

# Symbolic Computation Software Composability Protocol and its **GAP** implementation

Alexander Konovalov<sup>3</sup> (joint work with Sebastian Freund<sup>1</sup>,  
Peter Horn<sup>2</sup>, Steve Linton<sup>3</sup> and Dan Roozmond<sup>4</sup>)

<sup>1</sup>Technical University of Berlin

<sup>2</sup>University of Kassel

<sup>3</sup>University of St Andrews

<sup>4</sup>Technical University of Eindhoven

Supported by the EU FP6 project 26133 "**SCIENCE** – **S**ymbolic **C**omputation **I**nfrastructure for **E**urope"

2<sup>nd</sup> SCIENCE Workshop, Palaiseau, France, 19-21 January, 2009



# Outline

- SCSCP - a simple OpenMath RPC framework
- Some OpenMath-related design aspects of SCSCP
- SCSCP implementation in GAP with examples



# Composability

## Goals:

- to deliver robust, usable and flexible extensions to our systems allowing any system to be easily used as a server or a client for both general and problem-specific services.
- to allow these services to be provided as “standard” Web services
- to be open to connections to other clients and servers, both other powerful mathematical systems and special-purpose programs

## Most common restrictions which can be seen from some existing previous attempts

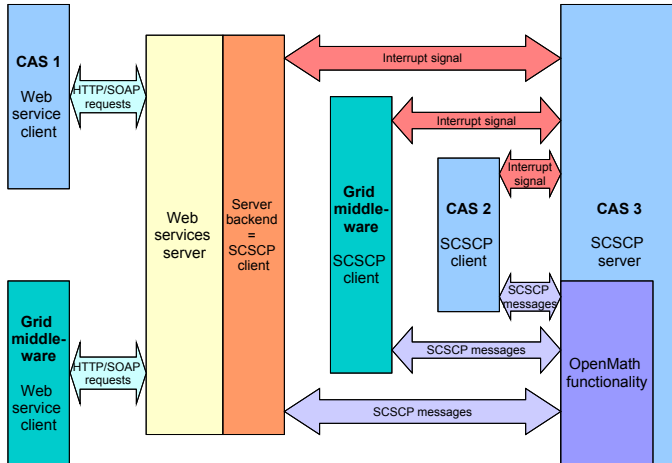
- interfaces do not support remote communication
- transmission of large or complex objects may be difficult
- Support of new system requires new I/O convertor. It relies upon the I/O format, may be subject to parsing errors and needs update if I/O format of the linked system changes
- not enough deeply (syntax, cd) and widely (other CAS) supported data encoding format (OpenMath)
- not interactive, just database access (Web services)
- not enough robust (ParGAP)
- less efficient for irregular parallel computing (ParGAP)
- shaped to deal with the particular problem (dc)
- may not work in some operating systems
- may be not easy customisable by the end-user

# Features of Composability Model

- A lightweight sockets based RPC protocol (the **S**ymbolic **C**omputation **S**oftware **C**omposability **P**rotocol)
  - SCSCP can be used directly for communication between systems
  - SCSCP uses OpenMath for data encoding
  - SCSCP implementation stays mainly within systems, rather than in wrappers
- SCSCP servers and clients can be “translated” into standard Web services servers and clients through Java proxy programmes common for all SCSCP framework
- Grid middleware (SymGrid-Par) also uses SCSCP to talk to CAS



# The Big Picture



# SCSCP: OpenMath inside

- The feature of SCSCP is that protocol messages represented as OpenMath objects using OpenMath content dictionaries **scscp1**, **scscp2** developed for this purpose
- **SCSCP** specification defines semantical and technical descriptions and allowed sequences of OpenMath-encoded messages to and from CAS:
  - remote procedure call
  - returning result of successfully completed procedure
  - returning a signal about procedure termination

## Flexible enough?

- May be limited to functionality/data types for which CDs exist
  - Avoid this by allowing transient CDs, which contain symbols specific to that service, obtainable from the server on request
- Encoding may be unreasonably bulky, or encoding costs may be too high for some applications
  - Perfectly OK for services which pass real data in some private format encoded in an `OMSTRING`, `OMBYTES` or `OMFOREIGN` element, if that suits the application.
  - Also working on new CDs for efficient representation of some common cases (eg matrices over finite fields)

# GAP implementation of the SCSCP

- GAP Package SCSCP (in development):  
`http://www.cs.st-andrews.ac.uk/~alexk/scscp.htm`
- Allows GAP to work as an SCSCP server and client over the TCP/IP protocol
- Uses GAP packages IO, GAPDoc and OpenMath.dev
- Needs functionality for the exception and error handling in GAP.dev added by Steve Linton (will appear in GAP 4.5)
- We may provide on demand a client's version for GAP 4.4.12, which works in Linux, Windows and Mac OS
- Extends GAP package OpenMath with:
  - new symbols from scscp1 and scscp2 OM CDs
  - support of OM attributes (OMATTR, OMATP)
  - support of OM references (OMR)



## User-level functionality:

- The underlying technology is well-hidden: the end-user may know nothing about OpenMath and SCSCP !!!
- The service provider installs procedures available as SCSCP services and starts the SCSCP server
- The client sends request to the server and gets back result
- Store/Retrieve procedures allowing to work with remote objects
- InputOutputTCPStreams, compatible with other kinds of streams in GAP

## Configuring SCSCP GAP server

1. Specify some default parameters in

`gap4r4/pkg/scscp/config.g`

2. Put all what you need in the configuration file (e.g. `myserver.g`), including:

- loading all necessary packages
- other required GAP code or its reading from other file(s)
- installation of SCSCP procedures using

```
InstallSCSCPprocedure("NameForClient",  
InternalName );
```

- starting the server with the command

```
RunSCSCPserver( <parameters> );
```

3. Start GAP with `gap myserver.g`

## Examples of simple calls

### myserver.g

```
...  
FactorialAsString := x -> String(Factorial( x ) );  
...  
InstallSCSCPprocedure( "Factorial", Factorial );  
InstallSCSCPprocedure( "WS_Factorial", FactorialAsString );  
...
```

### GAP session

```
gap> EvaluateBySCSCP( "Factorial", [ 10 ], "localhost", 26133 );  
rec( object := 3628800,  
      attributes := [ [ "call_ID", "localhost:26133:27096" ] ] )  
  
gap> EvaluateBySCSCP( "WS_Factorial", [ 10 ], "localhost", 26133 );  
rec( object := "3628800",  
      attributes := [ [ "call_ID", "localhost:26133:27096" ] ] )
```

## Group identification: three possible approaches

### group $\rightarrow$ group id

- client: GAP (slow machine? no small groups library?)
- server: fast and complete GAP installation

### list of permutations $\rightarrow$ group id

- client: CAS which "understands" permutations
- server: complete GAP installation

### pcgs code (integer that encodes the group) $\rightarrow$ group id

- client: GAP in Windows - ANUPQ package is not working :(
- server: GAP in UNIX environment - ANUPQ works :)

## Three approaches (continued)

### group $\rightarrow$ group id

```
InstallSCSCPprocedure( "WS_IdGroup", IdGroup );
```

### list of permutations $\rightarrow$ group id

```
IdGroupByGenerators := function( permlist )  
  return IdGroup( Group( permlist ) );  
end;
```

```
InstallSCSCPprocedure( "GroupIdentificationService", IdGroupByGenerators );
```

### pcgs code (integer that encodes the group) $\rightarrow$ group id

```
IdGroup512ByCode:=function( code )  
  local G, H;  
  G := PcGroupCode( code, 512 ); # recreate the group from code  
  H := PcGroupFpGroup( PqStandardPresentation( G ) );  
  return IdStandardPresented512Group( H );  
end;
```

```
InstallSCSCPprocedure( "IdGroup512ByCode", IdGroup512ByCode );
```

# How it works

We need an auxiliary function for the client ...

## Client's counterpart for the 3rd example

```
IdGroup512:=function( G )
local code, result;
if Size( G ) <> 512 then
  Error( "G must be a group of order 512 !!!\n" );
fi;
code := CodePcGroup( G );
result := EvaluateBySCSCP( "IdGroup512ByCode",
                          [ code ],
                          "scscp.st-and.ac.uk",
                          26133 );

return result.object;
end;
```

## Example: remote objects

- Objects may "live" on the server while the client has only a reference (cookie) pointing to them
- There is no need to transmit objects repeatedly over the network
- The client may be a CAS which does not have objects of that type at all
- The client may retrieve object in its default OpenMath representation (be careful about its subobjects!)
- Also, the actual object may be deleted from the server

## Example: stateful SCSCP service

Let  $G = \langle \sigma_1, \sigma_2, \dots, \sigma_n \rangle$  be a permutation group. For the orbit computation, we would like a service that takes a point  $k$  and returns a set of all distinct images of  $k$  under  $\sigma_1, \sigma_2, \dots, \sigma_n$

### Code for the server

```
PointImages := function( G, n )
  local g;
  return Set( List( GeneratorsOfGroup(G), g -> n^g ) );
end;

InstallSCSCPprocedure( "PointImages", PointImages );
```

Now we can create the group  $G$  on the server as a remote object and next time pass only the reference to it!



# Processes

- `EvaluateBySCSCP` uses a combination of `NewProcess` and `CompleteProcess`
- These works with new objects - *processes*
- Process is associated with corresponding *InputOutputTCPStream*
- You may start process, send a request and collect the result later, doing something else in the meantime
- There are functions for:
  - processes synchronization
  - waiting until the first available result and abandoning remaining processes
- We can terminate process locally (working on remote termination)

# Parallelizing existing code with SCSCP is user-friendly

- See example of Karatsuba multiplication for polynomials in the SCSCP package manual
- We have master-slave version of the orbit computation algorithm
- This provides a way of using several versions of GAP on a multi-core machine
- Nevertheless, minimizing overhead might require some more tricks (optimizing data format, better scheduling, etc.)

## Further *SCIENCE*tific work

- Goal is simple to use, robust and flexible client and server implementations for all systems
- First public releases of basic SCSCP implementations are expected in April 2009
- We would be very interested in
  - connecting other systems, or stand-alone programs
  - test problems and use cases
    - for cooperating systems
    - for parallel computation
    - for CA-based Web services

# References



*SCIENCE* — *Symbolic Computation Infrastructure for Europe*, project homepage

<http://www.symbolic-computation.org/>



S. Freundt, P. Horn, A. Konovalov, S. Linton, D. Roozmond. *Symbolic Computation Software Composability Protocol (SCSCP) specification*, Version 1.2, 2008

<http://www.symbolic-computation.org/scscp>



A. Konovalov, S. Linton. *SCSCP* — *Symbolic Computation Software Composability Protocol*, GAP package, in development

<http://www.cs.st-andrews.ac.uk/~alexk/scscp.htm>



D. Roozmond. *OpenMath Content Dictionary scscp1*

<http://www.win.tue.nl/SCIENCE/cds/scscp1.html>



D. Roozmond. *OpenMath Content Dictionary scscp2*

<http://www.win.tue.nl/SCIENCE/cds/scscp2.html>

