the desired verbosity level for any of the classes making up selector
##  is equal to or greater than level, then this function will Print "#I  "
##  then do ApplyFunc(Print, <data> ...) (where <data> may be multiple
##  arguments, then Print a newline
##
##  Eventually, if the message is not to be printed, then the third and
##  subsequent arguments will not be evaluated either, but this means making
##  Info into a keyword
##



Info := function(arg)
  local selectors, level, usage;

  # Check and unpack the arguments
  usage := "Usage : Info(<selectors>, <level>, <data>...)";
  if Length(arg) < 3 then
      Error(usage);
  fi;
  selectors := arg[1];
  level := arg[2];
  if not IsInfoSelector(selectors)
     or not IsInt(level)
     or level < 0
  then
      Error(usage);
  fi;

  # Now decide what to do and then do it,
  # note that we 'or' the classes together
  if ForAny(selectors, ic -> InfoLevel(ic) >= level) then
      Print("#I  ");
      ApplyFunc(Print, arg{[3..Length(arg)]});
      Print("\n");
  fi;
end;




#############################################################################
##
#E  info.g ends here
##
```